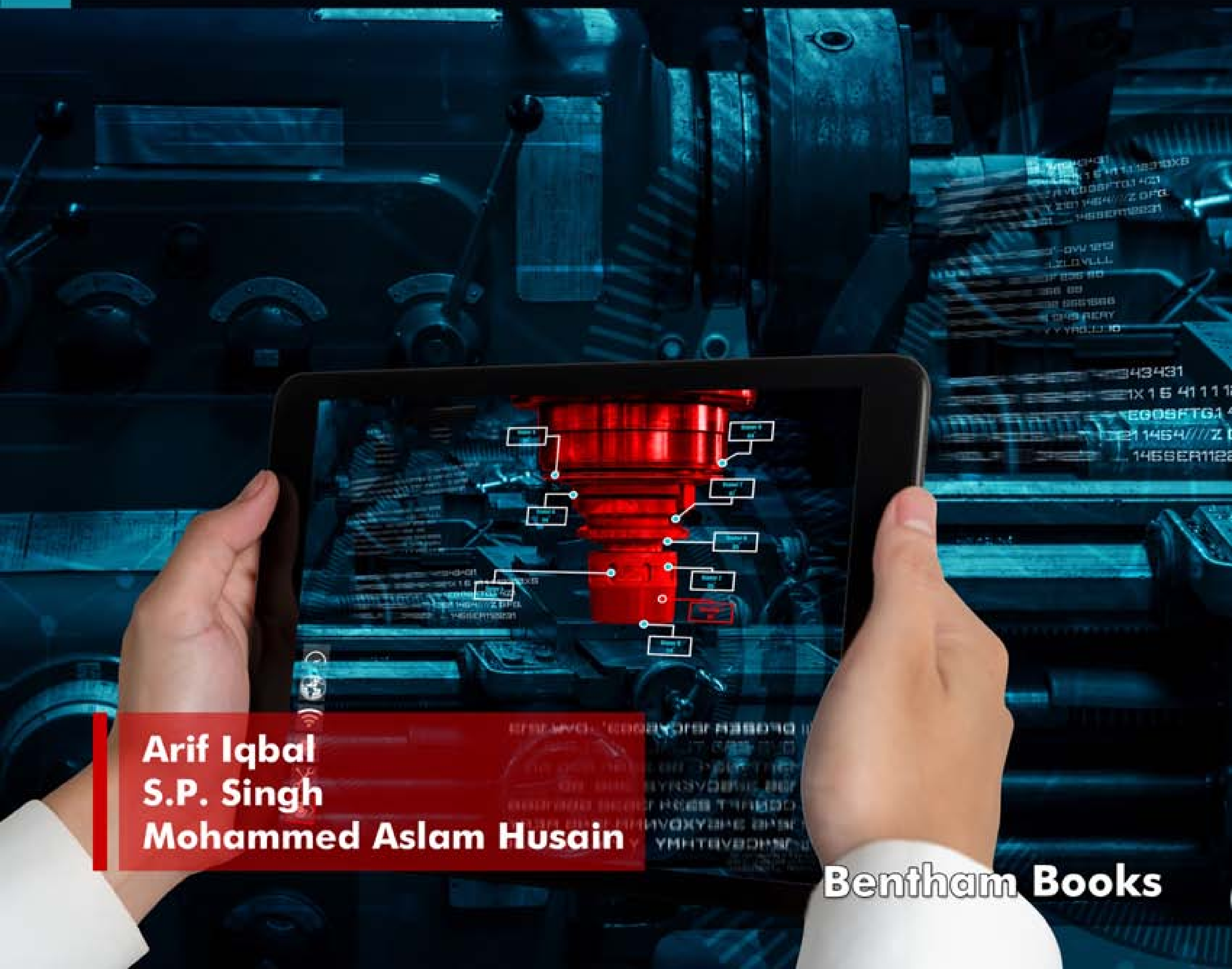
A HANDBOOK ON ENGINEERING COMPUTATION AND MODELING IN MATLAB AND SIMULINK



Arif Iqbal
S.P. Singh
Mohammed Aslam Husain

Bentham Books

A Handbook on Engineering Computation and Modeling in MATLAB and Simulink

Authored by

Arif Iqbal

S. P. Singh

Mohammed Aslam Husain

Department of Electrical Engineering
Rajkiya Engineering College
Ambedkar Nagar (U. P.)-224122
India

A Handbook on Engineering Computation and Modeling in MATLAB and Simulink

Authors: Arif Iqbal, S. P. Singh and Mohammed Aslam Husain

ISBN (Online): 979-8-89881-642-1

ISBN (Print): 979-8-89881-643-8

ISBN (Paperback): 979-8-89881-644-5

Published by Bentham Science Publishers Pte. Ltd. Singapore,

in collaboration with Eureka Conferences, USA.

First published in 2026.

need for a court order if at any point you breach any terms of this License Agreement. In no event will any delay or failure by Bentham Science Publishers in enforcing your compliance with this License Agreement constitute a waiver of any of its rights.

3. You acknowledge that you have read this License Agreement, and agree to be bound by its terms and conditions. To the extent that any other terms and conditions presented on any website of Bentham Science Publishers conflict with, or are inconsistent with, the terms and conditions set out in this License Agreement, you acknowledge that the terms and conditions set out in this License Agreement shall prevail.

