Estd:2008

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to Osmania University & Approved by AICTE, New Delhi)

LABORATORY MANUAL

ELECTRICAL SIMUALATION LABORATORY

BE, VII Semester (CBCS): 2020-21

NAME:__________________________________

ROLL NO:_______________________________

BRANCH:_________________________________

SEM:____________________________________

DEPARTMENT OF ELECTRICAL AND ELECTRONCS ENGINEERING

Empower youth- Architects of Future World

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

VISION

To produce ethical, socially conscious and innovative professionals who would contribute to sustainable technological development of the society.

MISSION

To impart quality engineering education with latest technological developments and interdisciplinary skills to make students succeed in professional practice.

To encourage research culture among faculty and students by establishing state of art laboratories and exposing them to modern industrial and organizational practices.

To inculcate humane qualities like environmental consciousness, leadership, social values, professional ethics and engage in independent and lifelong learning for sustainable contribution to the society.

DEPARTMENT
OF
ELECTRICAL AND ELECTRONICS ENGINEERING

LABORATORY MANUAL

ELECTRICAL SIMULATION LABORATORY

Prepared
By
Mr. JARAPALA RAMESH BABU,

Assistant Professor

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

VISION

To become a reputed centre for imparting quality education in Electrical and Electronics Engineering with human values, ethics and social responsibility.

MISSION

- To impart fundamental knowledge of Electrical, Electronics and Computational Technology.
- To develop professional skills through hands-on experience aligned to industry needs.
- To undertake research in sunrise areas of Electrical and Electronics Engineering.
- To motivate and facilitate individual and team activities to enhance personality skills.

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

PROGRAM EDUCATIONAL OBJECTIVES

BE-Electrical Engineering graduates shall be able to:

- **PEO1**. Utilize domain knowledge required for analyzing and resolving practical Electrical Engineering problems.

- **PEO2**.Willing to undertake inter-disciplinary projects, demonstrate the professional skills and flair for investigation.
- **PEO3**. Imbibe the state of the art technologies in the ever transforming technical scenario.

- **PEO4**. Exhibit social and professional ethics for sustainable development of the society.

Estd:2008

METHODIST
COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELCTRONICS ENGINEERING

PROGRAM OUTCOMES

Engineering Graduates will have ability to:

- **PO1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of electrical and electronics engineering problems.
- **PO2. Problem analysis:** Identify, formulate, review research literature, and analyze complex electrical and electronics engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- **PO3. Design/development of solutions:** Design solutions for complex electrical and electronics engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- **PO4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex electrical and electronics engineering activities with an understanding of the limitations.
- **PO6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional electrical and electronics engineering practice.
- **PO7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
- **PO.8 Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the electrical and electronics engineering practice.
- **PO9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES

At the end of BE program Electrical and Electronics Engineering graduates will be able to:

- **PSO1**.Provide effective solutions in the fields of Power Electronics, Power Systems and Electrical Machines using MATLAB/MULTISIM.
- **PSO2.** Design and Develop various Electrical and Electronics Systems, particularly Renewable Energy Systems.
- **PSO3.** Demonstrate the overall knowledge and contribute for the betterment of the society.

Estd:2008

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

I. PREREQUISITE(S):

Level	Credits	Semester	Prerequisites
UG	1	1	Electrical simulation

II. SCHEME OF INSTRUCTIONS

Lectures	Tutorials	Practicals	Credits
0	0	2	1

III. SCHEME OF EVALUATION & GRADING

S. No	Component	Duration	Maximum Marks
	Continuous Internal Evaluation (CIE)		
1.	Internal Examination – I and II	1 hour each	25
	CIE (Total)		**25**
2.	**Semester End Examination** (University Examination)	3 hours	**50**
		TOTAL	**75**

%Marks Range	>=90	80 to < 90	70 to < 80	60 to < 70	50 to < 60	40 to < 50	< 40	Absent
Grade	S	A	B	C	D	E	F	Ab
Grade Point	10	9	8	7	6	5	0	-

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

CO No.	Course Outcome	Taxonomy Level
751.1	**Compose (Write)** MATLAB code using some basic commands.	**Create**
751.2	**Develop** MATLAB code for analyzing power system network by obtaining line parameters, Z, Y matrices, and Economics of power systems	**Apply**
751.3	Simulate the concepts of Electrical Circuits, to **design** a led, lag, led and lag compensator and obtain the characteristics by Control Systems and interpret data.	**Create**
751.4	**Demonstrate (Determine)** the knowledge of programming environment, compiling, debugging, linking and executing variety of programs in MATLAB.	**Evaluate**
751.5	Demonstrate ability to **develop** Simulink models for various electrical systems.	**Apply**
751.6	Validate simulated results from programs/Simulink **models** with theoretical calculations.	**Apply**

MAPPING OF COs WITH POs & PSOs

Correlation Level: High – 3; Medium – 2; Low – 1

PO / CO	PO 1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PS01	PSO2	PSO3
C751.1	1	1	2	2	3	-	-	-	1	1	1	-	1	1	-
C751.2	1	1	2	1	1	3	1	1	2	1	1	-	1	1	-
C751.3	1	1	2	2	3	-	-	-	1	1	1	-	1	1	-
C751.4	1	2	3	1	1	1	2	1	1	1	1	-	1	1	-
C751.5	1	1	2	1	1	3	1	1	2	1	1	-	1	1	-
C751.6	1	1	2	1	1	3	1	1	2	1	1	-	1	1	-
C751	**1**	**1**	**2.16**	**1.33**	**1.66**	**2.5**	**1.25**	**1**	**1.5**	**1**	**1**	-	**1**	**1**	-

LABORATORY CODE OF CONDUCT

1. Students should report to the labs concerned as per the scheduled time table.
2. Students, who report late to the labs will not be permitted to perform the experiment scheduled for the day.
3. Students to bring a 100 pages note book to enter the readings /observations while performing the experiment.
4. After completion of the experiment, certification of the staff in-charge concerned, in the observation book is necessary.
5. Staff member in-charge shall evaluate for 25 marks, each experiment, based on continuous evaluation which will be entered in the continuous internal evaluation sheet.
6. The record of observations, along with the detailed procedure of the experiment performed in the immediate previous session should be submitted for certification by the staff member in-charge.
7. Not more than three students in a group would be permitted to perform the experiment on the equipment-based lab set up. However only one student is permitted per computer system for computer-based labs.
8. The group-wise division made at the start of the semester should be adhered to, and no mix up with any other group would be allowed.
9. The components required, pertaining to the experiment should be collected from the stores in-charge, after duly filling in the requisition form / log register.
10. After the completion of the experiment, students should disconnect the setup made by them, and return all the components / instruments taken for the purpose, in order.
11. Any damage of the equipment or burn-out of components will be charged at cost as a penalty or the total group of students would be dismissed from the lab for the semester/year.
12. Students should be present in the lab for the total time duration, as scheduled.
13. Students are required to prepare thoroughly, before coming to Laboratory to perform the experiment.
14. Procedure sheets / data sheets provided to the students, if any, should be maintained neatly and returned after the completion of the experiment.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

DO'S AND DON'TS IN THE LABORATORY

Do's

1. Remove your footwear before you enter the lab.
2. Always keep quiet. Be considerate to other lab users.
3. Report any problems with the computer to the person in charge.
4. Shut down the computer properly.

Don'ts

1. Do not bring any food or drinks in the computer room.
2. Do not touch any part of the computer with wet hands.
3. Do not hit the keys on the computer too hard.
4. Don't damage, remove, or disconnect any labels, parts, cables or equipment.
5. Do not install or download any software or modify or delete any system files on any lab computers.
6. If you leave the lab, do not leave your personal belongings unattended.

Before Leaving Lab:

- Place the stools under the lab bench.
- Turn off the power to all instruments.
- Return all the equipment to lab assistant .
- Turn off the main power switch to the lab bench.
- Please check the laboratory notice board regularly for updates.

METHODIST

COLLEGE OF ENGINEERING AND TECHNOLOGY

Estd:2008

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

CONTENTS

Sl. No.	Name of Experiment	Page No.
1	Verification of Network theorems a. Thevinin's theorem b. Superposition theorem c. Maximum power transfer theorem.	
2	Series and Parallel resonance.	
3	Bode plot, Root-Locus plot and Nyquist plot.	
4	Transfer function analysis (i) Time response for Step input (ii) Frequency response for Sinusoidal input.	
5	Load flow studies	
6	Fault analysis.	
7	Economic Power Scheduling	
8	Design of filters (Low pass filter)	
9	Chopper fed dc motor drives.	
10	VSI /CSI Fed induction motors drives. Doubly fed Induction motor.	
	Additional Experiments	
11	Automatic Generation control	
12	Z-Bus Building algorithm using Matlab Software	

INTRODUCTION TO MATLAB

What is MATLAB?

MATLAB (short for MATrixLABoratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

- Math and computation
- Algorithm development
- Modelling, simulation and prototyping
- Data analysis, exploration and visualization
- Scientific and engineering graphics
- Application development, including Graphical User Interface (GUI) building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non-interactive language such as C, C++ or Fortran.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering and science. In industry, MATLAB is the tool of choice for high-productivity research, development and analysis.

MATLAB features a family of application-specific solution called *Toolboxes*. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB function (m-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, image processing, simulation and many others.

MATLAB System

The MATLAB system consists of five main parts:

1. **The MATLAB language.** This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both “programming in the small” to rapidly create quick and dirty throw-away programs, and “programming in the large” to create complete large and complex application programs.
2. **The MATLAB working environment.** This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB’s applications.
3. **Handle Graphics**. This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.
4. **The MATLAB mathematical function library.** This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and

complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions, and fast Fourier transforms.

5. **The MATLAB Application Program Interface (API).** This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

The Advantages of MATLAB

MATLAB has many advantages compared to conventional computer languages for technical problem solving. Among them are:

1. **Ease of Use**. MATLAB is an interpreted language. Program may be easily written and modified with the built-in integrated development environment and debugged with the MATLAB debugger. Because the language is so easy to use, it is ideal for the rapid prototyping of new programs.
2. **Platform Independence**. MATLAB is supported on many different computer systems, providing a large measure of platform independence. At the time of this writing, the language is supported on Windows NT/2000/XP, Linux, several versions of UNIX and the Macintosh.
3. **Predefined Function**. MATLAB comes complete with an extensive library of predefined functions that provide tested and pre-packaged solutions to many basic technical tasks. For examples, the arithmetic mean, standard deviation, median, etc. these and hundreds of other functions are built right into the MATLAB language, making your job much easier. In addition to the large library of function built into the basic MATLAB language, there are many special-purpose toolboxes available to help solve complex problems in specific areas. There is also an extensive collection of free user-contributed MATLAB programs that are shared through the MATLAB Web site.
4. **Device-Independent Plotting**. Unlike most other computer languages, MATLAB has many integral plotting and imaging commands. The plots and images can be displayed on any graphical output device supported by the computer on which MATLAB is running.
5. **Graphical User Interface**. MATLAB includes tools that allow a programmer to interactively construct a graphical user interface, (GUI) for his or her program. With this capability, the programmer can design sophisticated data-analysis programs that can be operated by relatively inexperienced users.
6. **MATLAB Compiler**. MATLAB's flexibility and platform independence is achieved by compiling MATLAB programs into a device-independent p-code, and then interpreting the p-code instructions at runtime. Unfortunately, the resulting programs can sometimes execute slowly because the MATLAB code is interpreted rather than compiled.

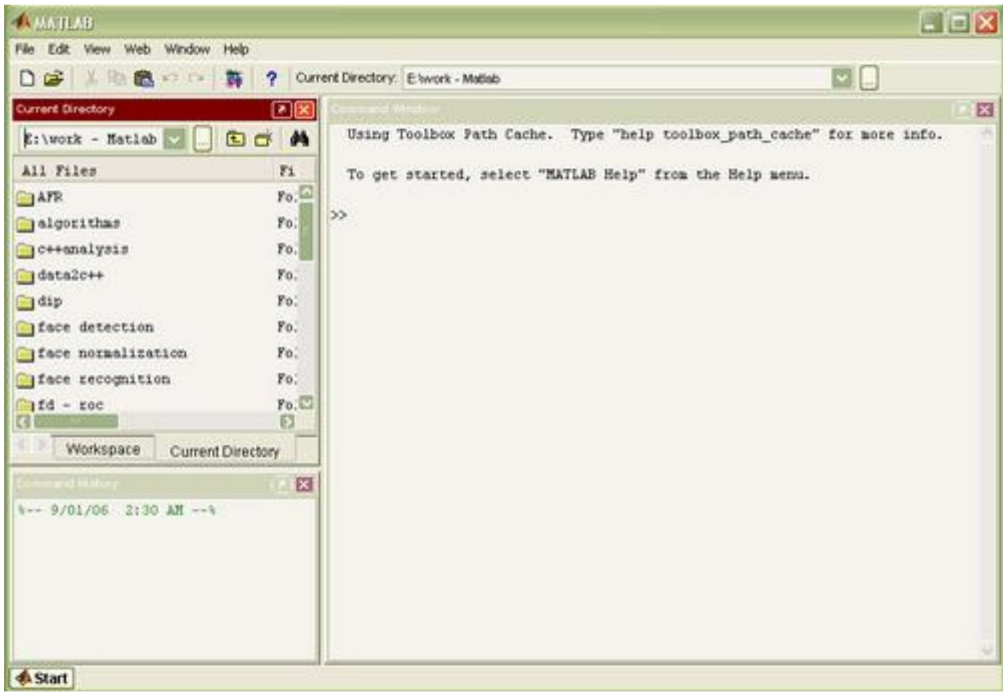
Disadvantages of MATLAB

MATLAB has two principal disadvantages. The first is that it is an interpreted language and therefore can execute more slowly than compiled languages. This problem can be mitigated by properly structuring the MATLAB program, and by the use of the MATLAB compiler to compile the final MATLAB program before distribution and general use.

The second disadvantage is cost: a full copy of MATLAB is five to ten times more expensive than a conventional C or Fortran compiler. This relatively high cost is more than offset by then reduced time required for an engineer or scientist to create a working program, so MATLAB is cost-effective for businesses. However, it is too expensive for most individuals to consider purchasing. Fortunately, there is also an inexpensive Student Edition of MATLAB, which is a great tool for students wishing to learn the language. The Student Edition of MATLAB is essentially identical to the full edition.

Starting MATLAB

You can start MATLAB by double-clicking on the MATLAB icon or invoking the application from the *Start* menu of Windows. The main MATLAB window, called the MATLAB Desktop, will then pop-up and it will look like this:



```
%-- 9/01/06  2:30 AM --%
```

```
Using Toolbox Path Cache.  Type "help toolbox_path_cache" for more info.

To get started, select "MATLAB Help" from the Help menu.

>>
```

Figure 1: The Default MATLAB desktop

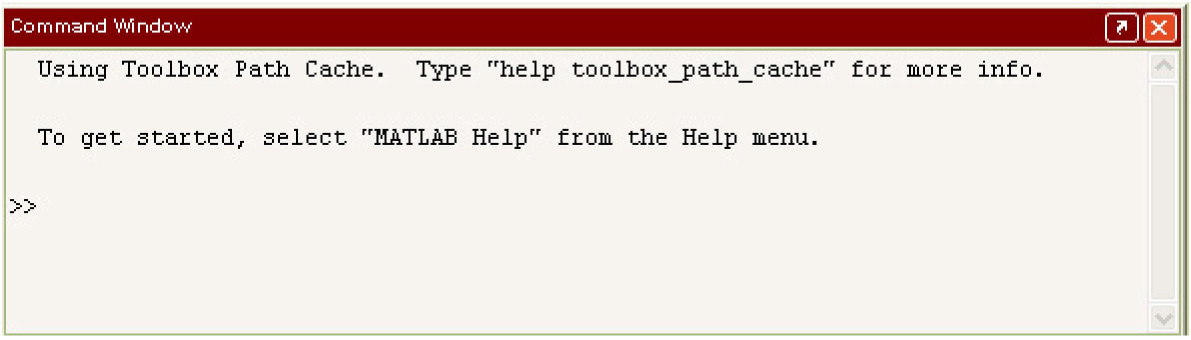
When MATLAB executes, it can display several types of windows that accept commands or display information. It integrates many tools for managing files, variables and applications within the MATLAB environment. The major tools within or accessible from the MATLAB desktop are:

1. The Current Directory Browser
2. The Workspace Window
3. The Command Window
4. The Command History Window
5. The Start Button
6. The Help Window

If desired, this arrangement can be modified by selecting an alternate choice from [View] [Desktop Layout]. By default, most MATLAB tools are “docked” to the desktop, so that they appear inside the desktop window. However, you can choose to “undock” any or all tools, making them appear in windows separate from the desktop.

The Command Window

The *Command Window* is where the command line prompt for interactive commands is located. Once started, you will notice that inside the MATLAB command window is the text:
Click in the command window to make it active. When a window becomes active, its titlebar darkens. The “>>” is called the *Command Prompt*, and there will be a blinking cursor right after it waiting for you to type something. You can enter interactive commands at the command prompt (>>) and they will be executed on the spot.



```
  Using Toolbox Path Cache.  Type "help toolbox_path_cache" for more info.

  To get started, select "MATLAB Help" from the Help menu.

>>
```

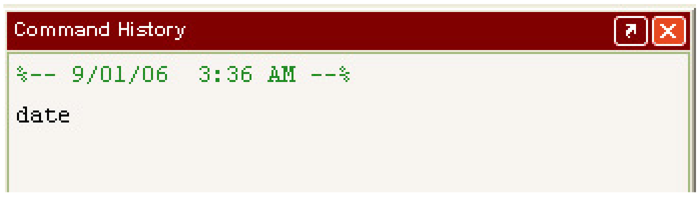
Figure 2: The Command Window

As an example, let's enter a simple MATLAB command, which is the date command. Click the mouse where the blinking cursor is and then type date and press the ENTER key. MATLAB should then return something like this:
Where the current date should be returned to you instead of 01-Sep-2006. Congratulation! You have just successfully executed your first MATLAB command!
To get started, select "MATLAB Help" from the Help menu.

```
>> date
ans = 01-Sep-2006
```

The Command History Window

The *Command History Window*, contains a log of commands that have been executed within the command window. This is a convenient feature for tracking when developing or debugging programs or to confirm that commands were executed in a particular sequence during a multi-step calculation from the command line.

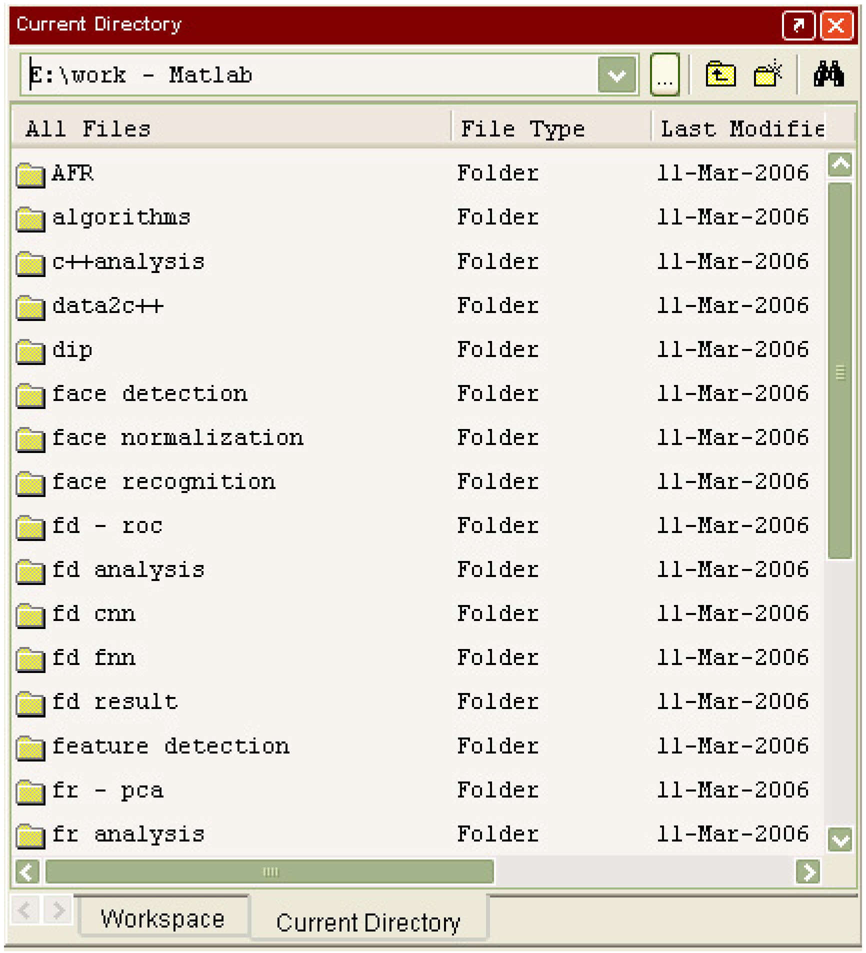


```
%-- 9/01/06  3:36 AM --%
date
```

Figure 3: The Command History Window

The Current Directory Browser

The *Current Directory Browser* displays a current directory with a listing of its contents. There is navigation capability for resetting the current directory to any directory among those set in the path. This window is useful for finding the location of particular files and scripts so that they can be edited, moved, renamed or deleted. The default directory is the *Work* subdirectory of the original MATLAB installation directory



Current Directory

E:\work - Matlab

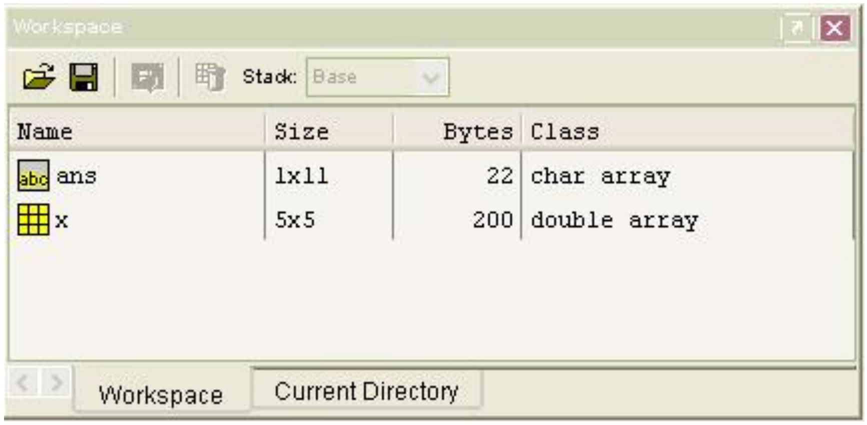
All Files	File Type	Last Modifie
AFR	Folder	11-Mar-2006
algorithms	Folder	11-Mar-2006
c++analysis	Folder	11-Mar-2006
data2c++	Folder	11-Mar-2006
dip	Folder	11-Mar-2006
face detection	Folder	11-Mar-2006
face normalization	Folder	11-Mar-2006
face recognition	Folder	11-Mar-2006
fd - roc	Folder	11-Mar-2006
fd analysis	Folder	11-Mar-2006
fd cnn	Folder	11-Mar-2006
fd fnn	Folder	11-Mar-2006
fd result	Folder	11-Mar-2006
feature detection	Folder	11-Mar-2006
fr - pca	Folder	11-Mar-2006
fr analysis	Folder	11-Mar-2006

Figure 4: The Directory Browser

.

The Workspace Window

The *Workspace Window* provides an inventory of all the items in the workspace that are currently defined, either by assignment or calculation in the Command Window or by importing with a load or similar command from the MATLAB command line prompt.



Workspace

Stack: Base

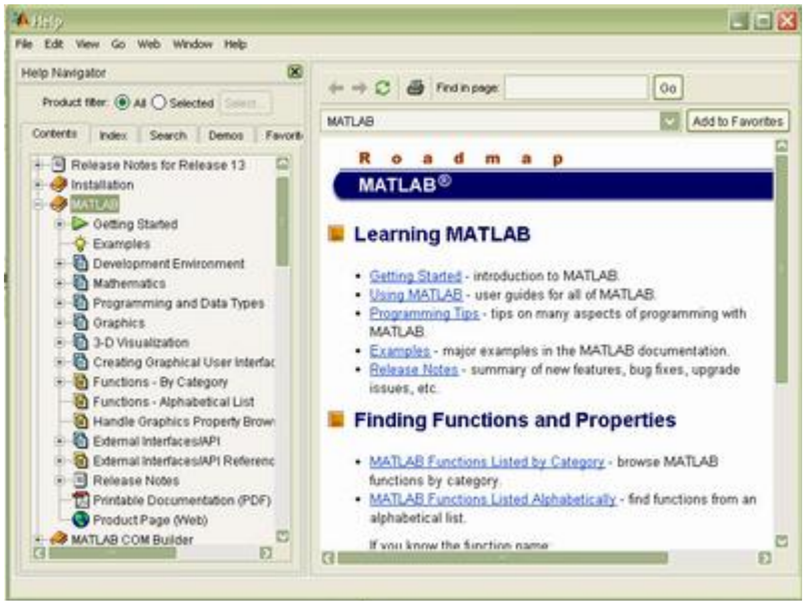
Name	Size	Bytes	Class
ans	1x11	22	char array
x	5x5	200	double array

Figure 5: The Workspace Window

These items consist of the set of arrays whose elements are variables or constants and which have been constructed or loaded during the current MATLAB session and have remained stored in memory. Those which have been cleared and no longer are in memory will not be included. The Workspace Window shows the name of each variable, its value, its array size, its size in bytes, and the class. Values of a variable or constant can be edited in an *Array Editor* which is launched by double clicking its icon in the Workspace Window.

The Help Window

You can access the online help in one of several ways. Typing help at the command prompt will reveal a long list of topics on which help is available. Just to illustrate, try typing help general. Now you see a long list of "general purpose" MATLAB commands. In addition, you can also get help on the certain command. For example, date command.


Help
File Edit View Go Web Window Help
Help Navigator
Product filter: All Selected Select...
Contents Index Search Demos Favorit
Release Notes for Release 13
Installation
MATLAB
Getting Started
Examples
Development Environment
Mathematics
Programming and Data Types
Graphics
3-D Visualization
Creating Graphical User Interfac
Functions - By Category
Functions - Alphabetical List
Handle Graphics Property Brow
External Interfaces/API
External Interfaces/API Referenc
Release Notes
Printable Documentation (PDF)
Product Page (Web)
MATLAB COM Builder
Find in page:
Go
MATLAB
Add to Favorites

R o a d m a p

MATLAB®

Learning MATLAB

- Getting Started - introduction to MATLAB.
- Using MATLAB - user guides for all of MATLAB.
- Programming Tips - tips on many aspects of programming with MATLAB.
- Examples - major examples in the MATLAB documentation.
- Release Notes - summary of new features, bug fixes, upgrade issues, etc.

Finding Functions and Properties

- MATLAB Functions Listed by Category - browse MATLAB functions by category.
- MATLAB Functions Listed Alphabetically - find functions from an alphabetical list.

If you know the function name

Figure 6: The Help Window

The output of help also refers to other functions that are related. In this example the help also tells you, See also NOW, CLOCK, DATENUM. You can now get help on these functions using the three different commands as well

```
>> help date
DATE Current date as date string.
S = DATE returns a string containing the date in dd-mmm-yyyy format.
See also NOW, CLOCK, DATENUM.
```

There is a much more user-friendly way to access the online help, namely via the MATLAB *Help Browser*. Separate from the main desktop layout is a Help desktop with its own layout. This utility can be launched by selecting [Help] □ [MATLAB Help] from the Help pull down menu. This Help desktop has a right side which contains links to help with functions, help with graphics, and tutorial type documentation.

The Start Button

The *Start Button* (see figure 1) allows a user to access MATLAB tools, desktop tools, help files, etc. it works just like the Start button on a Windows desktop. To start a particular tool, just click on the Start Button and select the tool from the appropriate sub-menu.

Interrupting Calculations

If MATLAB is hung up in a calculation, or is just taking too long to perform an operation, you can usually abort it by typing [CTRL + C] (that is, hold down the key labeled CTRL, and press C).

Ending a Session

One final note, when you are all done with your MATLAB session you need to exit MATLAB. To exit MATLAB, simply type quit or exit at the prompt. You can also click on the special symbol that closes your windows (usually an × in the upper right-hand corner). Another way to exit is by selecting [File] [Exit MATLAB]. Before you exit MATLAB, you should be sure to save any variables, print any graphics or other files you need, and in general clean up after yourself.

MATLAB Commands and Functions

General Purpose Commands

Table.1: Operators and Special Characters:

Operators and Special Characters	
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.\	Array left-division operator.
./	Array right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; enclosures array elements.
.	Decimal point.
…	Ellipsis; line-continuation operator.
,	Comma; separates statements and elements in a row.
;	Semicolon; separates columns and suppresses display.
%	Percent sign; designates a comment and specifies formatting.
_	Quote sign and transpose operator.
._	Nonconjugated transpose operator.
=	Assignment (replacement) operator.
Commands for Managing a Session	
clc	Clears Command window.
clear	Removes variables from memory.
exist	Checks for existence of file or variable.
global	Declares variables to be global.
help	Searches for a help topic.
lookfor	Searches help entries for a keyword.
quit	Stops MATLAB.
who	Lists current variables.
whos	Lists current variables (long display).
Special Variables and Constants	
ans	Most recent answer.
eps	Accuracy of floating-point precision.
i,j	The imaginary unit -1.
Inf	Infinity.

NaN	Undefined numerical result (not a number).
pi	The number π .
System and File Commands	
cd	Changes current directory.
date	Displays current date.
delete	Deletes a file.
diary	Switches on/off diary file recording.
dir	Lists all files in current directory.
load	Loads workspace variables from a file.
path	Displays search path.
pwd	Displays current directory.
save	Saves workspace variables in a file.
type	Displays contents of a file.
what	Lists all MATLAB files in the current directory.
wklread	Reads .wk1 spreadsheet file.

Table.2:Input/output and Formatting Commands:

Input/Output Commands	
disp	Displays contents of an array or string.
fscanf	Read formatted data from a file.
format	Controls screen-display format.
fprintf	Performs formatted writes to screen or file.
input	Displays prompts and waits for input.
;	Suppresses screen printing.
Format Codes for fprintf and fscanf	
%s	Format as a string.
%d	Format as an integer.
%f	Format as a floating point value.
%e	Format as a floating point value in scientific notation.
%g	Format in the most compact form: %f or %e.
\n	Insert a new line in the output string.
\t	Insert a tab in the output string.
Numeric Display Formats	
format short	Four decimal digits (default).
format long	16 decimal digits.
format short e	Five digits plus exponent.
format long e	16 digits plus exponents.
format bank	Two decimal digits.
format +	Positive, negative, or zero.
format rat	Rational approximation.
format compact	Suppresses some line feeds.
format compact	X Resets to less compact display mode.

Table.3:Vector, Matrix and Array Commands:

Array Commands	
cat	Concatenates arrays.
find	Finds indices of nonzero elements.
length	Computers number of elements.
linspace	Creates regularly spaced vector.
logspace	Creates logarithmically spaced vector.
max	Returns largest element.
min	Returns smallest element.
prod	Product of each column.
reshape	Change size
size	Computes array size.

Sort	Sorts each column.
sum	Sums each column.
Special Matrices	
eye	Creates an identity matrix.
ones	Creates an array of ones.
zeros	Creates an array of zeros.
Matrix Arithmetic	
cross	Computes cross products.
dot	Computes dot products.
Matrix Commands for Solving Linear Equations	
det	Computes determinant of an array.
inv	Computes inverse of a matrix.
pinv	Computes pseudoinverse of a matrix.
rank	Computes rank of a matrix.
rref	Computes reduced row echelon form.

Table.4: Plotting Commands:

Basic xy Plotting Commands	
axis	Sets axis limits.
fplot	Intelligent plotting of functions.
grid	Displays gridlines.
plot	Generates xy plot.
print	Prints plot or saves plot to a file
title	Puts text at top of plot.
xlabel	Adds text label to x-axis.
ylabel	Adds text label to y-axis.
Plot Enhancement Commands	
axes	Creates axes objects.
close	Closes the current plot.
close all	Closes all plots.
figure	Opens a new figure window.
gtext	Enables label placement by mouse.
hold	Freezes current plot.
legend	Legend placement by mouse.
refresh	Redraws current figure window.
set	Specifies properties of objects such as axes.
subplot	Creates plots in subwindows.
text	Places string in figure.
Specialized Plot Commands	
bar	Creates bar chart.
loglog	Creates log-log plot.
polar	Creates polar plot.
semilogx	Creates semilog plot (logarithmic abscissa).
semilogy	Creates semilog plot (logarithmic ordinate).
stairs	Creates stairs pot.
stem	Creates stem plot.
Three-Dimensional Plotting Commands	
contour	Creates contour plot.
mesh	Creates three-dimensional mesh surface plot.
meshc	Same as mesh with contour plot underneath.
meshz.	Same as mesh with vertical lines underneath
plot3	Creates three-dimensional plots from lines and points.
surf	Creates shaded three-dimensional mesh surface plot.
surfc	Same as surf with contour plot underneath.

meshgrid	Creates rectangular grid.
waterfall	Same as mesh with mesh lines in one direction.
zlabel	Adds text label to *z*-axis.
Histogram Functions	
bar	Creates a bar chart.
hist	Aggregates the data into equally spaced bins.
histc	Aggregates the data into unequally spaced bins.

Table.5: Colors, Symbols and Line Types:

Color	**Symbol**	**Line**
y yellow	. point	- solid
m magenta	o circle	: dotted
c cyan	x x-mark	-. dash dotted
r red	+ plus	-- dashed
g green	* star	
b blue	d diamond	
w white	v triangle (down)	
k black	^ triangle (up)	
	< triangle (left)	
	> triangle (right)	
	p pentagram	
	h hexagram	

Table.6: Programming:

Logical and Relational Operators	
==	Relational operator: equal to.
~=	Relational operator: not equal to.
<	Relational operator: less than.
<=	Relational operator: less than or equal to.
>	Relational operator: greater than.
>=	Relational operator: greater than or equal to.
&	Logical operator: AND.
|	Logical operator: OR.
~	Logical operator: NOT.
xor	Logical operator: EXCLUSIVE OR.
Program Flow Control	
break	Terminates execution of a loop.
case	Provides alternate execution paths within switch structure.
else	Delineates alternate block of statements.
elseif	Conditionally executes statements.
end	Terminates for, while, and if statements.
error	Display error messages.
for	Repeats statements a specific number of times
if	Executes statements conditionally.
otherwise	Default part of switch statement.
return	Return to the invoking function.
switch	Directs program execution by comparing point with case expressions.
warning	Display a warning message.
while	Repeats statements an indefinite number of times.
Logical Functions	
any	True if any elements are nonzero.
all	True if all elements are nonzero.
find	Finds indices of nonzero elements.
finite	True if elements are finite.

isnan	True if elements are undefined.
isinf	True if elements are infinite.
isempty	True if matrix is empty.
isreal	True if all elements are real.
M-Files	
eval	Interpret strings containing Matlab expressions.
feval	Function evaluation.
function	Creates a user-defined function M-file.
global	Define global variables.
nargin	Number of function input arguments.
nargout	Number of function output arguments.
script	Script M-files
Timing	
cputime	CPU time in seconds.
clock	Current date and time as date vector.