Bentham Science Publishers Pte. Ltd.
80 Robinson Road #02-00
Singapore 068898
Singapore
Email: subscriptions@benthamscience.net

BENTHAM SCIENCE

CONTENTS

FOREWORD

In today's rapidly advancing technological landscape, the importance of computational tools in engineering education and practice cannot be overstated. MATLAB and Simulink have become indispensable platforms for engineers across various fields, offering powerful capabilities for computation, modeling, and simulation. With this handbook, **"Engineering Computation and Modeling in MATLAB and Simulink",** students and professionals alike are provided with a practical and accessible guide that builds foundational skills essential to engineering problem-solving and design.

This book is tailored for both Undergraduate (UG) and Postgraduate (PG) engineering students, as well as professionals seeking a quick yet comprehensive start in computational techniques. By organizing concepts in a clear and systematic manner, the book enables readers to delve into MATLAB/Simulink with confidence. Readers are guided from basic engineering computations to advanced modeling and simulation techniques, gaining the ability to develop and analyze systems in areas such as Electrical Power Engineering, Control Systems, Power Electronics, and Electrical Machines. This structured approach to learning opens doors to creating real-time prototypes, bridging the gap between theory and application.

With its thoughtfully designed Sections A and B, this book ensures a step-by-step progression, starting from built-in commands in MATLAB and advancing toward complex system modeling using Simulink. Worked examples and tutorials provide hands-on experience, enabling users to deepen their understanding and immediately apply what they've learned to real-world problems.

I commend the authors for this contribution to engineering education and professional development. This book stands as an invaluable resource, equipping its readers with the skills to harness MATLAB/Simulink effectively and to drive innovation in both academic and industrial settings.

G. Nalankilli
Rajkiya Engineering College
Ambedkar Nagar-224122, India

PREFACE

This book is intended to serve as a handbook for the Engineering students at the Undergraduate (UG) and Postgraduate (PG) levels of various Technical Institutions in an organized and easily understandable way.

This book is written primarily as an introduction to the basics of Engineering Computation and Modeling using MATLAB/Simulink. This book serves as a quick start point of modeling and simulation within a few hours of various Engineering problems by using various available toolboxes as a user-friendly interface.

This includes a quick simulation by using various available built-in functions together with the development of a block diagram using Simulink in MATLAB. The available functions and toolboxes of MATLAB/Simulink are used to study, design, and analyse the various circuits and systems, which can be used to develop a real-time prototype of the considered system.

We hope that the present book will be useful to the students and teachers as well.

Arif Iqbal
S. P. Singh
Mohammed Aslam Husain
Department of Electrical Engineering
Rajkiya Engineering College
Ambedkar Nagar (U. P.)-224122
India

Dedicated to our Parents

BACKGROUND AND PURPOSE

With rapid technological advancements across various engineering domains, computational and simulation tools have become integral to the development and analysis of complex systems. These tools enable in-depth understanding and swift iteration, allowing engineers and researchers to enhance and optimize designs effectively. Among these, MATLAB programming and model development using Simulink have gained widespread global adoption due to their versatility and ease of use.

The primary reasons for MATLAB's popularity include its user-friendly interface, an extensive range of built-in functions, and a broad selection of specialized toolboxes tailored for various applications. These features not only facilitate efficient simulation and modeling but also reduce the learning curve, enabling users to focus more on problem-solving and innovation. Consequently, MATLAB and Simulink have become indispensable resources for both academic and industrial applications, accelerating the pace of technological progress.

This book is designed to simplify the use of MATLAB and Simulink for engineers, researchers, and students alike, offering a comprehensive guide to simulations with detailed, step-by-step examples. Focusing on Electrical Engineering, it covers applications across key subdomains, providing readers with practical insights and tools to enhance their modeling and analytical skills. By following these examples, readers can deepen their understanding and directly apply simulations to real-world engineering challenges, empowering them to make meaningful contributions to technological progress.

ABSTRACT

"Engineering Computation and Modeling in MATLAB and Simulink" is a practical and comprehensive handbook designed to provide an easy entry into engineering computations, modeling, and simulation using MATLAB/Simulink. This book caters to a wide audience, from undergraduate and postgraduate students to working professionals, serving as a quick reference for simulating and analyzing various engineering systems.

Organized into two sections, this handbook offers a structured approach to mastering MATLAB/Simulink for engineering applications:

Section A introduces the basics of engineering computations using MATLAB's built-in commands, followed by illustrative examples and hands-on tutorials. Readers can start solving problems quickly, learning essential MATLAB functionalities through practical exercises.

Section B focuses on the modeling and simulation of engineering systems using Simulink, with detailed examples that guide users in designing systems within Electrical Power Engineering, Electrical Machines, Control Systems, and Power Electronics. Each simulation step is clearly explained, allowing readers to gain a deeper understanding of how to develop and analyze real-world engineering systems in a simulated environment.

Through worked-out examples and step-by-step instructions, this book supports academic, industrial, and R&D applications, enabling readers to extend their simulations to meet diverse analytical and investigative needs. With its focus on accessible yet comprehensive instruction, "Engineering Computation and Modeling using MATLAB and Simulink" is an essential resource for anyone looking to enhance their skills in this vital area of engineering.

Benefits to the Audience:

1. **Quick Start to MATLAB Programming:** This book serves as a quick guide to MATLAB programming, introducing readers to its vast library of built-in functions. These functions simplify tasks across multiple fields of Science and

Technology, making it an invaluable resource for users looking to solve complex problems efficiently.