Table.7: Mathematical Functions:

Exponential and Logarithmic Functions	
exp(x)	Exponential; ex.
log(x)	Natural logarithm; ln(x).
log10(x)	Common (base 10) logarithm; log(x)= log10(x).
sqrt(x)	Square root; *x*.
Trigonometric Functions	
acos(x)	Inverse cosine; arcos x = cos –1 (x).
acot(x)	Inverse cotangent; arccot x = cot –1(x).
acsc(x)	Inverse cosecant; arcs x = csc –1 (x).
asec(x)	Inverse secant; arcsec x = sec –1 (x).
asin(x)	Inverse sine; arcsin x = sin –1 (x).
atan(x)	Inverse tangent; arctan x = tan –1 (x).
atan2(y,x)	Four-quadrant inverse tangent.
cos(x)	Cosine; cos(x).
cot(x)	Cotangent; cot(x).
csc(x)	Cosecant; csc(x).
sec(x)	Secant; sec(x).
sin(x)	Sine; sin(x).
tan(x)	Tangent; tan(x).
Hyperbolic Functions	
acosh(x)	Inverse hyperbolic cosine; cosh –1 (x).
acoth(x)	Inverse hyperbolic cotangent; coth –1 (x).
acsch(x)	Inverse hyperbolic cosecant; csch –1 (x).
asech(x)	Inverse hyperbolic secant; sech –1 (x).
asinh(x)	Inverse hyperbolic sine; sinh –1 (x).
atanh(x)	Inverse hyperbolic tangent; tanh –1 (x).
cosh(x)	Hyperbolic cosine; cosh(x).
coth(x)	Hyperbolic cotangent; cosh(x)/sinh(x).
csch(x)	Hyperbolic cosecant; 1/sinh(x).
sech(x)	Hyperbolic secant; 1/cosh(x).
sinh(x)	Hyperbolic sine; sinh(x).
tanh(x)	Hyperbolic tangent; sinh(x)/cosh(x).
Complex Functions	
abs(x)	Absolute value; |x|.
angle(x)	Angle of a complex number x.
conj(x)	Complex conjugate of x.
imag(x)	Imaginary part of a complex number x.

real(x)	Real part of a complex number x.
Statistical Functions	
erf(x)	Computes the error function *erf* (x).
mean	Calculates the average.
median	Calculates the median.
std	Calculates the standard deviation.
Random Number Functions	
rand	Generates uniformly distributed random numbers between 0 and 1.
randn	Generates normally distributed random numbers.
Numeric Functions	
ceil	Rounds to the nearest integer toward •.
fix	Rounds to the nearest integer toward zero.
floor	Rounds to the nearest integer toward - •.
round	Rounds towards the nearest integer.
sign	Signum function.

Table.8: Numerical Methods:

Polynomial and Regression Functions	
conv	Computes product of two polynomials
deconv	Computes ratio of polynomials.
eig	Computes the eigenvalues of a matrix.
poly	Computes polynomial from roots.
polyfit	Fits a polynomial to data.
polyval	Evaluates polynomial and generates error estimates.
roots	Computes polynomial roots.
Interpolation Functions	
interp1	Linear and cubic-spline interpolations of a function of one variable.
interp2	Linear interpolation of a function of two variables.
spline	Cubic-spline interpolation.
unmkpp	Computes the coefficients of cubic-spinepolynomials.
Root Finding and Minimization	
fmin	Finds minimum of single-variable function.
fmins	Finds minimum of multivariable function.
fzero	Finds zero of single-variable function.
Numerical Integration Functions	
quad	Numerical integration with adaptive Simpson's rule.
quadl	Numerical integration with adaptive Lobatto quadrature.
trapz	Numerical integration with the trapezoidal rule.
Numerical Differentiation Functions	
diff(x)	Computes the difference between adjacent elements in vectorx.
polyder	Differentiates a polynomial, a polynomial product, or a polynomial quotient.
Symbolic Solution of Algebraic Equations	
det	Returns the determinant of a matrix.
eig	Returns the eigenvalues (characteristic roots) of a matrix.
inv	Returns the inverse of a matrix.
poly	Returns the characteristic polynomial of a matrix.

EXPT. NO. 1(a).VERIFICATION OF SUPERPOSITION THEOREM

Aim: To verify Superposition theorem.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

"In a linear network with several independent sources which include equivalent sources due to initial conditions, and linear dependent sources, the overall response in any part of the network is equal to the sum of individual responses due to each independent source, considered separately, with all other independent sources reduced to zero".

Procedure:

Step 1:

1. Make the connections as shown in the circuit diagram by using MULTISIM/MATLAB Simulink.

2. Measure the response 'I' in the load resistor by considering all the sources 10V, 15V and 8V in the network.

Step 2:

1. Replace the sources 15V and 8V with their internal impedances (short circuited).

2. Measure the response 'I_1' in the load resistor by considering 10V source in the network.

Step 3:

1. Replace the sources 10V and 8V with their internal impedances (short circuited).

2. Measure the response 'I_2' in the load resistor by considering 15V source in the network.

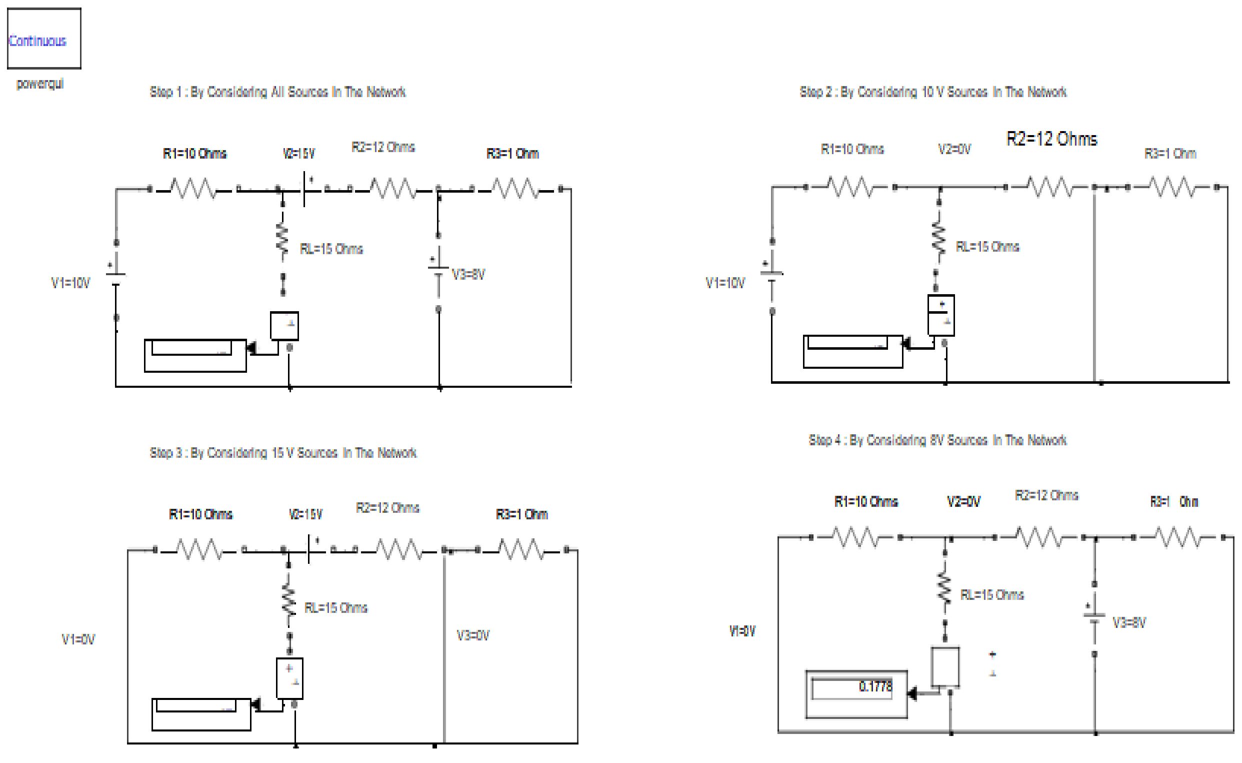
Step 4:

1. Replace the sources 10V and 15V with their internal impedances (short circuited).

2. Measure the response 'I_3' in the load resistor by considering 8V source in the network.

   The responses obtained in step 1 should be equal to the sum of the responses obtained in step 2, 3 and 4.

$I=I_1+I_2+I_3$, Hence Superposition Theorem is verified.



Current through Load Resistor 15 Ohms :

Considering 10V Source I1: 0.2667A

Considering 15VSource I2 : - 0.3333A

Considering 8VSource I3 : 0.1778A

Total Current : I1+I2+I3=0.2667-0.3333+0.1778
=0.1112A

With all the sources in the network I = 0.1111A

I=I1+I2+I3

Hence SuperPosition Theorem is Verified.

Fig.1.(a): Superposition Theorem

EXP. NO. 1(b).VERIFICATION OF THEVENIN'S THEOREM

Aim: To Verify Thevenin's Theorem.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

"Any two terminal networks consisting of linear impedances and generators may be replaced at the two terminals by a single voltage source acting in series with impedance. The voltage of the equivalent source is the open circuit voltage measured at the terminals of the network and the impedance, known as Thevenin's equivalent impedance, Z_{TH}, is the impedance measured at the terminals with all the independent sources in the network reduced to zero".

Procedure:

Step 1:

1. Make the connections as shown in the circuit diagram by using MULTISIM/MATLAB Simulink.
2. Measure the response 'I' in the load resistor by considering all the sources in the network.

Step 2: Finding Thevenin's Resistance(R_{TH})

1. Open the load terminals and replace all the sources with their internal impedances.
2. Measure the impedance across the open circuited terminal which is known as Thevenin's Resistance.

Step 3: Finding Thevenin's Voltage (V_{TH})

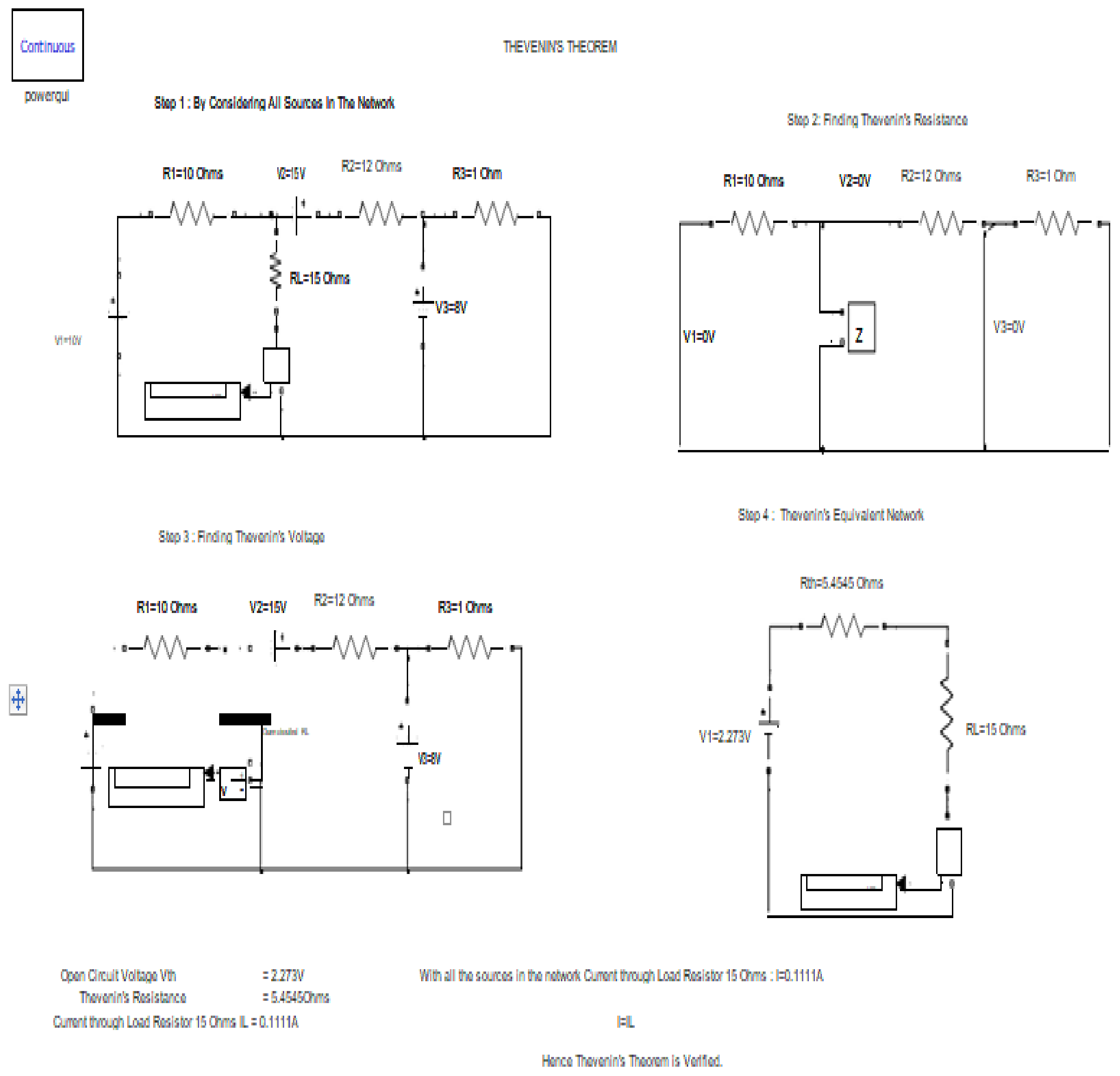
1. Open the load terminals and measure the voltage across the open circuited terminals.
2. Measured voltage will be known as Thevenin's Voltage.

Step 4: Thevenin's Equivalent Circuit

1. V_{TH} and R_{TH} are connected in series with the load.
2. Measure the current through the load resistor I= ______.

Current measured from Thevenin's Equivalent Circuit should be same as current obtained from the actual circuit.

$I = I_L$, Hence Thevenin's Theorem is verified.



Open Circuit Voltage Vth = 2.273V

Thevenin's Resistance = 5.4545Ohms

Current through Load Resistor 15 Ohms IL = 0.1111A

With all the sources in the network Current through Load Resistor 15 Ohms : I=0.1111A

I=IL

Hence Thevenin's Theorem is Verified.

Fig.1(b): Thevenin's Theorem

EXPT. NO. 1(c).VERIFICATION OF MAXIMUM POWER TRANSFER THEOREM

Aim: To Verify Maximum Power Transfer Theorem.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

"In any circuit the maximum power is transferred to the load when the load resistance is equal to the source resistance. The source resistance is equal to the Thevenin's equal resistance".

Procedure:

Step 1:

1. Make the connections as shown in the circuit diagram by using Multisim/MATLAB Simulink.

2. Measure the Power across the load resistor by considering all the sources in the network.

Step 2: Finding Thevenin's Resistance (R_{TH})

1. Open the load terminals and replace all the sources with their internal impedances.

2. Measure the impedance across the open circuited terminal which is known as Thevenin's Resistance.

Step 3: Finding Thevenin's Voltage (V_{TH})

1. Open the load terminals and measure the voltage across the open circuited terminals.

2. Measured voltage will be known as Thevenin's Voltage.

Step 4: Measuring Power for different Load Resistors

1. V_{TH} and R_{TH} are connected in series with the load.

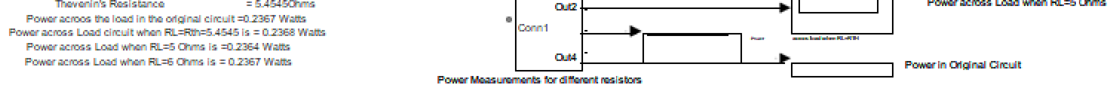
2. Measure power across the load by considering R_L=R_{TH}.

3. Measure power by using P = ___.

4. Verify the power for different values of load resistors (i.e. R_L>R_{TH} and R_L<R_{TH})

Power measured from the above steps results in maximum power dissipation when R_L=R_{TH}.

Hence Maximum Power Transfer Theorem is verified.