2. **Hands-On Simulink Simulation:** Readers will gain a practical introduction to Simulink, allowing them to develop visual models and simulate systems with ease. This quick start in Simulink opens doors to dynamic modeling and simulation across all areas of Engineering and Technology, enhancing their ability to visualize and analyze complex systems.

3. **Broad Applicability for Students and Professionals:** Designed to benefit both students and working professionals, this book caters to individuals from diverse engineering domains. Whether for academic research or industrial application, it provides the foundational tools and knowledge to harness MATLAB and Simulink effectively, empowering readers to advance their technical skills and contribute to innovation in their fields.

These benefits collectively make the book a valuable resource, equipping readers with the skills to start their journey in MATLAB and Simulink and apply these tools in real-world scenarios.

CHAPTER 1

Introduction to MATLAB

Abstract: MATLAB, short for *Matrix Laboratory*, is a high-performance programming language and interactive environment developed by MathWorks. It is widely used in engineering, science, economics, and other fields for numerical computation, data analysis, and visualization. MATLAB's user-friendly interface, combined with its powerful built-in functions and toolboxes, makes it an indispensable tool for researchers, engineers, and students. This chapter provides an overview of MATLAB's key features, advantages, and applications. It highlights MATLAB's ease of use, platform independence, and extensive library of predefined functions, which simplify complex technical tasks. Additionally, the chapter explores MATLAB's capabilities in areas such as control systems, signal processing, data visualization, machine learning, and financial modeling. By integrating computation, visualization, and programming into a single environment, MATLAB remains a leading platform for technical computing and problem-solving across diverse domains.

Keywords: Data analysis, Data visualization, Engineering computation, MATLAB, Technical computing.

INTRODUCTION

MATLAB, an acronym for *Matrix Laboratory*, is a high-performance programming language and interactive environment designed for numerical computation, data analysis, and visualization [1]. Developed by MathWorks, MATLAB is widely used in engineering, science, economics, and other fields that require advanced mathematical computations and data processing. Its user-friendly interface, combined with powerful built-in functions and toolboxes, makes it an indispensable tool for researchers, engineers, and students alike. MATLAB's features include a rich set of built-in functions for mathematical computations, an intuitive programming environment, and the ability to integrate with other programming languages such as C, C++, and Python. It also offers tools for generating reports, deploying applications, and performing real-time simulations. With its combination of simplicity, power, and flexibility, MATLAB continues to be a leading platform for technical computing and problem-solving across diverse domains.

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

This chapter offers an overview of MATLAB's advantageous features, structured into three key sections. Section 1.2 outlines the advantages of MATLAB, emphasizing its ease of learning, platform independence, extensive library of predefined functions, and Graphical User Interface (GUI) capabilities, which collectively make it a preferred tool for technical computing. Section 1.3 examines the features of MATLAB, exploring its high-level language for numerical computation, interactive environment for iterative problem-solving, vast mathematical libraries, advanced visualization tools, and robust application development capabilities. Finally, section 1.4 explores the use of MATLAB across various domains, including engineering, scientific research, data visualization, machine learning, financial modeling, and education, showcasing its versatility and wide-ranging applications. Together, these sections provide a comprehensive overview of MATLAB's capabilities and its significance in addressing complex problems across diverse fields.

ADVANTAGES OF MATLAB

MATLAB has several advantages [2, 3] in comparison with other computational languages for computing engineering problems, which include the following:

- **Easy to learn:** It is more fun to self-learn MATLAB from available resources due to its user-friendly interface. Online help documentation can be accessed by typing "help" or "look for" in the command window.

- **Easy to use:** MATLAB is extremely easy to use. MATLAB is considered a technical computing language that is easy to use. Simple computation may be performed by typing the instructions in the command line, but for a complex problem, a program may be easily written by using hundreds of built-in functions with MATLAB debugger. MATLAB provides several tools to write programs, which are easy to use.

- **Platform independence:** MATLAB is supported by most of the system platforms, including all versions of Windows, Linux, Unix, and Macintosh. A program or data file written on a system platform may be read on any other platform. This makes the MATLAB operation system-independent.

- **Predefined functions:** Unlike other programming languages, MATLAB provides hundreds of predefined functions for many technical tasks, making the computations much easier. Additionally, MATLAB provides many special-purpose toolboxes for complex problems in specific engineering domains. For example, a separate toolbox is available for computation in Signal Processing, Communications, Control Systems, Neural Networks, Image Processing, etc.

- **Graphical User Interface (GUI):** In MATLAB, tools are available to construct an interactive Graphical User Interface (GUI) for any program. Hence, inexperienced users can also operate the program to analyse the sophisticated data.

FEATURES OF MATLAB

MATLAB is a powerful and versatile platform that offers a wide range of features that simplify numerical computation, data visualization, and application development. Below are the key features that make MATLAB a preferred tool for professionals, researchers, and students:

- It is a high-level programming language that is designed for numerical computation, data visualization, and application development.

- It also offers an interactive environment that supports iterative exploration, design, and problem-solving.

- It provides an extensive library of mathematical functions, covering areas such as linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations.

- It provides built-in graphics capabilities for data visualization, along with tools for creating customized plots and charts.

- MATLAB's programming interface provides development tools designed to enhance code quality, maintainability, and maximize performance.