Thevenin's Resistance = 5.4545Ohms
Power across the load in the original circuit =0.2367 Watts
Power across Load circuit when RL=Rth=5.4545 is = 0.2368 Watts
Power across Load when RL=5 Ohms is =0.2364 Watts
Power across Load when RL=6 Ohms is = 0.2367 Watts
Out2
Conn1
Out4
Power across Load when RL=5 Ohms
Power in Original Circuit
Power Measurements for different resistors

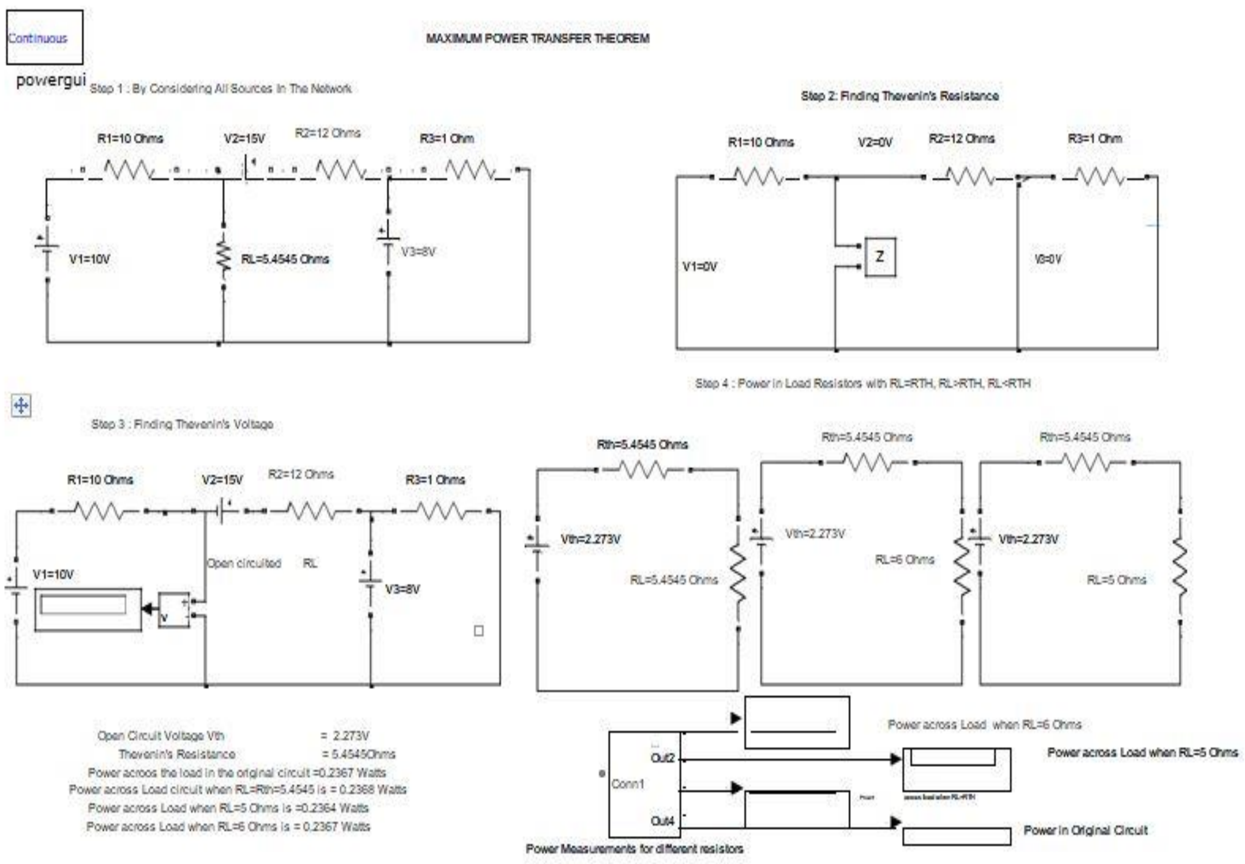

Continuous
powergui
MAXIMUM POWER TRANSFER THEOREM
Step 1 : By Considering All Sources In The Network
R1=10 Ohms
V2=15V
R2=12 Ohms
R3=1 Ohm
V1=10V
RL=5.4545 Ohms
V3=8V
Step 2: Finding Thevenin's Resistance
R1=10 Ohms
V2=0V
R2=12 Ohms
R3=1 Ohm
V1=0V
Z
V3=0V
Step 4 : Power in Load Resistors with RL=RTH, RL>RTH, RL<RTH
Step 3 : Finding Thevenin's Voltage
R1=10 Ohms
V2=15V
R2=12 Ohms
R3=1 Ohms
V1=10V
Open circuited RL
V3=8V
Rth=5.4545 Ohms
Vth=2.273V
RL=5.4545 Ohms
Rth=5.4545 Ohms
Vth=2.273V
RL=6 Ohms
Rth=5.4545 Ohms
Vth=2.273V
RL=5 Ohms
Open Circuit Voltage Vth = 2.273V
Thevenin's Resistance = 5.4545Ohms
Power across the load in the original circuit =0.2367 Watts
Power across Load circuit when RL=Rth=5.4545 is = 0.2368 Watts
Power across Load when RL=5 Ohms is =0.2364 Watts
Power across Load when RL=6 Ohms is = 0.2367 Watts
Power across Load when RL=6 Ohms
Out2
Conn1
Out4
Power across Load when RL=5 Ohms
Power in Original Circuit
Power Measurements for different resistors

Fig.1(c): Maximum Power Transfer Theorem

M-File Program for Maximum Power Transfer Theorem

```
clc;
close all;
clear all;
v=input('Enter the Voltage in Volts :');
rth=input('Enter the value of Thevenins Resistance:');
rl=1:0.0001:12;
i=v./(rth+rl);
p=i.^2.*rl;
plot(rl,p);
grid;
title('Maximum Power');
xlabel('Load Resistance in Ohms------->');
ylabel('Power Dissipation in watts-------->');
```

Thevenins Voltage V_{th}=

Thevenins Resistance r_{th}=

Current is i=

Power Dissipated P=

Result:

Viva Questions:

1. Statement of superposition and thevenin’s theorem?

2. What is the condition for Maxima theorem?

3. What is MATLAB?

4. What is the purpose of simulating in MATLAB software?

5. What are the advantages of MATLAB software?

EXPT. NOT. 2(a).SERIES RESONANCE

Aim: - To obtain the plot of of frequency vs. X_L, frequency vs. X_C, frequency vs. impedance andfrequency vs. current for the given series RLC circuit and determine the resonant frequency and check by theoretical calculations.
R = 15 Ω , C = 10 μF, L = 0.1 H, V = 50V vary frequency in steps of 1 Hz using Matlab.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Program:

```
%Program to find the Series Resonance
clc;
clear all;
close all;

r=input('enter the resistance value----->');
l=input('enter the inductance value------>');
c=input('enter the capacitance value----->');
v=input('enter the input voltage------->');
f=5:2:300;
xl=2*pi*f*l;
xc=(1./(2*pi*f*c));
x=xl-xc;
z=sqrt((r^2)+(x.^2));
i=v./z;
%plotting the graph
subplot(2,2,1);
plot(f,xl);
grid;
xlabel('frequency');
ylabel('X1');
subplot(2,2,2);
plot(f,xc);
grid;
xlabel('frequency');
ylabel('Xc');
subplot(2,2,3);
plot(f,z);
grid;
xlabel('frequency');

ylabel('Z');
```

```
subplot(2,2,4);
plot(f,i);
grid;

xlabel('frequency');
ylabel('I');
```

Result:

```
enter the resistance value-----

enter the inductance value------

enter the capacitance value-----

enter the input voltage-------
```

Viva Questions:

1. What is the Resonance?
2. What is the Series Resonance frequency?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT. NO. 2(b).PARALLEL RESONANCE

Aim: To obtain the graphs of frequency vs. B_L**, frequency**vs. B_C, frequency vs. admittance and frequency vs. current vary frequency in steps for the given circuit and find the resonant frequency and check by theoretical calculations.
R = 1000ohms , C = 400 *u* F, L = 1 H, V = 50V vary frequency in steps of 1 Hz using Matlab.

Apparatus:

S.No.	**NAME**	**No.**
1	PC	1
2	MATLAB Programing Software	1

Program:

```
%Program to find the Parallel Resonance
clc;
clear all;
close all;
r=input('enter the resistance value----->');
l=input('enter the inductance value------>');
c=input('enter the capacitance value----->');
v=input('enter the input voltage------->');
f=0:2:50;
xl=2*pi*f*l;
xc=(1./(2*pi*f*c));
b1=1./xl;
bc=1./xc;
b=b1-bc;

g=1/r;
y=sqrt((g^2)+(b.^2));
i=v*y;
%plotting the graph
subplot(2,2,1);
plot(f,b1);
grid;
xlabel('frequency');
ylabel('B1');
subplot(2,2,2);
plot(f,bc);
grid;
xlabel('frequency');
ylabel('Bc');
subplot(2,2,3);
plot(f,y);
grid;
xlabel('frequency');
```

```
ylabel('Y');
subplot(2,2,4);
plot(f,i);
rid;
xlabel('frequency');
ylabel('I');
```

Result:

Viva Questions:

1. What is the Resonance?
2. What is the Parallel Resonance frequency?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO:3(a).ROOT LOCUS

Aim: To obtain the root locus of the system whose transfer function is defined by

$$G(S) = \frac{(S+5)}{S^2+7S+25}$$

Apparatus:

S.No.	**NAME**	**No.**
1	PC	1
2	MATLAB Programing Software	1

Procedure:

1. Input the numerator and denominator co-efficient.
2. Formulate the transfer function using the numerator and denominators co-efficient with the help of function T = tf (num, den)
3. Plot the root locus of the above transfer function using rlocus(t).

Program:

```
%Program to find the root locus of transfer function%

   (s+5)
% -----------
% s^2+7s+25
clc;
clear all;
close all;
% initialzations
num=input('enter the numerator coefficients---->');
den=input('enter the denominator coefficients---->');
%Transfer function
sys=tf(num,den);
rlocus(sys);
```

PROGRAM OUTPUT:

```
enter the numerator coefficients---->
enter the denominator coefficients---->
```

Result:

EXPT.NO:3(b).BODE PLOT

Aim: To obtain the bode plot and to calculate the phase margin, gain margin, phase cross over and gaincross over frequency for the systems whose open loop transfer function is given as follows.

$$G(s) = \frac{25(S+1)\,(S+7)}{S(S+2)\,(S+4)\,(S+8)}$$

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

The gain margin is defined as the change in open loop gain required to make the systemunstable. Systems with greater gain margins can withstand greater changes in system parameters before becoming unstable in closed loop. Keep in mind that unity gain in magnitude is equal to a gain of zero in dB.

The phase margin is defined as the change in open loop phase shift required to make a closed loop system unstable.

The phase margin is the difference in phase between the phase curve and -180 deg at the point corresponding to the frequency that gives us a gain of 0dB (the gain cross over frequency, Wgc).

Likewise, the gain margin is the difference between the magnitude curve and 0dB at the point corresponding to the frequency that gives us a phase of -180 deg (the phase cross over frequency, Wpc).

Procedure:

1. Input the zeroes, poles and gain of the given system.
2. Formulate the transfer function from zeroes, poles and gain of the system.
3. Plot the bode plot using function bode (t).
4. Estimate PM,GM, W_{PC} , and W_{GC}. Using function margin.

Program:

```
Clc;
Clear all;
Close all;
% initialzations
k=input('enter the gain---->');
z=input('enter the zeros---->');
p=input('enter the ploes---->');
t=zpk(z,p,k);
bode(t);
[Gm,Pm,Wcg,Wcp]=margin(t);
disp(Gm);
disp(Pm);
disp(Wgc);
disp(Wpc);
```

PROGRAM Output:

```
enter the gain---->
enter the zeros---->
enter the ploes---->
```

RESULTS:

EXPT.NO:3(c).NYQUIST PLOT

Aim: To obtain the Nyquist plot and to calculate the phase margin, gain margin, phase cross over and gain cross over frequency for the systems whose open loop transfer function is given as follows.

$$G(S) = \frac{50(S+1)}{S(S+3)(S+5)}$$

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Procedure:

1. Input the zeroes, poles and gain of the given system.
2. Formulate the transfer function from zeroes, poles and gain of the system.
3. Plot the nyquist plot using function nyquist(t).
4. Estimate PM,GM, W_{PC} , and W_{GC}. Using function margin.

Program:

```
%Program to find the Nyquist Plot

%   50(s+1)
% --------
% s(s+3)(s+5)
clc;
clear all;
close all;
% initialzations
num=input('enter the numerator coefficients---->');
den=input('enter the denominator coefficients---->');
sys=tf(num,den);
nyquist(sys);
title('system1');
[Gm,Pm,Wcg,Wcp]=margin(sys);
disp(Gm);
disp(Pm);
disp(Wgc);
disp(Wpc);
```

Results:

```
enter the numerator coefficients---->
enter the denominator coefficients---->
```

Viva Questions:

1. What is the Rootlocus?
2. What is the Phase margin?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO.4.TRANSFER FUNCTION ANALYSIS

A) Time Response for Step Input
B) Frequency Response for SinusoidalInput

A) Time Response for Step Input

Aim: To find the A) Time response for step input B) Frequency response for sinusoidal input.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

The general expression of transfer function of a second order control system is given as

$$\frac{C(s)}{R(s)} = \frac{Wn^2}{Wn^2+2\xi Wns+Wn^2}$$

Here, ζ and $\omega_n$are damping ratio and natural frequency of the system respectively
There are number of common terms in transient response characteristics and which are

1. Delay time (td) is the time required to reach at 50% of its final value by a time response signalduring its first cycle of oscillation. $T_d = \frac{1+0.7\xi}{Wn^2}$

2. Rise time (tr) is the time required to reach at final value by a under damped time response signalduring its first cycle of oscillation. If the signal is over damped, then rise time is counted as thetime required by the response to rise from 10% to 90% of its final value.

$$\beta=\tan^{-1}\left[\frac{\sqrt{1-\xi^2}}{\xi}\right] \quad ; \quad T_r=\frac{1}{Wd}\tan^{-1}\left[-\frac{\sqrt{1-\xi^2}}{\xi}\right]= \frac{\pi-\beta}{Wd}$$

3.Peak time (tp) is simply the time required by response to reach its first peak i.e. the peak of firstcycle of oscillation, or first overshoot.

$$T_p=\frac{\pi}{Wd} = \frac{\pi}{Wn\sqrt{1-\xi^2}}$$

4. Maximum overshoot (Mp) is straight way difference between the magnitude of time response and magnitude of its steady state. Maximum overshoot is expressed in term of percentage of steady-state value of the response. As the first peak of response is normally maximum in magnitude, maximum and steady-state value of a response.

$$M_p=e^{-\pi\xi/\sqrt{1-\xi^2}}$$

$$M_p\%=e^{-\pi\xi/\sqrt{1-\xi^2}}*100\%$$

5. Settling time (ts) is the time required for a response to become steady. It is defined as the time required by the response to reach and steady with value.

$T_s = \frac{4}{\xi W n}$ (2% Criterion)

6. Steady-state error (e_{ss}) is the difference between actual output and desired output at the infinite range of time. $e_{ss} = \text{Lim}_{t\text{-}\alpha} [r(t)-c(t)]$

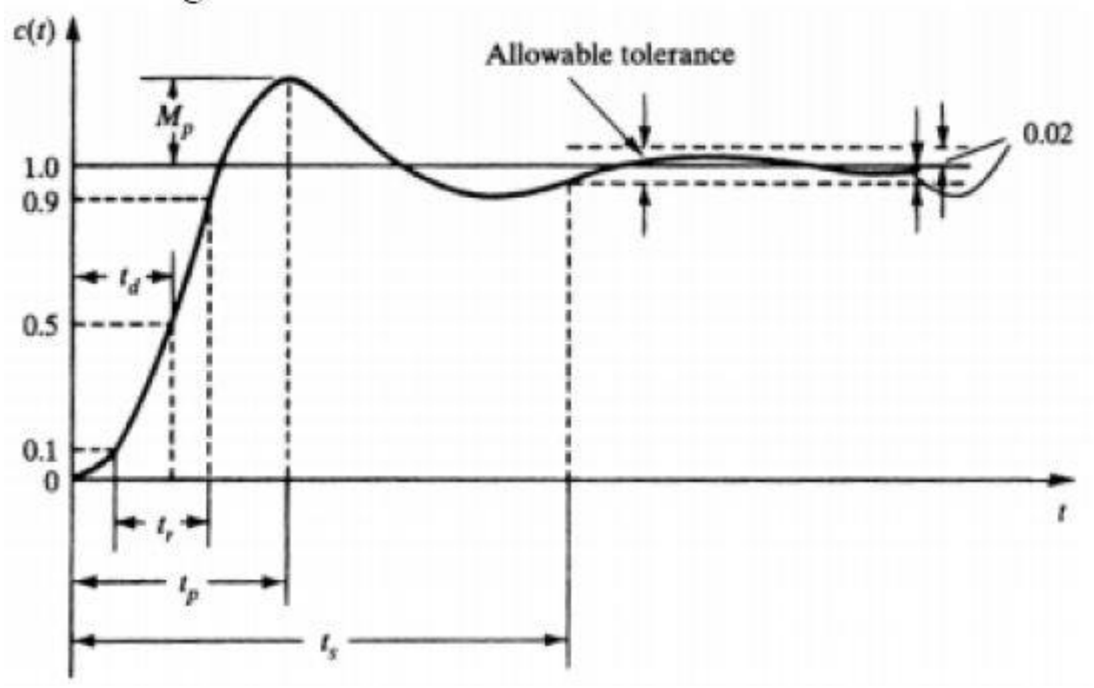


Fig.1:Characterstics of Time Response

<u>Problem Statement:</u> For the closed loop system defined by

$$\frac{C(S)}{R(S)} = \frac{100}{S^2+12S+100}$$

Plot the unit step response curve and find time domain specifications

<u>PROGRAM: Time Response for Step Input</u>

```
clc;
clear all;
close all;
num=input('enter the numerator coefficients----> );
den=input('enter the denominator coefficients----> );system=tf(num,den);
system
step(system)
grid on;
wn=sqrt(den(1,3));
zeta= den(1,2)/(2*wn);
wd=wn*sqrt(1-zeta^2);
disp('Delay time in seconds is')
td=(1+0.7*zeta)/wd
disp('Rise time in seconds is')
theta=atan(sqrt(1-zeta^2)/zeta);
```

```
tr=(pi-theta)/wd
disp('Peak time in seconds');
tp=pi/wd
disp('Peak overshoot is');
mp=exp(-zeta*pi/sqrt(1-zeta^2))*100
disp('settling time in seconds is');
ts=4/(zeta*wn)
```

Program Output:

```
enter the numerator coefficients---->
enter the denominator coefficients---->
```

Result:

B) Frequency Response for Sinusoidal Input

By the term frequency response, we mean the steady-state response of a system to a sinusoidal input. Industrial control systems are often designed using frequency response methods. Many techniques are available in the frequency response methods for the analysis and design of control systems. Whenever it is not possible to obtain the transfer function of a system through analytical techniques, frequency response test can be used to compute its transfer function. The design and adjustment of open-loop transfer function of a system for specified closed-loop performance is carried out more easily in frequency domain. Further, the effects of noise and parameter variations are relatively easy to visualize and assess through frequency response. The Nyquist criteria is used to extract information about the stability and the relative stability of a system in frequency domain.

$$T(s) = \frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2}$$

Substitute, $s = j\omega$ in the above equation.

$$T(j\omega) = \frac{\omega_n^2}{(j\omega)^2 + 2\delta\omega_n(j\omega) + \omega_n^2}$$

$$\Rightarrow T(j\omega) = \frac{\omega_n^2}{-\omega^2 + 2j\delta\omega\omega_n + \omega_n^2} = \frac{\omega_n^2}{\omega_n^2\left(1 - \frac{\omega^2}{\omega_n^2} + \frac{2j\delta\omega}{\omega_n}\right)}$$

$$\Rightarrow T(j\omega) = \frac{1}{\left(1 - \frac{\omega^2}{\omega_n^2}\right) + j\left(\frac{2\delta\omega}{\omega_n}\right)}$$