- It offers tools for developing applications with customizable Graphical User Interfaces (GUIs).

- MATLAB includes functions that enable the integration of MATLAB-based algorithms with external applications and programming languages, such as C, Java, .NET, and Microsoft Excel.

CHAPTER 2

Fundamentals of MATLAB

Abstract: This chapter introduces the fundamental aspects of MATLAB, a high-level language known for its matrix-based operations and extensive built-in functions. MATLAB has become a widely used computational tool across various domains, replacing traditional programming languages such as FORTRAN and C/C++. The chapter briefly covers essential features, including the MATLAB desktop environment, editor window, figure window, help window, and input-output operations. Additionally, it presents data types, dimensions, case sensitivity, output display, and methods for starting MATLAB. Hence, the chapter provides a strong foundation for beginners to navigate and utilize MATLAB effectively.

Keywords: Command, Desktop, Editor, Help, Input-output, MATLAB, Window, Workspace.

INTRODUCTION

MATLAB has emerged as a powerful tool in scientific computing, engineering, data analysis, and beyond, often replacing traditional programming languages like FORTRAN and C/C++ due to its efficiency and versatility. MATLAB is considered a high-level language, based on Matrix operations. MATLAB is popular due to its user-friendliness and computational power with various built-in functions [1].

This chapter serves as a quick starting point for mastering MATLAB and leveraging its computational power. The following section familiarizes readers with the MATLAB desktop environment on all operating systems, which includes key components such as the Editor Window for writing and debugging scripts, the Figure Window for visualizing data and graphical outputs, and the Help Window for accessing documentation and learning about built-in functions. It also covers essential input-output operations, enabling users to interact with data files and display results effectively [2, 3].

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

MATLAB DESKTOP

This is the default window that appears when MATLAB is launched [2-4], as shown in Fig. (**1**). It consists of the following windows:

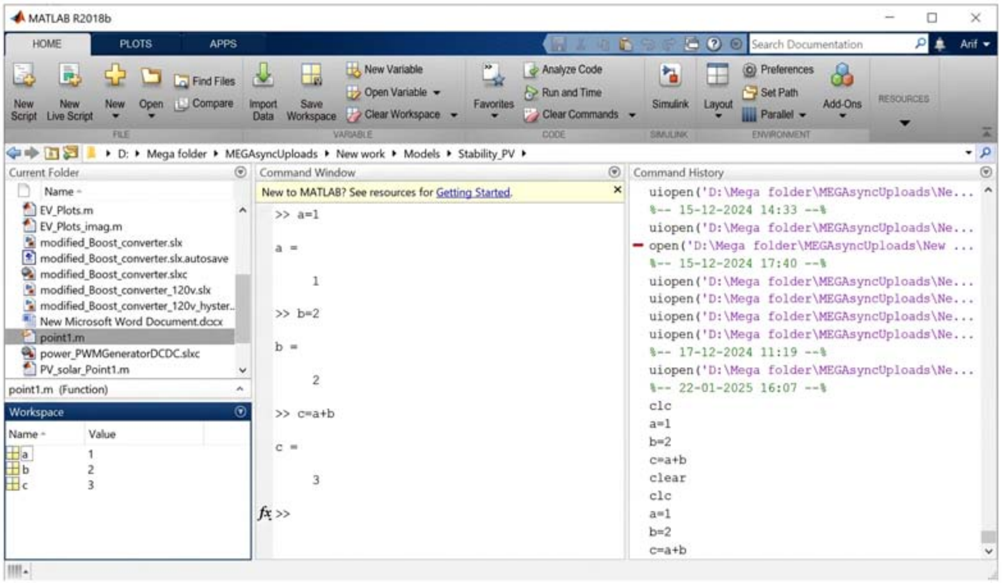


Fig. (1). MATLAB Desktop with subwindows

Command Window

The Command Window is the primary interface for MATLAB, identified by the MATLAB command prompt (>>). All commands, including those for executing user-written programs, are entered here at the prompt. Depending on the version of the installed software, this window may contain some sub-windows, which may be ignored by beginners. As the software evolves, developers often include advanced features to cater to experienced users, but these can be explored later.

Current Directory Window

The Current Directory Window is located to the left of the Command Window in MATLAB's default desktop layout. This pane displays all the files in the current working directory and allows for easy file navigation. Ensure that this is the directory where the user intends to work so MATLAB can access the files and save any newly created files.

If the current directory is changed by navigating through the file system, it should be verified that the change is also reflected in the small window above the Command Window labeled *Current Directory*. These two are interconnected—modifying the directory in one automatically updates the other.

Once a file is selected (by clicking on it), you can right-click to access various options, such as running m-files, renaming, or deleting them. This functionality streamlines file management within MATLAB.

Workspace

The Workspace sub-window lists all the generated variables, along with their types and sizes. It provides tools for managing, viewing, and interacting with these variables efficiently. Therefore, variables can be further used (such as in a graphical plot) by using the options available via right-clicking the mouse.

Command History

This pane records all commands entered in the Command Window, even across multiple sessions (*e.g.*, work done on different days such as Monday, Thursday, and the following Wednesday). Any previously entered command can be executed by double-clicking it in this pane. Additionally, multiple commands can be selected to create an m-file by using a right-click and choosing the relevant option from the context menu.