Let, $\frac{\omega}{\omega_n} = u$ Substitute this value in the above equation.

$$T(j\omega) = \frac{1}{(1 - u^2) + j(2\delta u)}$$

Magnitude of $T(j\omega)$ is -

$$M = |T(j\omega)| = \frac{1}{\sqrt{(1 - u^2)^2 + (2\delta u)^2}}$$

Program: Frequency Response for Sinusoidal Input

```
%Frequency Response of second order system
clc;
clear all;
close all;
num=input('enter the numerator coefficients---->');
den=input('enter the denominator coefficients---->');
%Transfer function
sys=tf(num,den);
wn=sqrt(den(1,3));
zeta= den(1,2)/(2*wn);
w=linspace(0,2);
u=w/wn;
len=length(u);
for k=1:len
m(k)=1/(sqrt((1-u(k)^2)+(2*zeta*u(k))^2));
phi(k)=-atan((2*zeta*u(k))/(1-u(k)^2))*180/pi;
end
subplot(1,2,1)
plot(w,m)
xlabel('normalized frequency')
ylabel('magnitude')
subplot(1,2,2)plot(w,phi)
xlabel('normalized frequency')
ylabel('phase')
disp('resonant peak is');
mr=1/(2*zeta*sqrt(1-zeta^2))
disp('resonant frequency in rad/sec is');
wr=wn*sqrt(1-2*zeta^2)
disp('bandwidth in rad/sec is');
wb=wn*sqrt(1-2*zeta^2+sqrt(2-4*zeta^2+4*zeta^4))
disp('phase margin in degrees is')
pm=180+(atan(2*zeta/sqrt(-2*zeta^2+sqrt(4*zeta^4 +1))))*180/pi
```

Program Output:

```
enter the numerator coefficients---->
enter the denominator coefficients---->
```

Result:

Viva Questions:

1. Explain the Delay time?

2. Draw the circuit of 2nd order system?

3. What is MATLAB?

4. What is the purpose of simulating in MATLAB software?

5. What are the advantages of MATLAB software?

EXPT.NO.5. LOAD FLOW STUDIES

A) Y bus Formation

Aim: To develop a mat lab program for Y bus FORMATION

Apparatus:

S.No.	**NAME**	**No.**
1	PC	1
2	MATLAB Programing Software	1

Procedure:

1. Switch on the computer with mat lab software

2. Double click the mat lab icon

3. In the command window go to file and open new M-file or editor file

4. Type the program code without errors

5. Save the file with .m extinction

6. Now debug the file for errors

7. If they are any errors and warnings rectify them and save the file

8. Again save and debug so that errors are avoided

9. To observe the output go to command window after debugging

10. Type Quit in the command window to exit from the mat lab.

Flowchart:

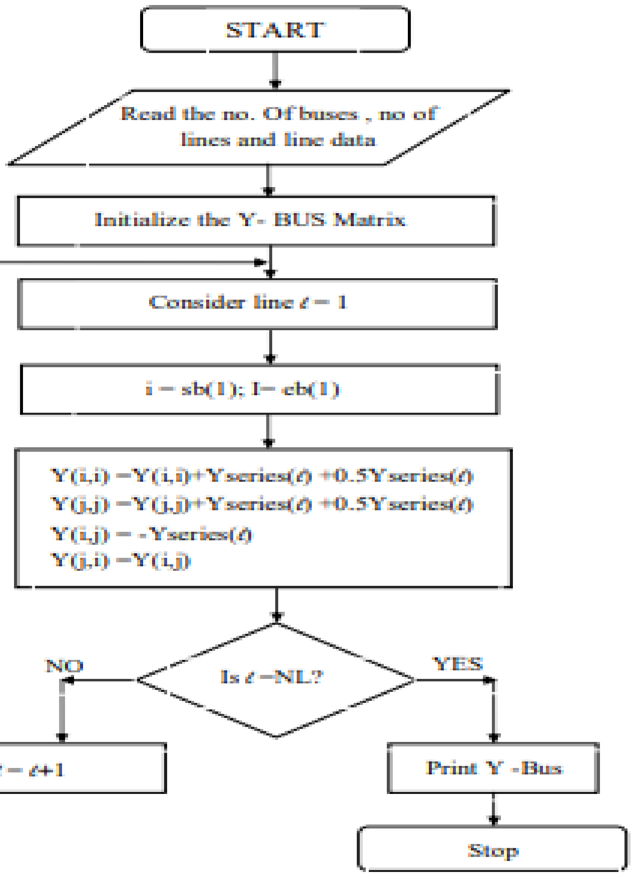

START
Read the no. Of buses , no of lines and line data
Initialize the Y- BUS Matrix
Consider line ℓ = 1
i = sb(1); I= eb(1)
Y(i,i) =Y(i,i)+Yseries(ℓ) +0.5Yseries(ℓ)
Y(j,j) =Y(j,j)+Yseries(ℓ) +0.5Yseries(ℓ)
Y(i,j) = -Yseries(ℓ)
Y(j,i) =Y(i,j)
Is ℓ =NL?
NO
YES
ℓ = ℓ+1
Print Y -Bus
Stop

Fig.1:Flow chart of Ybus System.

Program:

```
clc;
linedata = [ 1     2      0.02    0.06     0.03    1;
             1     3      0.08    0.24     0.025  1;
             2     3      0.06    0.18     0.02   1;
             2     4      0.06    0.18     0.02   1;
             2     5      0.04    0.12     0.015  1;
             3     4      0.01    0.03     0.01   1;
             4     5      0.08    0.24     0.24   1;];

lp = linedata(:,1);          % From bus number...
lq = linedata(:,2);          % To bus number...
r = linedata(:,3);           % Resistance, R...
x = linedata(:,4);           % Reactance, X...
ycp = linedata(:,5);         % Ground Admittance, B/2...
a = linedata(:,6);           % Tap setting value..
z = r + i*x;   % Z matrix...
y = 1./z;                    % To get inverse of each element...
ycp = i*ycp;   % Make B imaginary...

nbus = max(max(lp),max(lq));   % no. of buses...
```

```
nline = length(lp);   % no. of branches...
Y = zeros(nbus,nbus);         % InitialiseYBus...

% Formation of the Off Diagonal Elements...

for k=1:nline
    Y(lp(k),lq(k)) = Y(lp(k),lq(k))-y(k)/a(k);
    Y(lq(k),lp(k)) = Y(lp(k),lq(k));
end

% Formation of Diagonal Elements....

for m =1:nbus
for n =1:nline
iflp(n) == m
          Y(m,m) = Y(m,m) + y(n)/(a(n)^2) + ycp(n);
elseiflq(n) == m
          Y(m,m) = Y(m,m) + y(n) + ycp(n);
end
end
end
 Y                 % Bus Admittance Matrix..
```

PRECAUTIONS:

1. Avoid spelling errors while typing
2. Save the file with extension of .m
3. Type the program in the editor window only.

Result:

B) PROGRAMMING OF POWER FLOW USING NEWTON-RAPHSON METHOD.

Aim: To simulate power flow using Newton-Raphson Method.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Procedure:

1. Switch on the computer with mat lab software

2. Double click the mat lab icon

3. In the command window go to file and open new M-file or editor file

4. Type the program code without errors

5. Save the file with .m extinction

6. Now debug the file for errors

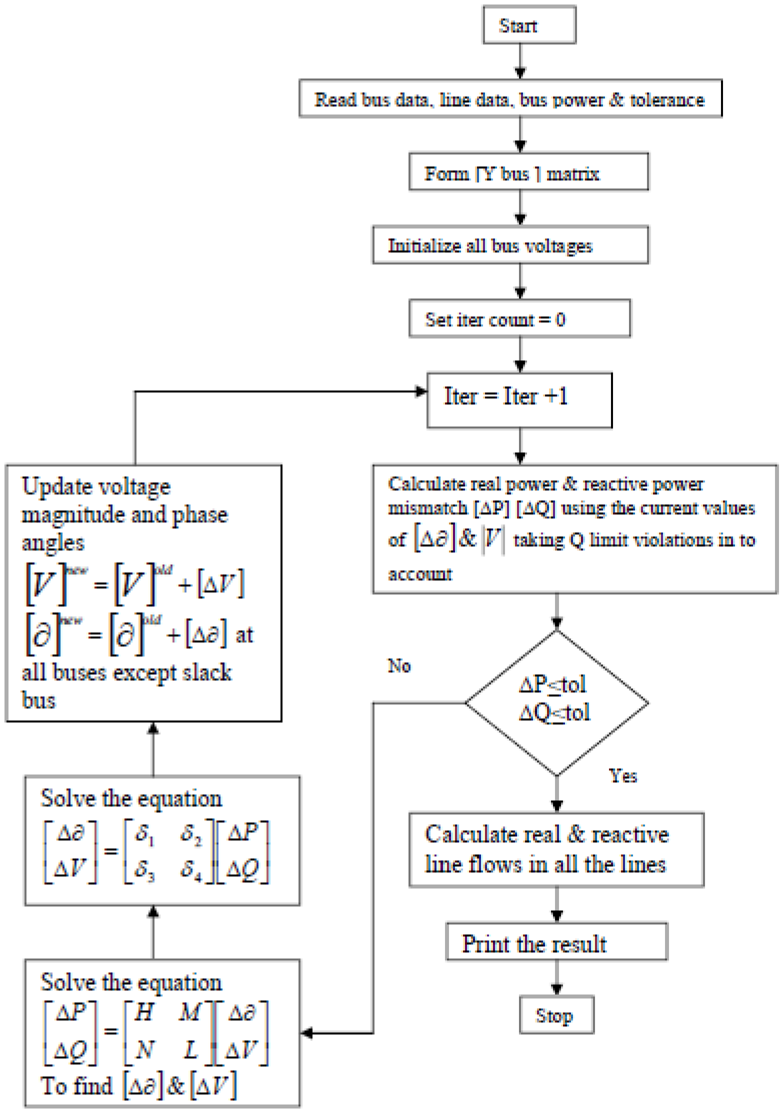
7. If they are any errors and warnings rectify them and save the file

8. Again save and debug so that errors are avoided

9. To observe the output go to command window after debugging

10. Type Quit in the command window to exit from the mat lab.

Flow Chart:



Start
Read bus data, line data, bus power & tolerance
Form [Y bus] matrix
Initialize all bus voltages
Set iter count = 0
Iter = Iter +1
Calculate real power & reactive power mismatch [ΔP] [ΔQ] using the current values of [Δ∂] & |V| taking Q limit violations in to account
ΔP≤tol
ΔQ≤tol
No
Yes
Calculate real & reactive line flows in all the lines
Print the result
Stop
Solve the equation
[ΔP; ΔQ] = [H M; N L][Δ∂; ΔV]
To find [Δ∂] & [ΔV]
Solve the equation
[Δ∂; ΔV] = [δ1 δ2; δ3 δ4][ΔP; ΔQ]
Update voltage magnitude and phase angles
[V]new = [V]old + [ΔV]
[∂]new = [∂]old + [Δ∂] at all buses except slack bus

Fig.2:Flow chart of N-R Methods.

Program:

```
clc;

nbus = 3;

Y = [ 0-15i  0+10i  0+5i

      0+10i  0-15i  0+5i

      0+5i   0+5i   0-10i];


busdata = [1    1     1    0     0       0          0       0;

           2    2     1.1  0     5.3217  0          0       0;

           3    3     1    0    -3.6392  -0.5339    0       0;];

bus = busdata(:,1);        % Bus Number..
```

```
type = busdata(:,2);        % Type of Bus 1-Slack, 2-PV, 3-PQ..
V = busdata(:,3);           % Specified Voltage..
del = busdata(:,4);         % Voltage Angle..
P = busdata(:,5);           % Real power
Q = busdata(:,6);           % Reactive power
Qmin = busdata(:,7);        % Minimum Reactive Power Limit..
Qmax = busdata(:,8);        % Maximum Reactive Power Limit..
Psp = P;   % P Specified..
Qsp = Q;   % Q Specified..
G = real(Y);   % Conductance matrix..
B = imag(Y);   % Susceptance matrix..
pv = find(type == 2 | type == 1);   % PV Buses..
pq = find(type == 3);               % PQ Buses..
npv = length(pv);   % No. of PV buses..
npq = length(pq);   % No. of PQ buses..
Tol = 1;
Iter =1;
while (Tol > 0.000001)   % Iteration starting..
    P = zeros(nbus,1);
    Q = zeros(nbus,1);
% Calculate P and Q
fori = 1:nbus
for k = 1:nbus
            P(i) = P(i) + V(i)* V(k)*(G(i,k)*cos(del(i)-del(k)) + B(i,k)*sin(del(i)-del(k)));
            Q(i) = Q(i) + V(i)* V(k)*(G(i,k)*sin(del(i)-del(k)) - B(i,k)*cos(del(i)-del(k)));
end
end
% Checking Q-limit violations..
ifIter<= 7 &&Iter> 2    % Only checked up to 7th iterations..
for n = 2:nbus
if type(n) == 2
if Q <Qmin(n)
               Q=Qmin;
elseif Q >Qmax(n)
```

```
                Q=Qmax;
end
end
end
end

% Calculate change from specified value
dPa = Psp-P;
dQa = Qsp-Q;
    k = 1;
dQ = zeros(npq,1);
fori = 1:nbus
if type(i) == 3
dQ(k,1) = dQa(i);
            k = k+1;
end
end
dP = dPa(2:nbus);
    M = [dP; dQ];   % Mismatch Vector
% Jacobian terms
% J1 - Derivative of Real Power Injections with Angles..
    J1 = zeros(nbus-1,nbus-1);
fori = 1:(nbus-1)
        m = i+1;
for k = 1:(nbus-1)
            n = k+1;
if n == m
for n = 1:nbus
J1(i,k) = J1(i,k) + V(m)* V(n)*(-G(m,n)*sin(del(m)-del(n)) + B(m,n)*cos(del(m)-del(n)));
end
                J1(i,k) = J1(i,k) - V(m)^2*B(m,m);
else
J1(i,k) = V(m)* V(n)*(G(m,n)*sin(del(m)-del(n)) - B(m,n)*cos(del(m)-del(n)));
end
```

```
end

end


% J2 - Derivative of Real Power Injections with V..

    J2 = zeros(nbus-1,npq);

fori = 1:(nbus-1)

        m = i+1;

for k = 1:npq

            n = pq(k);

if n == m

for n = 1:nbus

     J2(i,k) = J2(i,k) + V(n)*(G(m,n)*cos(del(m)-del(n)) + B(m,n)*sin(del(m)-del(n)));

end

                J2(i,k) = J2(i,k) + V(m)*G(m,m);

else

J2(i,k) = V(m)*(G(m,n)*cos(del(m)-del(n)) + B(m,n)*sin(del(m)-del(n)));

end

end

end


% J3 - Derivative of Reactive Power Injections with Angles..

    J3 = zeros(npq,nbus-1);

fori = 1:npq

        m = pq(i);

for k = 1:(nbus-1)

            n = k+1;

if n == m

for n = 1:nbus

                    J3(i,k) = J3(i,k) + V(m)* V(n)*(G(m,n)*cos(del(m)-del(n)) + B(m,n)*sin(del(m)-del(n)));

end

                J3(i,k) = J3(i,k) - V(m)^2*G(m,m);

else

                J3(i,k) = V(m)* V(n)*(-G(m,n)*cos(del(m)-del(n)) - B(m,n)*sin(del(m)-del(n)));

end
```

```
end
end

% J4 - Derivative of Reactive Power Injections with V..
    J4 = zeros(npq,npq);
fori = 1:npq
        m = pq(i);
for k = 1:npq
            n = pq(k);
if n == m
for n = 1:nbus
                    J4(i,k) = J4(i,k) + V(n)*(G(m,n)*sin(del(m)-del(n)) - B(m,n)*cos(del(m)-del(n)));
end
                J4(i,k) = J4(i,k) - V(m)*B(m,m);
else
J4(i,k) = V(m)*(G(m,n)*sin(del(m)-del(n)) - B(m,n)*cos(del(m)-del(n)));
end
end
end

    J = [J1 J2; J3 J4];   % Jacobian Matrix..
X = inv(J)*M;   % Correction Vector
dTh = X(1:nbus-1);      % Change in Voltage Angle..
dV = X(nbus:end);       % Change in Voltage Magnitude..

% Updating State Vectors..

    del(2:nbus) = dTh + del(2:nbus);    % Voltage Angle..
    k = 1;
fori = 2:nbus
if type(i) == 3
            V(i) = dV(k) + V(i);   % Voltage Magnitude..
            k = k+1;
end
```

```
end


    Tol = max(abs(M));    % Tolerance..

Iter

V

del

J

Iter = Iter + 1;

End
```

Result:

Viva Questions:

1. What is Bus?
2. What is the comparison of GS, NR and FDCL?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO.6. FAULT ANALYSIS

Aim: To Analyze Fault analysis of power system.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

Short circuits and other abnormal conditions often occur on a power system. Short circuits areusually called "faults" by power system engineers. Some defects, other than short circuits arealso termed as faults.

Faults are caused either by insulation failures or by conducting path failures. The failureof insulation results in short circuits which are very harmful as they may damage someequipment of the power system. Most of the faults in transmission and distribution lines arecaused by over voltages due to lightning or switching surges, or by external conducting objectsfalling on overhead lines. Overvoltages due to lightning or switching surges cause flashover onthe surface of insulators resulting in short circuits. Short circuits are also caused by tree branchesor other conducting objects falling on the overhead lines.

The fault impedance being low, the fault currents are relatively high. The fault currentsbeing excessive, they damage the faulty equipment and the supply installation. Also, the systemvoltage may reduce to a low level, windings and busbars may suffer mechanical damage due tohigh magnetic forces during faults and the individual generators in a power station or group ofgenerators in different power stations may loose synchronism

The symmetrical fault occurs when all the three conductors of a three-phase line are brought together simultaneously into a short–circuit condition as shown in Figure 1.