EDITOR WINDOW

In this window, a program, *i.e.,* a set of commands, is created, edited, and saved in an M-file using any text editor of the system [2]. The same commands in the m-file can also be executed in this window, as well as by calling the file from the command window. This window provides a flexible editing environment best suited to our workflow. The Editor window showing a simple program to create a straight line is shown in Fig. (**2**).

CHAPTER 3

Toolboxes in MATLAB

Abstract: MATLAB offers a suite of add-ons, application-specific solutions known as toolboxes, which are essential for users seeking to learn and apply specialized technologies. These toolboxes are extensive collections of MATLAB functions (m-files) that expand the MATLAB environment to address specific types of problems. Developed by MathWorks, Inc., these toolboxes are integral to the MATLAB environment and serve as a graphical extension of the system. Their seamless integration with MATLAB makes them a valuable resource for engineers and researchers across multiple engineering domains. Additionally, Simulink, another key component of the MATLAB family, provides a graphical platform for modeling and analyzing various systems. Together, MATLAB toolboxes and Simulink enable users to tackle complex challenges through advanced computational and modeling capabilities. A concise introduction has been provided for important toolboxes frequently used in MATLAB.

Keywords: Graphical User Interface (GUI), MATLAB, Simulink, Toolboxes.

INTRODUCTION

MATLAB features a family of add-on, application-specific solutions called the toolboxes. These are very important to most users of MATLAB. A toolbox allows you to learn and apply specialized technology [1]. Toolboxes are comprehensive collections of MATLAB functions (m-files) that extend the MATLAB environment to solve a particular class of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. MATLAB offers over 100 toolboxes [2]. Thus, we can say MATLAB basically works as a platform, and for solving a particular problem, the concerned toolbox is required. A few of the toolboxes frequently used in MATLAB are as follows:

•***Control System Toolbox™*** is used for ~~a~~ systematic analysis, design, and tuning of controllers. Using this toolbox, a system can be represented by using various models, particularly, state-space, transfer function, frequency-response, or pole-zero-gain. System behavior may be analyzed and visualized both in the time and frequency domains. This Toolbox is capable of auto-tuning compensators for SISO and MIMO systems using PID controllers with the required tuning objectives.

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

•***Signal Processing Toolbox™*** is a powerful toolbox used for the analysis, pre-processing, and feature extraction of sampled signals (uniform and non-uniform). The toolbox is also used to design filter circuits and perform their analysis, resampling, and smoothing, along with the estimation of the power spectrum. It is also used for its feature extraction functionality, particularly the determination of peak value, signal patterns, signal comparison, and performance indices (SNR and distortion). The Signal Analyzer app in the toolbox is also useful for analyzing multiple signals simultaneously in the time and frequency domains without the need for user-written programming codes.

•***Curve Fitting Toolbox™*** is widely used to fit the provided data into a curve and/or surface through predefined functions. Hence, it is well-suited to perform exploratory data analysis during pre- and post-processing of data, enabling quick comparison and removal of outliers. Regression analysis may also be conducted using various available linear and non-linear model libraries or user-defined equations. Non-parametric modeling techniques (such as interpolation, splines, and smoothing) are also supported in the toolbox.

•***Mapping Toolbox™*** uses functions and algorithms to transform geographic data and create display maps. Here, data are visualized in a geographic context, where maps are built and displayed using more than 60 map projections, and data from different sources are transformed into a consistent geographic coordinate system. The toolbox is useful for processing and customizing data using many techniques, particularly trimming, resampling, interpolation, and coordinate transformations. Data can be exported in file formats such as GeoTIFF, shapefile, and KML.

•***SimPowerSystem™*** (also known as Simscape Electrical™ and SimElectronics® depending on the installed version of MATLAB) consists of component libraries used for the design and simulation of electrical and electronics circuits, particularly, power electronics, electric drives, and power systems. Toolbox is also capable of simulating both simple and complex systems, like renewable energy systems and smart grids, for their performance evaluation. It may be used to design and implement various control systems to regulate electrical power systems. Other physical systems, like hydraulic, mechanical, and thermal systems, may be integrated with the developed electrical power system. The developed model can also be used on another simulation platform, particularly Hardware-In-the-Loop (HIL) systems, through C-code generation.

•***Communication Toolbox™*** is a combination of algorithms and applications to design, simulate, analyse, and verify communication systems. It is suitable for designing wireless communication systems, where the toolbox algorithms are used in modulation schemes, channel coding, and simulation of physical layer models, which can be standards-based or customized. This toolbox is also used for signal analysis, visualization of channel characteristics, and other performance indices. A communication system can be verified by connecting it with transmitter and receiver models using hardware support packages or RF instruments.

•***Fuzzy logic toolbox™*** consists of MATLAB functions, applications, and a Simulink block that are used to design, analyse, and simulate systems that operate on fuzzy logic. This toolbox is suitable for modeling the behaviour of complex systems using simple logic rules, implemented in a fuzzy inference system. It can also be used with other blocks in Simulink during the design of a comprehensive dynamic system.

•***Optimization Toolbox™*** consists of functions to determine the parameters resulting in the minimization and maximization of objectives within defined constraints. This toolbox operates by using different solvers, such as Linear Programming (LP), Nonlinear Programming (NLP), quadratic programming, Mixed-Integer Linear Programming (MILP), Second-Order Cone Programming (SOCP), nonlinear least squares, constrained linear least squares, and nonlinear equations. Optimal solutions may be obtained by using the toolbox's solvers in both continuous and discrete systems across different applications.