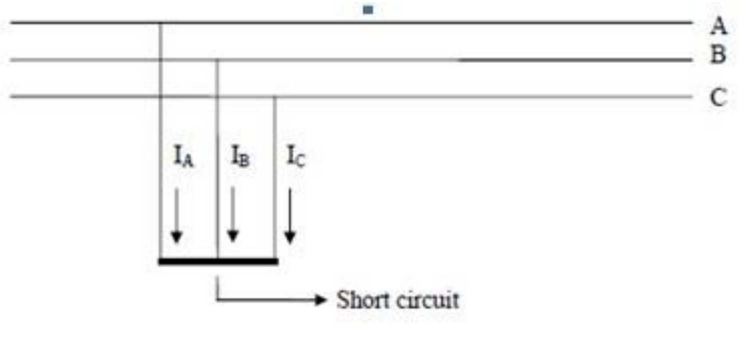


Fig.1: 3-Phase fault analysis.

This type of fault gives rise to symmetrical currents i.e. equal fault currents with 120^0 displacement. Thus referring to Figure 5.1, fault currents I_A, I_B and I_C will be equal in magnitude with 120^0 displacement among them. Because of balanced nature of fault, only one phase needs to be considered in calculations since condition in the other two phases will also be similar. A three-phase short circuit occurs rarely but it is most severe type of fault involving largest currents. For this reason, the balanced short-circuit calculations are performed to determine these large currents to be used to determine the rating of the circuit breakers.

Procedure:

1. Open Matlab-->Simulink--> File ---> New---> Model

2. Open Simulink Library and browse the components

3. Connect the components as per circuit diagram

4. Set the desired voltage and required frequency

5. Simulate the circuit using MATLAB

6. Plot the waveforms.

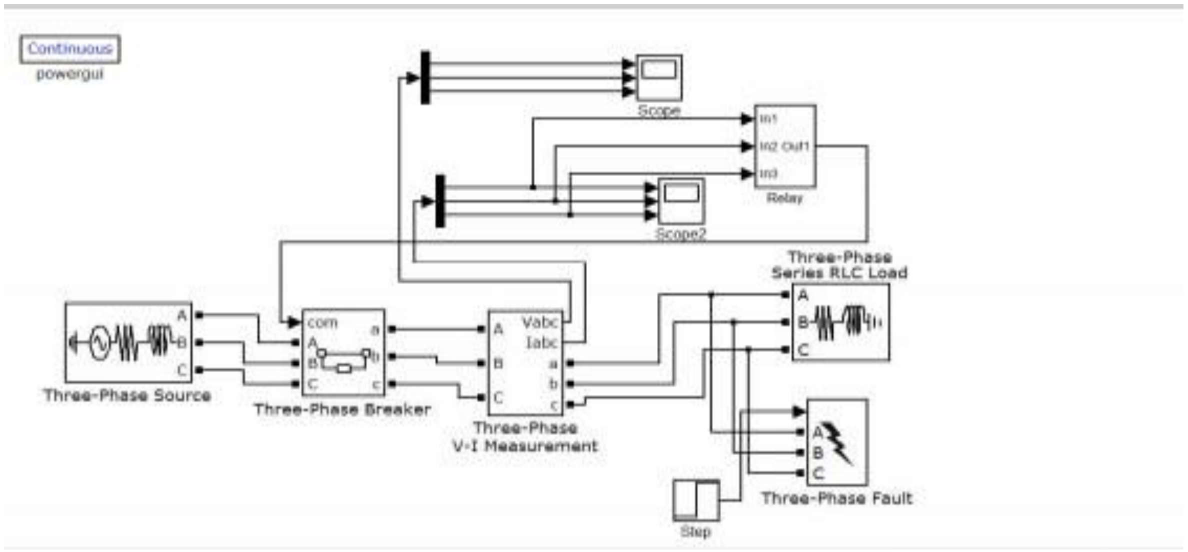


Fig.2:The distribution Model.

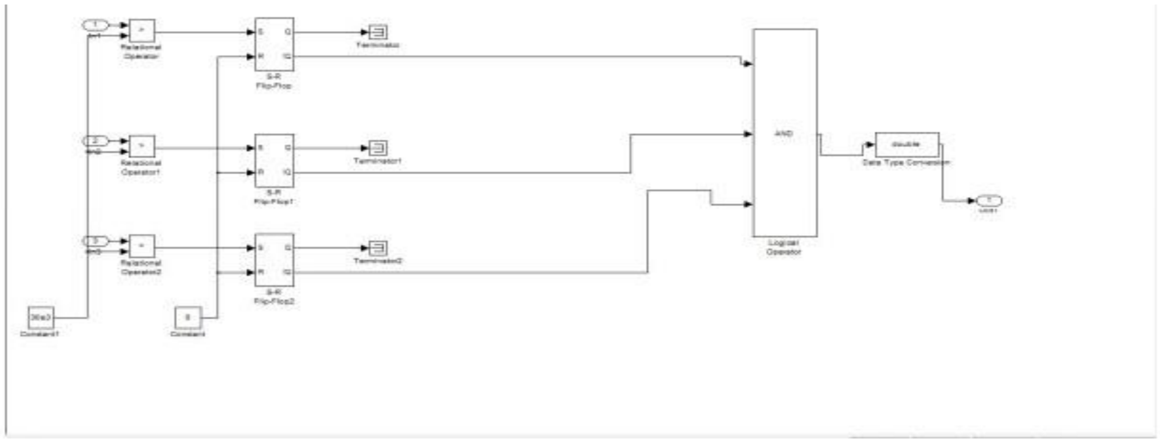


Fig.3:Relay Subsystem.

RESULTS:

Viva Questions:

1. What is the type of faults?
2. What is the different between LLF and LLL?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO.7. ECONOMIC POWER SCHEDULING

Aim: To understand the fundamentals of economic dispatch and solve the problem using classical method with and without line losses.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

Mathematical Model for Economic Dispatch of Thermal Units

Without Transmission Loss:

Statement of Economic Dispatch Problem

In a power system, with negligible transmission loss and with N number of spinning thermal generating units the total system load PD at a particular interval can be met by different sets of generation schedules

$\{PG_1$^(k) , PG_2^(k) , ………………PG_N^(K) }; k = 1,2,……..N_S

Out of these NS set of generation schedules, the system operator has to choose the set of schedules, which minimize the system operating cost, which is essentially the sum of the production cost of all the generating units. This economic dispatch problem is mathematically stated as an optimization problem.

Given: The number of available generating units N, their production cost functions, their operating limits and the system load PD,

To determine: The set of generation schedules,

PG_i; i=1,2,………………N ------(1)

Which minimize the total production cost

Min:F_T= $\Sigma F_i(PG_i)$ ------(2)

Power balance constraint

$\Sigma PG_i - P_D = 0$ ------(3)

And the operating limits

$PG_{i,min} \leq PG_i \leq PG_{i,max}$ ------(4)

The units production cost functions is usually approximated by quadratic function

$Fi(PG_i)=a_i PG^2{}_i+b_i PG_i+c_i$; i=1,2,------N --------(5)

Where a_i,b_i and c_i are constants.

Necessary conditions for the existence of solution to ED problem

The ED problem given by the equations (1) to (4). By omitting the inequality constraints (4) tentatively, the reduce ED problem (1),(2) and (3) may be restated as an unconstrained

Flowchart:

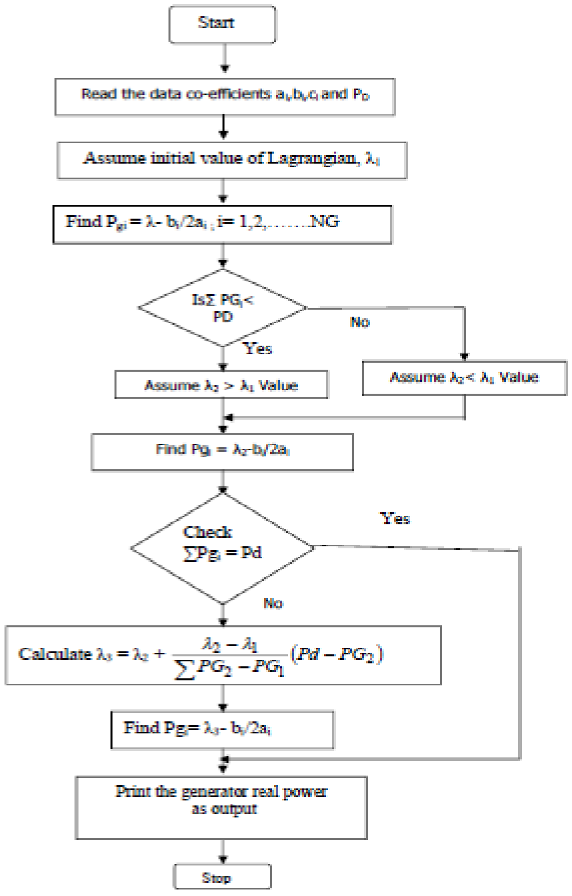

Start
Read the data co-efficients a_i,b_i,c_i and P_D
Assume initial value of Lagrangian, λ_1
Find P_gi = λ- b_i/2a_i ; i= 1,2,…….NG
IsΣ PG_i< PD
No
Yes
Assume λ_2 > λ_1 Value
Assume λ_2< λ_1 Value
Find Pg_i = λ_2-b_i/2a_i
Check ΣPg_i = Pd
Yes
No
Calculate λ_3 = λ_2 + (λ_2 − λ_1)/(ΣPG_2 − PG_1) (Pd − PG_2)
Find Pg_i= λ_3- b_i/2a_i
Print the generator real power as output
Stop

Fig.1:Flowchart of Economic Dispatch.

Procedure:

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program.
4. Execute the program by either pressing Tools – Run.
5. View the results.

Exercise-1:

The fuel cost functions for three thermal plants in \$/h are given by

$C1 = 500 + 5.3\ P1 + 0.004\ P_1^2$; P_1 in MW

$C2 = 400 + 5.5\ P2 + 0.006\ P_2^2$; P_2 in MW

$C3 = 200 +5.8\ P3 + 0.009\ P_3^2$; P_3 in MW

The total load , PD = 800MW.Neglecting line losses and generator limits, find the optimal dispatch and the total cost in \$/h by analytical method. Verify the result using MATLAB program.

Program:

```
alpha = [500; 400; 200];
beta = [5.3; 5.5; 5.8];
gamma = [0.004; 0.006; 0.009];
PD = 800;
DelP = 10;
lamda = input('Enter estimated value of Lamda = ');
fprintf(' ')
disp(['Lamda P1 P2 P3 DP'..............' grad Delamda'])
iter = 0;
while abs(DelP) >= 0.001
iter = iter + 1;
P = (lamda - beta)./(2*gamma);
DelP = PD - sum(P);
J = sum(ones(length(gamma),1)./(2*gamma));
Delamda = DelP/J;
disp([lamda,P(1),P(2),P(3),DelP,J,Delamda])
lamda = lamda + Delamda;
end
totalcost = sum(alpha + beta.*P + gamma.*P.^2)
```

Program OUTPUT:

```
>> Economicloadscheduling
Enter estimated value of Lamda =
```

Result:

Viva Questions:

1. What is the Economic seduling?
2. Explain the $Fi(PG_i)=a_i\ PG^2_i+b_i\ PG_i+c_i$?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO.8. DESIGN OF FILTERS (LOW PASS FILTER)

Aim: To understand the fundamentals of Low pass Filters.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

A filter is a device that changes the amplitude (height) of an AC voltage (a voltage in the form of a sine wave) as the frequency of the input voltage changes. Filters have two terminals. The input terminals take in the input voltage, which passes through the filter and onto the output terminals, where the resulting output waveform can be observed. Figure 1.1 is a basic representation of a filter.

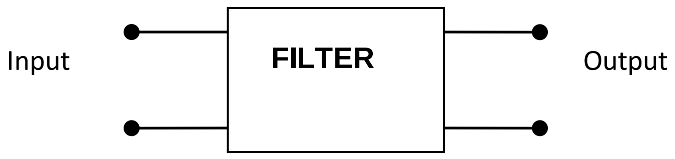


Fig.1: Circuit for the Filter.

There are several types of filters, but in this experiment, we will be looking at three types.

low-pass filter is a filter that allows a signal of a low frequency (i.e. a low amount of oscillations per second) to pass through it. Consequently, it attenuates (reduces) the amplitude of an input signal whose frequency is higher than the *cutoff* frequency.

high-pass filter is a filter that passes high frequencies well, but attenuates (or reduces) frequencies lower than the cutoff frequency.

band-pass filter is a device that passes frequencies within a certain range and rejects (attenuates) frequencies outside that range.

These three filters will be investigated in this experiment.

Low-Pass Filter

Figure (a) shows a simple low-pass filter consisting of a resistor and a capacitor, which should be constructed on your breadboard. Notice that the input is connected in series with the resistor, and the output is the voltage across the capacitor. The input and output have one common terminal, which is the low (ground, or reference) side of each.

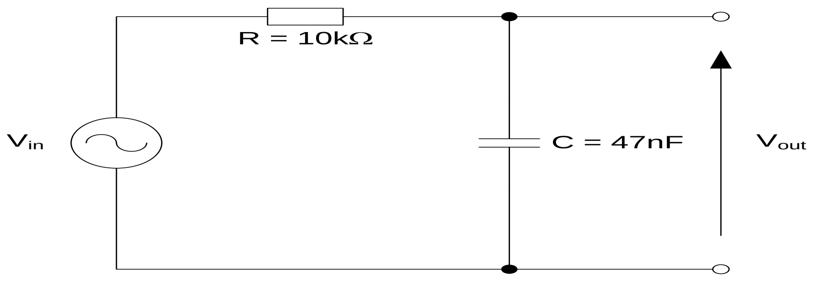


Fig.2: Low-pass filter.

$V_{in}(t)-V_{out}(t)=Ri(t)$

$Q_c(t)=CV_{out}(t)$

$i(t)=dQ_c/dt$

$V_{in}(t)-V_{out}(t)=RC(dV_{out}/dt)$

Procedure:

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program.
4. Execute the program by either pressing Tools – Run.
5. View the results.

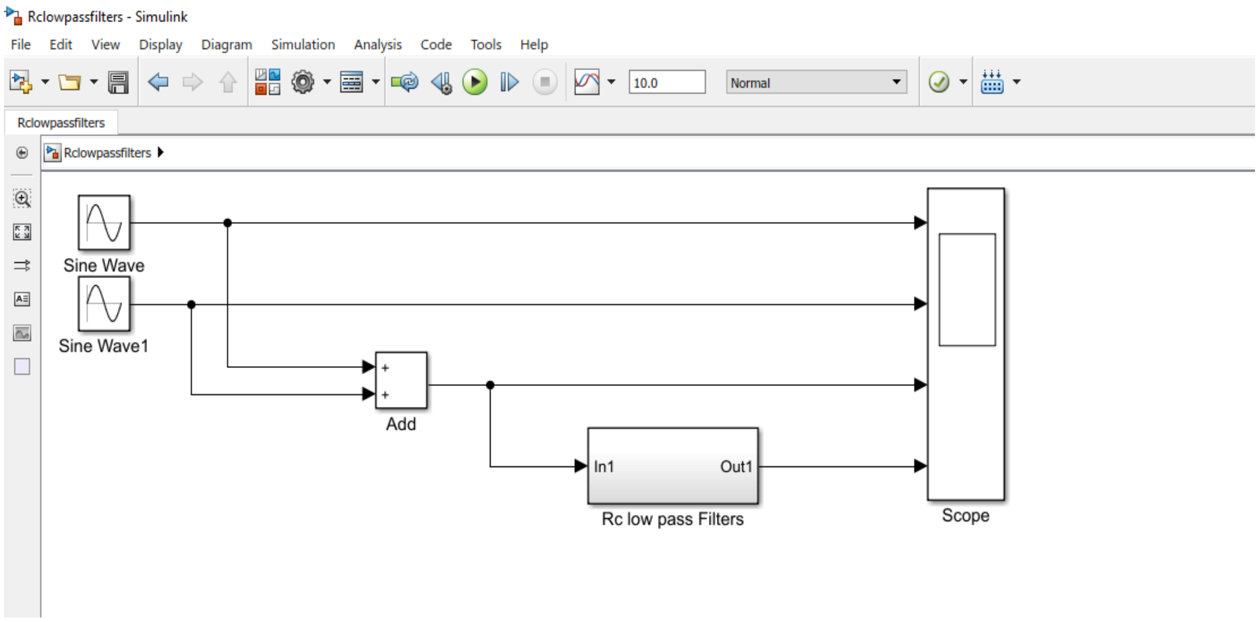

Rclowpassfilters - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
Normal
Rclowpassfilters
Sine Wave
Sine Wave1
Add
In1
Out1
Rc low pass Filters
Scope

Fig.3.:Simulink model of Low Pass Filter.