•***Image Processing Toolbox™*** consists of a comprehensive collection of standard algorithms for image processing, visualization, analysis, and algorithm development. Using this toolbox, some important image processing tasks are segmentation, enhancement, noise reduction, registration, transformation, and 3-D image processing.

•***Deep Learning Toolbox™*** is used to design and implement deep neural networks using **various** algorithms and applications. Conventional and Long-Short-Term Memory **(LSTM)** neural networks are used for classification and regression tasks on text data, time series, and images. Applications in Deep Learning Toolbox are used with graphical interfaces for designing, analysing, and training networks. The experiment manager application in the toolbox is suitable for effective management of experiments involving multiple deep learning models, tracking training parameters, analyzing results, and comparing code used in different experiments.

CHAPTER 4

Basic Mathematics

Abstract: MATLAB provides an extensive collection of mathematical functions, operators, and commands, enabling users to efficiently perform calculations, manipulate matrices and arrays, execute trigonometric operations, and solve linear equations. This chapter introduces fundamental mathematical operations in MATLAB, covering basic matrix and array manipulations, trigonometric functions, exponential and complex number operations, and script file execution. Additionally, methods for solving linear equations, computing eigenvalues and eigenvectors, and finding polynomial roots are discussed. With worked-out examples, this chapter aims to strengthen the reader's understanding of MATLAB's mathematical capabilities and its application in solving engineering and scientific problems.

Keywords: Array, Eigenvector, Eigenvalue, Functions, Linear equations, Matrix, Polynomials, Roots, Script file.

INTRODUCTION

MATLAB's robust mathematical capabilities form the foundation of its widespread usage in engineering, science, and applied mathematics. The platform includes built-in functions for handling matrix and array operations, trigonometric calculations, and algebraic computations with ease [1]. This chapter begins with an introduction to basic matrix and array operations, distinguishing between array-based and matrix-based computations. It then delves into essential mathematical functions, such as trigonometric, exponential, and complex number operations, demonstrating their applications through examples. Furthermore, MATLAB's capability to execute a sequence of commands using script files is explored, highlighting its advantages over the command window for program execution. The latter sections cover key mathematical problem-solving techniques, including solving systems of linear equations, computing eigenvalues and eigenvectors, and determining the roots of polynomials. Throughout this chapter, worked-out examples facilitate a hands-on learning approach, ensuring that readers gain practical proficiency in MATLAB's mathematical tools.

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

BASIC MATRIX AND ARRAY OPERATIONS

Two types of operations are supported in MATLAB, known as array and matrix operations [2]. In array operation, element-by-element operation between arrays is performed, whereas the rules of linear algebra are followed under matrix operation.

Considering the matrix/array operation between the given two matrices A and B, the frequently used commands for Arrays and Matrices are given in Tables **1** and **2**, respectively.

Table 1. Array operations in MATLAB.

S. No.	Operation	Command
1.	Addition	A+B
2.	Subtraction	A-B
3.	Multiplication	A.*B
4.	Right Division	A./B
5.	Left Division	A.\B
6.	Exponentiation	A.^B

Table 2: Matrix operations in MATLAB.

S. No.	Operation	Command
1.	Multiplication	A*B
2.	Right Division	A/B
3.	Left Division	A\B
4.	Inverse of Matrix A	inv(A)

Worked-out Examples

All the mathematical operations in MATLAB are elaborated in the examples in the following sections:

Basic Matrix Operations

Example 4.2.1: With the two 3×3 matrices A and B defined as:

$$A = \begin{bmatrix} 3 & 2 & 1 \\ 0 & 3 & 4 \\ -1 & 1 & -1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 & 0 \\ 2 & 6 & 4 \\ -1 & 0 & 2 \end{bmatrix}$$

Perform the following matrix operations:

(a) A+B

(b) A-B

(c) A*B

(d) A/B

(e) A\B

(f) inverse A

Solution:

(a) Addition of A and B

Command for A+B:

\>\> A= [3 2 1; 0 3 4; -1 1 -1];

\>\> B= [1 3 0; 2 6 4; -1 0 2];

\>\> A+B

The executed commands and results in MATLAB are shown in Fig. (**1**).

```
Command Window
>> A= [3   2   1;   0   3   4; -1   1    -1];
>> B= [1   3   0;   2   6   4; -1   0   2];
>> A+B

ans =

     4     5     1
     2     9     8
    -2     1     1
```

Fig. (1). Executed commands and results in the Command Window.

CHAPTER 5

Graphical Plots in MATLAB

Abstract: Graphical representation of data is a crucial aspect of scientific and engineering analysis. MATLAB provides a comprehensive set of plotting tools for both two-dimensional and three-dimensional data visualization. This chapter explores various graphical plotting techniques in MATLAB, including standard linear-axis plots, logarithmic plots, bar plots, polar plots, and surface plots. It also covers multiple graph plotting methods, special graphics, and curve fitting techniques. Through numerous worked-out examples, this chapter enables the reader to develop a thorough understanding of MATLAB's plotting capabilities with their engineering applications.

Keywords: 2-D, 3-D, Curve fitting, MATLAB, Plots, Subplots, Special graphics.

INTRODUCTION

Graphical plots are a critical tool for data visualization, particularly in fields such as science and engineering, where MATLAB is widely employed. MATLAB offers a comprehensive set of commands to generate diverse types of plots, including standard linear-axis plots, logarithmic and semi-logarithmic plots, bar and stair plots, polar plots, and three-dimensional contour, surface, and mesh plots [1, 2]. These plots can be customized to suit specific requirements by adjusting line styles (e.g., solid, dashed), colors, and thickness. Additionally, features such as line markers, grid lines, titles, and text annotations can be added to improve clarity and interpretation. MATLAB also allows for the display of multiple graphs within a single plot or the arrangement of several plots on the same page. To enhance readability when working with multiple datasets or graphs, a legend can be included, making it easier to distinguish between different data series. This flexibility and functionality make MATLAB a powerful tool for creating detailed and informative visualizations [2].

This chapter introduces fundamental plotting techniques in MATLAB, beginning with basic 2D plots using the plot function, followed by advanced multi-graph plotting techniques utilizing hold on, hold off, and the line command. Special graphical representations such as bar graphs, stem plots, pie charts, and stair plots are also covered. In the 3D domain, MATLAB offers various visualization tools, including mesh, surface, and contour plots, which help in representing complex

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

datasets. Additionally, curve fitting techniques using polynomial, exponential, logarithmic, and power functions are explained with step-by-step worked-out examples.

TWO-DIMENSIONAL PLOTS

The plot command is used to create two-dimensional plots between two variable vectors, say x and y [2, 3]. The plot command also accepts optional arguments that allow customization of: line's color and style, as well as the marker color and type, if desired. With these arguments, the command syntax is :
plot(x,y,‘line specifiers’,‘PropertyName’,PropertyValue)
For more detailed plotting options, consult MATLAB's documentation by using the command help plot, as illustrated in Fig. (**1**). The following worked examples demonstrate MATLAB's plotting capabilities:

Example 5.2.1: Generate a MATLAB plot of the curves given in Table **1**:

Table 1. Curve types and their equation.

S. No.	Curve Type	Equation	Range
1.	Straight Line	$y = 3x + 2$	$0 \leq x \leq 3$
2.	Sine function	$y = \sin(x)$	$0 \leq x \leq 4\pi$
3.	Cosine function	$y = \cos(x)$	$0 \leq x \leq 4\pi$

Solution: For all curve types mentioned above, the corresponding plotting commands should be written in a script file

Straight Line

Script file executed is:

```
x=0:0.1:3;           % defines range of 'x'(starting value
        % 0, end value 3 with interval of 0.1)
y =2*x+3;            % write given equation
plot(x, y,'*')       % plot with points marked '*'
xlabel('x');         % x-axis is marked as x
ylabel ('y= 2*x +3');  % y-axis is marked as y=2*x+3
title('straight line')
```

Command Window

```
>> help plot
 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, disconnected
    line objects are created and plotted as discrete points vertically at
    X.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

           b     blue          .     point              -     solid
           g     green         o     circle             :     dotted
           r     red           x     x-mark             -.    dashdot
           c     cyan          +     plus               --    dashed
           m     magenta       *     star             (none)  no line
           y     yellow        s     square
           k     black         d     diamond
           w     white         v     triangle (down)
                               ^     triangle (up)
                               <     triangle (left)
                               >     triangle (right)
                               p     pentagram
                               h     hexagram
```

Fig. (1). Help for a graphical 2D plot

The resulting figure after execution of commands is shown in Fig. (**2**).

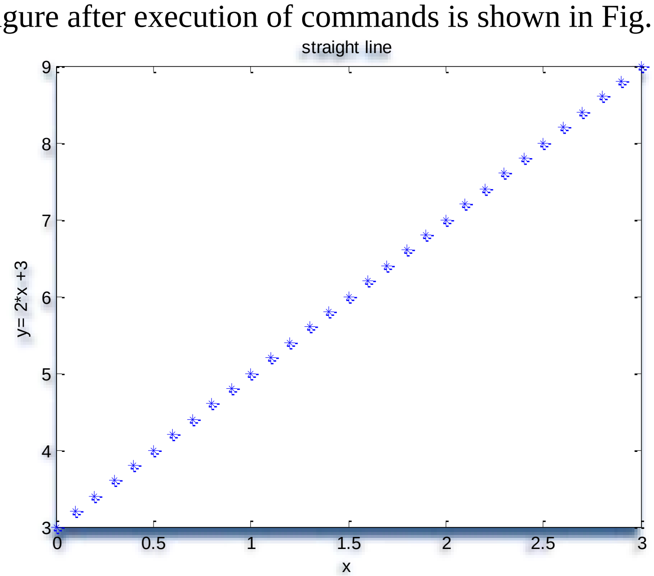


Fig. (2). Plot of a straight line.

CHAPTER 6

Solutions to Differential Equations

Abstract: This chapter explores the application of MATLAB in solving differential equations, a fundamental aspect of engineering and scientific computations. It covers essential mathematical operations such as differentiation and integration, solving Ordinary Differential Equations (ODEs), analyzing time responses of R-L-C circuits, determining the frequency response of transfer functions, and computing Laplace and inverse Laplace transforms. By leveraging MATLAB's built-in functions like diff (), int (), dsolve (), tf (), and laplace(), users can efficiently analyze and model complex engineering problems. The chapter provides step-by-step worked examples and practice questions to ensure a thorough understanding and hands-on learning for students and researchers.

Keywords: Differentiation, Frequency response, Inverse-laplace, Integration, Laplace, Ordinary Differential Equation (ODE), Plots, R-L-C circuit, Transfer function.