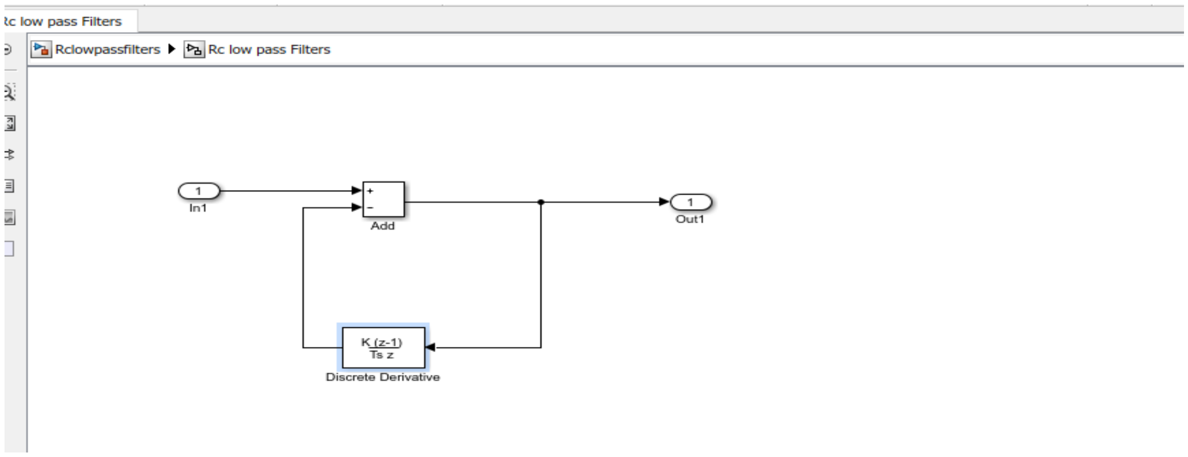


Fig.4: Desing of Sub System (low pass filters)

Result:

Viva Questions:

1. What is the Filter?

2. Explain the Low pass filter and High pass filters?

3. What is MATLAB?

4. What is the purpose of simulating in MATLAB software?

5. What are the advantages of MATLAB software?

EXPT.NO.9. CHOPPER FED DC MOTOR DRIVES

Aim: Chopper fed DC drives; the variable voltage to the armature of a dc motor for speed control can be obtained from a dc chopper.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

A dc motor fed by dc source through a chopper which consist of gto thyristor and freewheeling diode D.The motor drive mechanical load characteristics by inertia "J".Friction coefficient "B" and load torque TL.The motor uses the descrete dc machine provided in the machine library.The hysteresis current controller compares the sensed current with reference and generate the trigger signal for the gtothyrister to force the motor current to follow the reference. The speed controller loop uses a propotionalintegeral controller loop which provides the reference for the the current loop.

Procedure

- Open the matlab window
- Select "simulink"icon from the window
- Simulink library browser will appear
- Select "new model"from the file menu
- Select all the functional blocks required from the various libraries and copy them to the new model
- Connect the blocks according to the block diagram
- Select the parameters of various blocks according to requirements and initialize the model property
- Select the discrete state in the configuration parameters
- Simulate the completed block diagram and analysis the performance using the wave forms obtained using a "scope"
- Save the file using .mdl extension

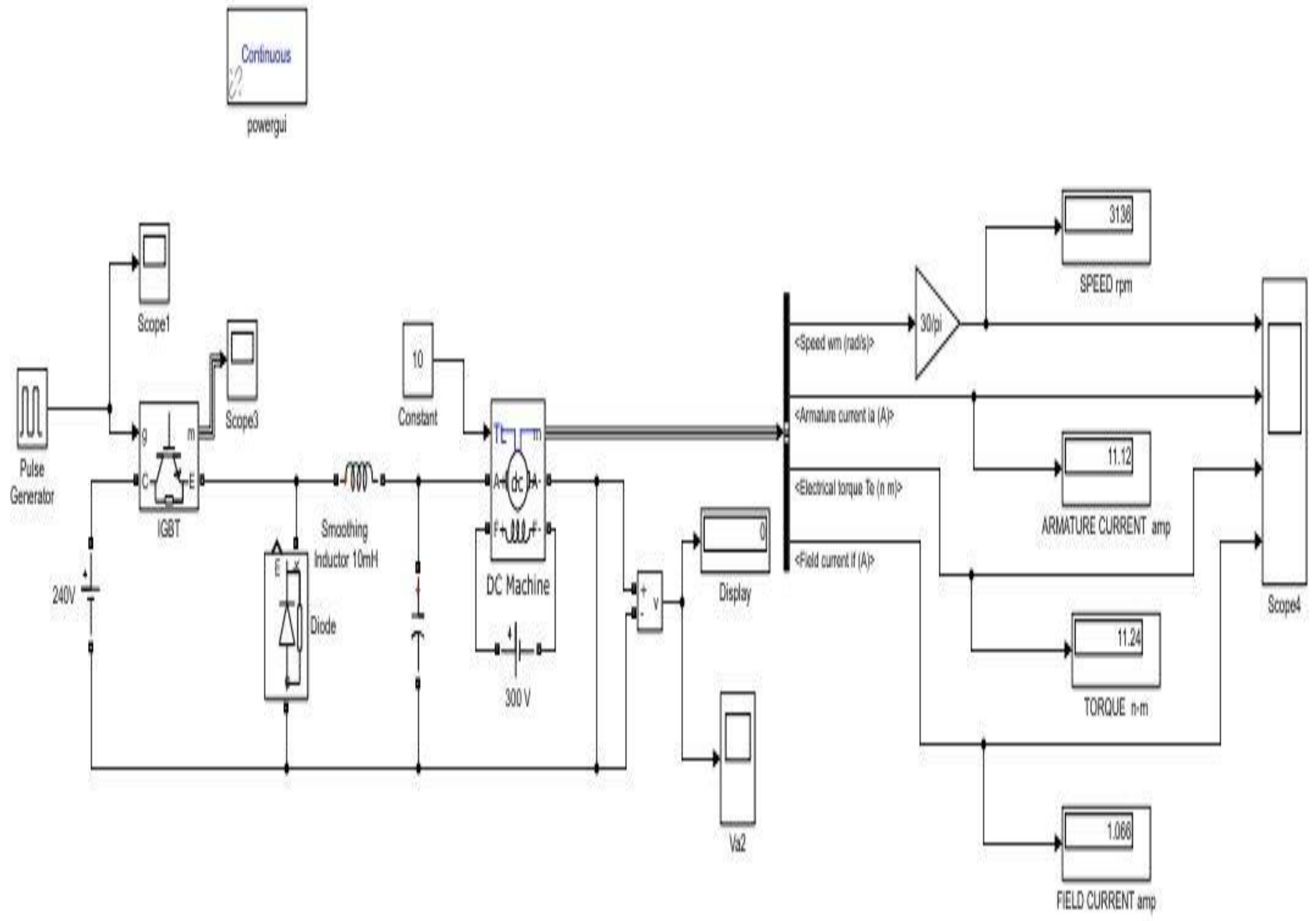


Fig.1: Matlab Simulink Model To Open Loop Control Dc Motor Drives.

Open Loop Output:

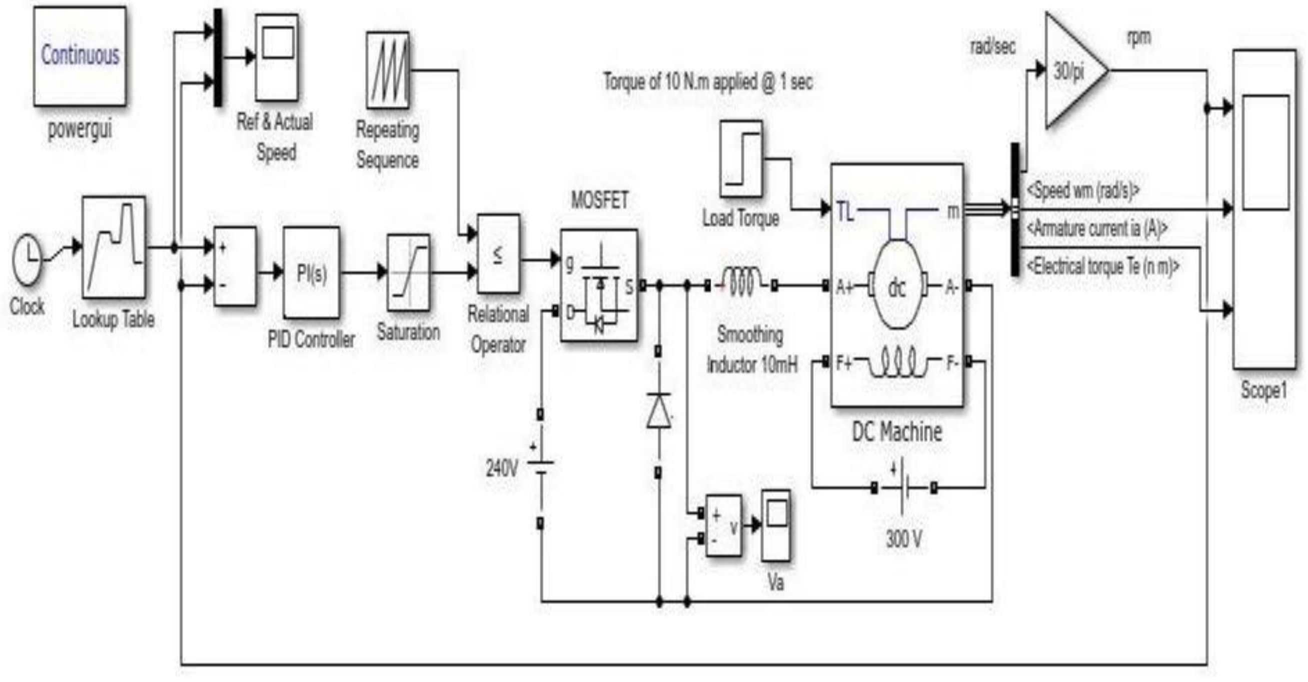


Fig.2: Matlab Simulink Model To Closed Loop Control Dc Motor Drives.

Rating of the elements used in above simulation:

DC input voltage – 240 V

DC machine rating – 5HP, 240 V, 1750rpm

Applied field voltage – 300 V

Torque of 10 N-m is applied @ 1 sec ,

L – 10mH

After simulation of the above model we are getting a graph of armature speed, armature current, electrical torque and armature voltage with respect to time.

The speed of a dc motor has been successfully controlled by using Chopper as a converter and Proportional-Integral type Speed and Current controller based on the closed loop model of DC motor. Initially a simplified closed loop model for speed control of DC motor is considered and requirement of current controller is studied. Then a generalized modelling of dc motor is done.Afterthat a complete layout of DC drive system is obtained. Then designing of current and speed controller is done. Now the simulation is done in MATLAB under varying load condition, varying reference speed condition and varying input voltage. The results are also studied and analyzed under above mentioned conditions. The model shows good results under all conditions employed during simulation. Since, the simulation of speed control of DC motor has been done. We can also implement it in hardware to observe actual feasibility. Here speed control of DC motor is done for rated and below rated speed. We can also control the speed of DC motor above rated speed and this can be done by field flux control.

Closedloop Output:

At simulation time stop =1

At simulation time stop =10

Result:

Viva Questions:

1. What is a Chopper?

2. What is the difference between chopper and inverter?

3. What is MATLAB?

4. What is the purpose of simulating in MATLAB software?

5. What are the advantages of MATLAB software?

6. Classify choppers.

EXPT.NO.10. VOLTAGE SOURCE INVERTER CONTROLLED INDUCTION MOTOR DRIVE

Aim: To control an induction motor drive using VSI

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

Voltage source inverter gives schematic diagram of a VSI fed induction motor drive using transistor is shown in below. self-commutated device can be used instead of transistor. Generally MOSFET is used in low voltage low power inverters, IGBT and power transistors are used up to medium power levels and GTO and IGCT are used for high power levels.

VSI can be operated as a stepped wave inverter or a pulse width modulated (PWM) inverter. When operated as a stepped wave inverter, transistor are a time difference of T/6 and each transistor is kept of one cycle. Resultant line voltage waveform is shown in fig. Frequency of the inverter operation is varied by varying T and the output voltage of inverter is varied by varying dc input Transistor Inver The various VSI controlled IM drive is shown below. i. When supply is dc, variable dc input voltage

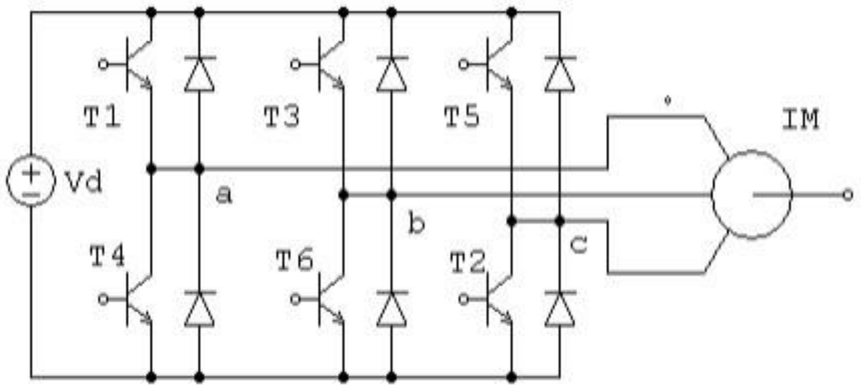


Fig.1: Transistor Inverter-fed Induction Motor Drive.

Procedure:

Open the matlab window
- Select ”simulink”icon from the window
- Simulink library browser will appear
- Select “new model”from the file menu
- Select all the functional blocks required from the various libraries and copy them to the new model
- Connect the blocks according to the block diagram
- Select the parameters of various blocks according to requirements and initialize the model property

- Select the discrete state in the configuration parameters
- Simulate the completed block diagram and analysis the performance using the wave forms obtained using a “scope”
- Save the file using .mdl extension

• Click the run button and analyse the outputs.

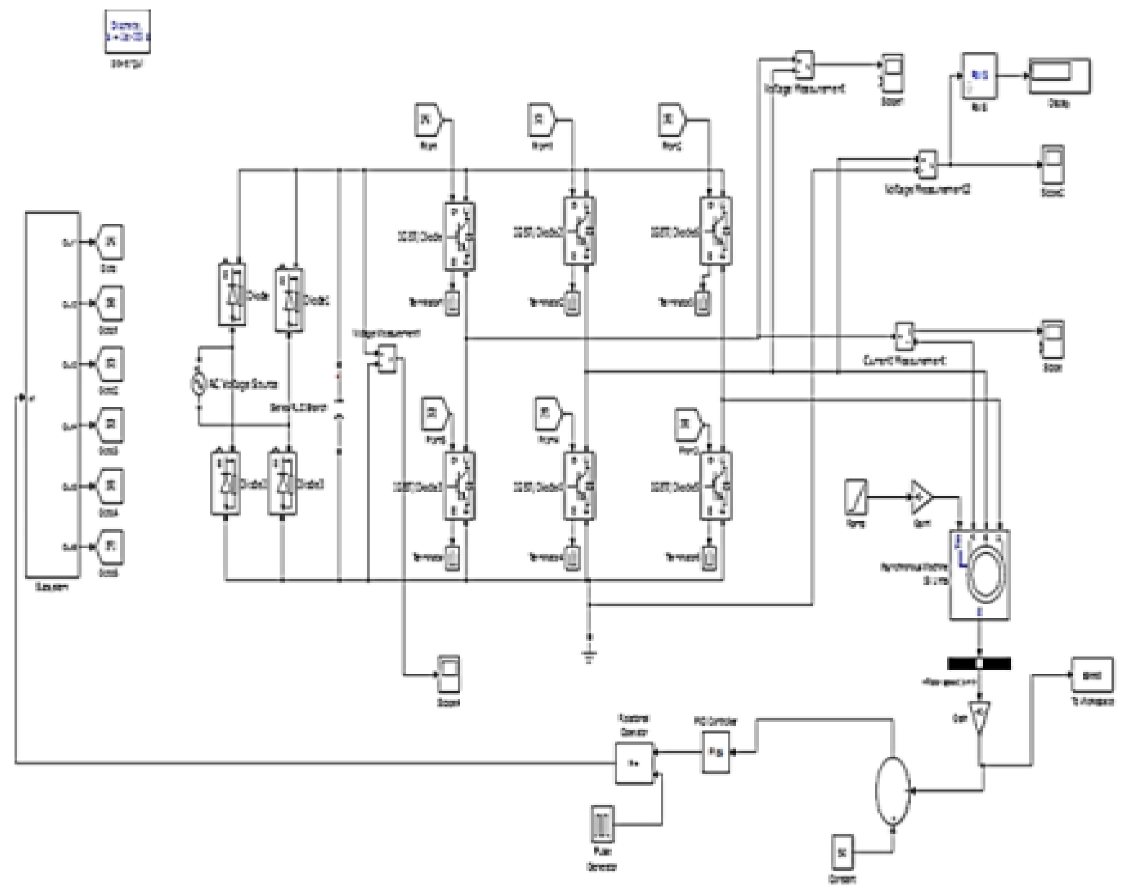


Fig.2:Simulated diagram of the VSI controlled Induction Motor Drive.

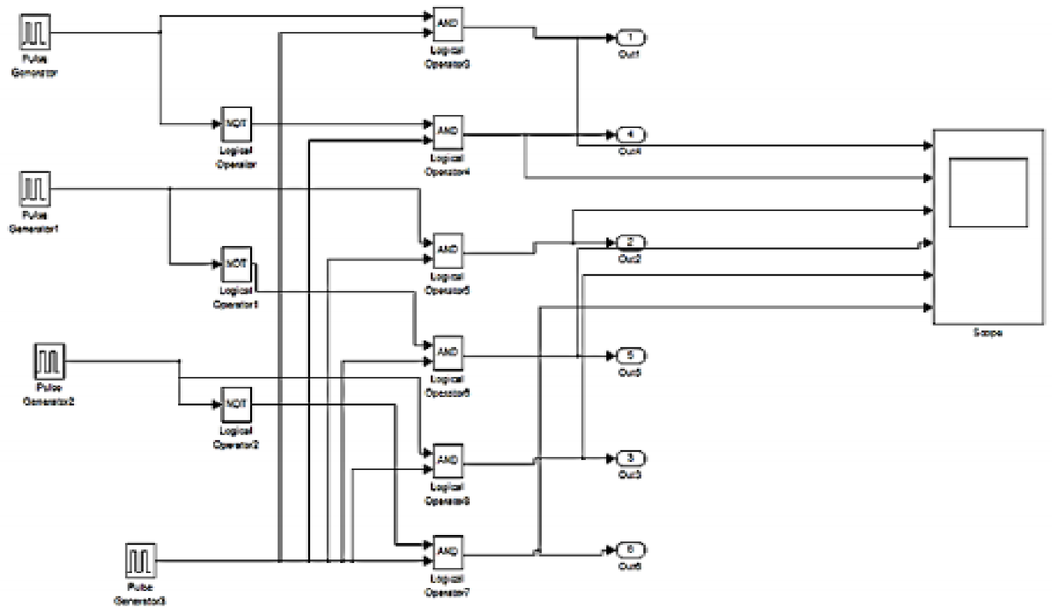


Fig.3: Gating Signal control for VSI..

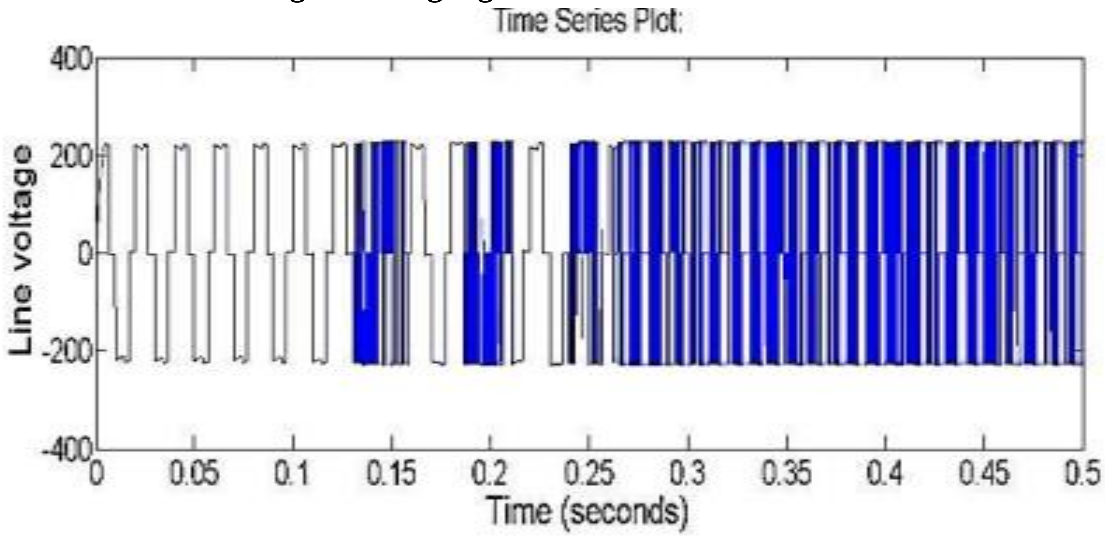


Fig.4:Line voltage.

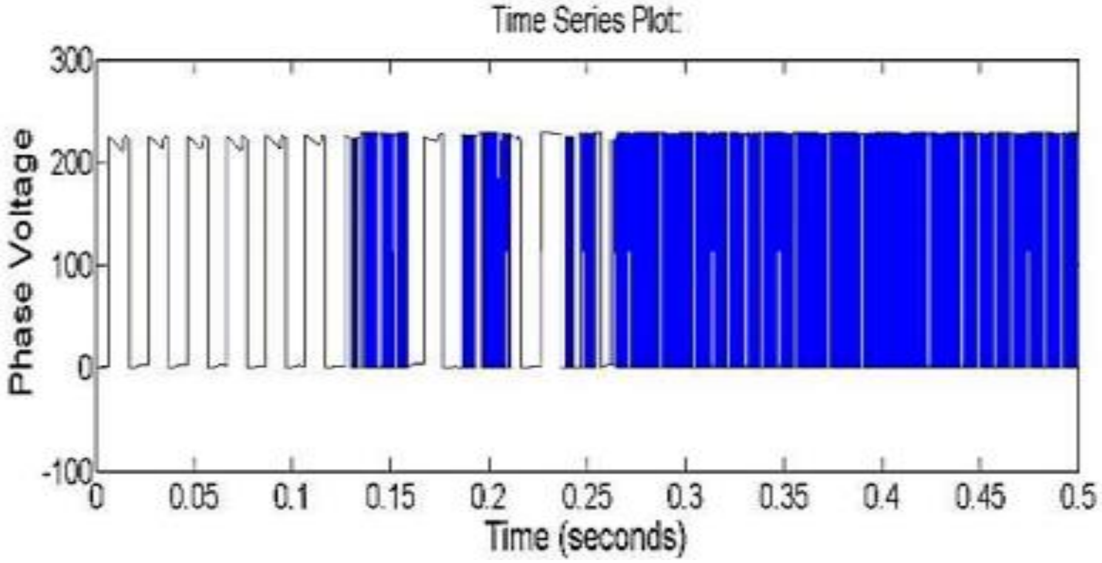


Fig.5:Phase Voltage.

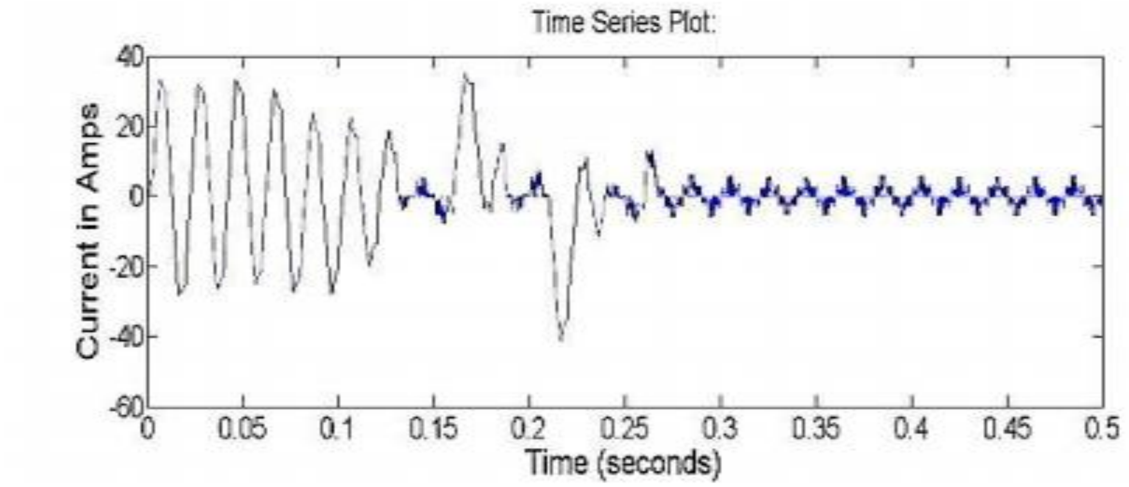


Fig.6:Current in R phase.

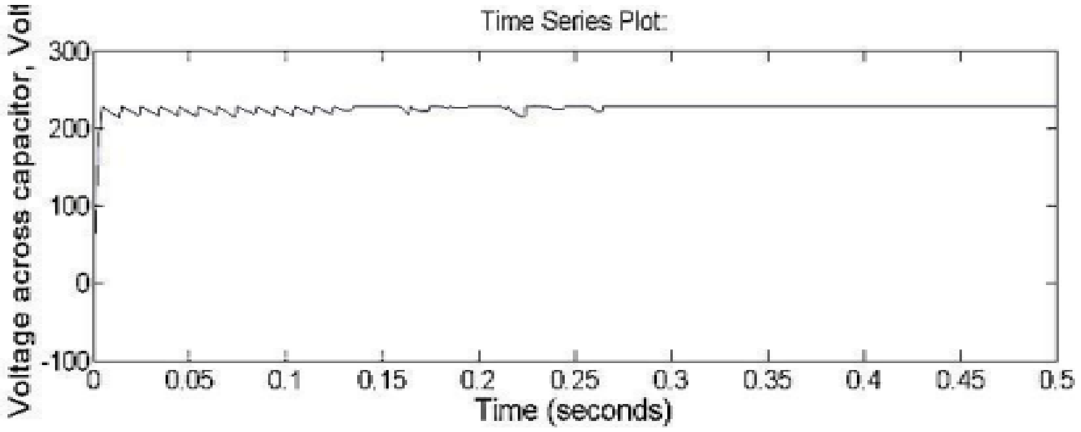
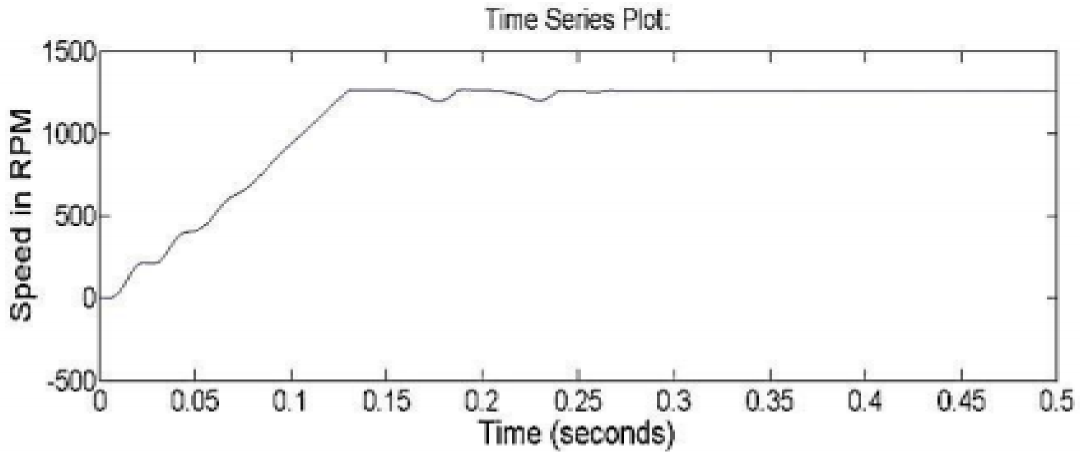


Fig.7:Voltage across Capacitor.



.

Fig.8:Speed in RPM.

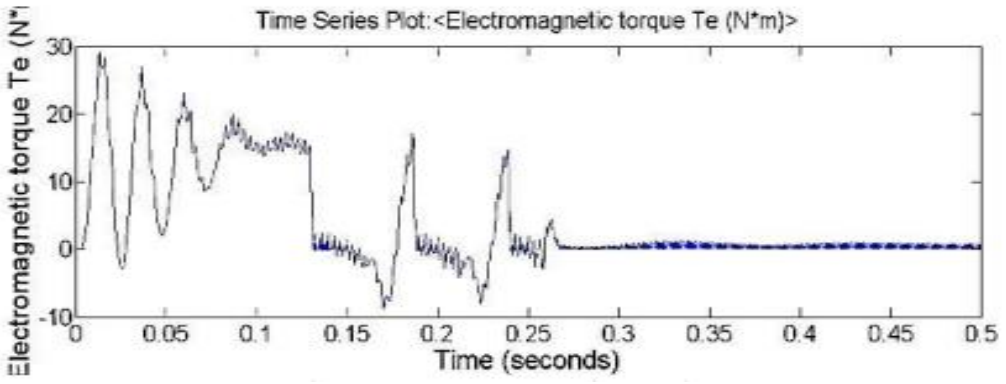


Fig.9:Torque .

<u>Result:</u>

<u>Viva Questions:</u>

1. What is a Voltage source Inverter?

2. What is the difference between chopper and Voltage source Inverter?

3. What is MATLAB?

4. What is the purpose of simulating in MATLAB software?

5. What are the advantages of MATLAB software?

EXPT.NO.11. AUTOMATIC GENERATION CONTROL

Aim: To determine the change in speed, frequency and steady state error corresponding to a load disturbance in a single area and a two area power system, with and without supplementary control using software.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

SIMULINK is an interactive environment for modelling, analyzing and simulating a wide variety of dynamic systems. SIMULINK provides a graphical user interface for constructing block diagram models using drag and drop operations. A system is configured in terms of block diagram representation using library of standard components. A system in block diagram representation is built easily and simulation results are displayed quickly.

Single Area System:

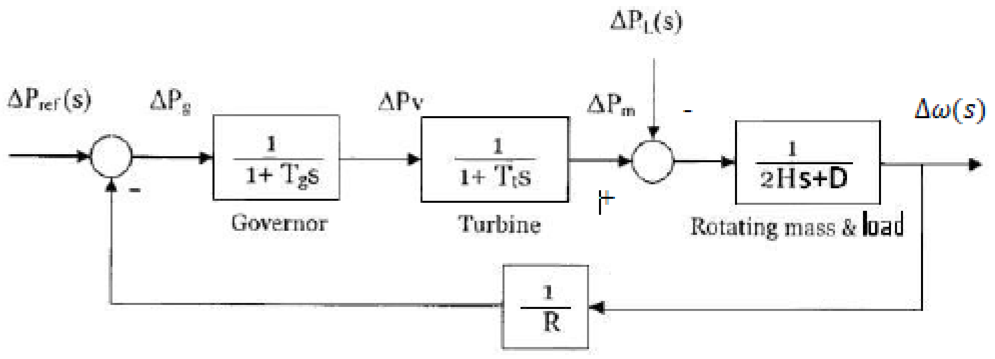


Fig.1: Load frequency control block diagram of an isolated power system

Problem 1:

An isolated power system has the following parameters:

Turbine time constant, Tt = 0.5 Sec

Governor time constant, Tg = 0.2 Sec

Generator time constant, H = 5 Sec

Governor Speed regulator, R = R pu.

The load varies by 0.8% for 1% change in frequency, i.e., D=0.8. The governor speed regulation is set to R= 0.05 pu. The turbine rated output is 250 MW. At normal frequency of 50 Hz a sudden load change of 50MW (ΔPL= 0.2pu) occurs. Construct a SIMULINK block 'diagram-and obtain the frequency deviation response for the condition given above.

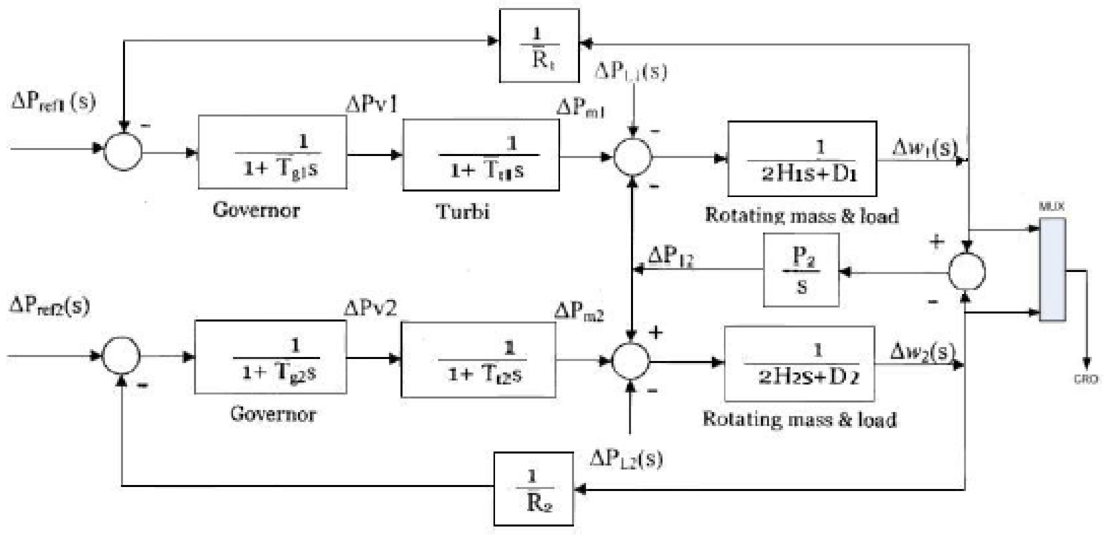


Fig .2: Two area system with only primary LFC Loop

RESULTS:

Viva Questions:

1. What is a primary LFC Loop?
2. What is the difference between single area and two area?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?

EXPT.NO.12. Z-BUS BUILDING ALGORITHM USING MATLAB SOFTWARE

Aim: To obtain the Zbus matrix for the given power system using Zbus building algorithm and to verify the same using MATLAB.

Apparatus:

S.No.	NAME	No.
1	PC	1
2	MATLAB Programing Software	1

Theory:

The Ybus /Zbus matrix constitutes the models of the passive portions of the power network. The impedance matrix is a full matrix and is most useful for short circuit studies. An algorithm for formulating [Zbus] is described in terms of modifying an existing bus impedance matrix designated as [Zbus]old. The modified matrix is designated as [Zbus]new. The network consists of a reference bus and a number of other buses. When a new element having self impedance Zb is added, a new bus may be created (if the new element is a tree branch) or a new bus may not be created (if the new element is a link). Each of these two cases can be subdivided into two cases so that Zb may be added in the following ways:

1. Adding Zb from a new bus to reference bus.
2. Adding Zb from a new bus to an existing bus.
3. Adding Zb from an existing bus to reference bus.
4. Adding Zb between two existing buses.

Procedure:

1. Enter the command window of the MATLAB.
2. Create a new M – file by selecting File - New – M – File
3. Type and save the program in the editor Window
4. Execute the program by either pressing Tools – Run.
5. View the results.

Exercise:

(i) Determine the and Z bus matrix for the power system network shown in fig.
(ii) Check the results obtained in using MATLAB.

PROBLEM ON FORMATION OF Zbus: Find the bus impedance matrix using Zbus building algorithm for the given power system whose reactance values are as follows.

Table.1

Sending end	Receiving end	Reactance values in ohms
0	1	J1.0
0	2	J0.8
1	2	J0.4
1	3	J0.2
2	3	J0.2
3	4	J0.008

%Program For Formation Of Zbus Using The Given Data:

```
z = [0 1 0 1.0
0 2 0 0.8
1 2 0 0.4
1 3 0 0.2
2 3 0 0.2
3 4 0 0.08];
Y = ybus(z)
Ibus = [-j*1.1; -j*1.25; 0; 0];
Zbus = inv(Y)
Vbus = Zbus*Ibus
```

Result:

Viva Questions:

1. What is a Bus?
2. What is the difference between Z-Bus and Y-Bus?
3. What is MATLAB?
4. What is the purpose of simulating in MATLAB software?
5. What are the advantages of MATLAB software?