INTRODUCTION

Differential equations form the backbone of numerous engineering applications, particularly in electrical circuit analysis, control systems, and signal processing. MATLAB, a powerful computational tool, simplifies the process of solving these equations by offering a range of built-in functions that automate complex calculations [1-4]. This chapter aims to equip readers with the necessary skills to use MATLAB for solving differential equations, covering differentiation, integration, and the symbolic and numerical solution of ODEs. Users will be tutored on solving differential equations through built-in MATLAB functions such as `diff()`, `int()`, and `laplace()`, etc. Furthermore, the analysis of R-L-C circuits, frequency response of transfer functions, and Laplace transformations is introduced with practical examples. By the end of this chapter, readers will gain proficiency in applying MATLAB's capabilities to solve real-world engineering problems efficiently.

DIFFERENTIATION AND INTEGRATION

The main mathematical operations of differentiation and integration are widely employed in engineering and science [5]. MATLAB provides built-in functions such as `diff()`, `gradient()`, and `int()` for numerical and symbolic computation.

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

The section details the application of these functions and their use in solving real-world problems. The tutorials will provide readers with step-by-step guidance on differentiation and integration techniques using MATLAB.

Commands for Differentiation

- diff(*f*) differentiates the symbolic expression *f* with respect to the default variable *t*.
- diff (*f*, *x*) differentiates the function *f* with respect to variable *x*
- diff (*f*, *n*) differentiates the function *f* by *n* times, where *n* is a positive integer.

Example 6.2.1: Differentiate the following:

(a) sin(*x*)
(b) sin(*x*) with respect to *x*
(c) sin(*x*), two times

Solution: MATLAB commands executed by the script file are given below, where syms x is used to declare a symbolic variable.

(a)The commands executed are as follows:

```
syms x
diff_sin = diff(sin(x))
disp ('First derivative of sin(x):')
disp(diff_sin)
```

Output:

```
diff_sin =
 cos(x)
 First derivative of sin(x):
cos(x)
```

(b)The commands executed are as follows:

```
syms x
diff_sin_wrt_x = diff(sin(x), x)
```

```
disp ('Derivative of sin(x) with respect to x:')
disp(diff_sin_wrt_x)
```

Output:

```
diff_sin_wrt_x =
 cos(x)
 Derivative of sin(x) with respect to x:
cos(x)
```

(c)The commands executed are as follows:

```
syms x
second_diff_sin = diff(sin(x), x, 2)
disp ('Second derivative of sin(x):')
disp(second_diff_sin)
```

Output:

```
second_diff_sin =
 -sin(x)
 Second derivative of sin(x):
-sin(x)
```

Example 6.2.2: Differentiate the following:

(a) x^5
(b) x^5 **w.r.t** x
(c) x^5 **four times**

Solution: MATLAB commands implemented in the script file are:

(a)The commands executed are as follows:

```
syms x
diff_x5 = diff(x^5)
disp ('First derivative of x^5:')
disp(diff_x5)
```

CHAPTER 7

Fundamentals of Simulink

Abstract: Simulink, an integral part of MATLAB®, serves as a powerful tool for modeling, simulating, and analyzing dynamic systems across multiple engineering domains. This chapter introduces Simulink as an efficient graphical simulation tool, providing an overview of its commonly used blocks, simulation settings, and applications. Through a series of worked-out examples, the chapter demonstrates the implementation of basic waveform generation, the simulation of differential equations, and the modeling of electrical circuits. Additionally, it explores advanced features such as custom block creation and the use of the *SimPowerSystems* toolbox for real-time system design. Overall, this chapter enhances the fundamental skills required to utilize Simulink effectively in engineering simulations and system design.

Keywords: Configuration, Custom block, Differential equation, Modeling, RLC circuit, Simulink, Simulation.

INTRODUCTION

Simulink, a product of MathWorks, is widely recognized as a versatile graphical programming environment for system modeling and simulation. It offers an intuitive, block-diagram-based approach to designing and analyzing dynamic systems. Simulink extends MATLAB's capabilities, allowing users to construct complex models with ease and simulate their behavior in real-time [1]. Its extensive library of pre-built blocks facilitates the design of various engineering applications, ranging from control systems to signal processing and power electronics.

This chapter provides a quick introduction to Simulink, beginning with its basic functionalities and commonly used blocks. This includes the initiation of Simulink, configuration of simulation parameters, and utilization of essential components for system modeling. The chapter also presents a variety of practical examples, including waveform generation, differential equation simulation, and RLC circuit analysis. Additionally, it delves into the creation of custom blocks and the integration of the *SimPowerSystems* toolbox, which expands Simulink's capabilities for power system simulations. Through the various worked-out examples, a strong foundation is built to effectively employ Simulink in multi-domain engineering applications.

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

STARTING WITH SIMULINK

Before starting Simulink, make sure that it has been installed in your MATLAB software. Once Matlab is launched, Simulink can be started by typing and executing in the Command Window [2, 3],

```
>>simulink
```

Alternatively, we can click on the Simulink icon, as shown in Fig. (**1**), which appears on the top bar of MATLAB's Command Window.

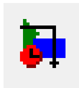


Fig. (1). Simulink icon

Upon execution of the Simulink, the various libraries will be opened, as shown in Fig. (**2**). From the Simulink Library Browser, a new Simulink model may be created with:

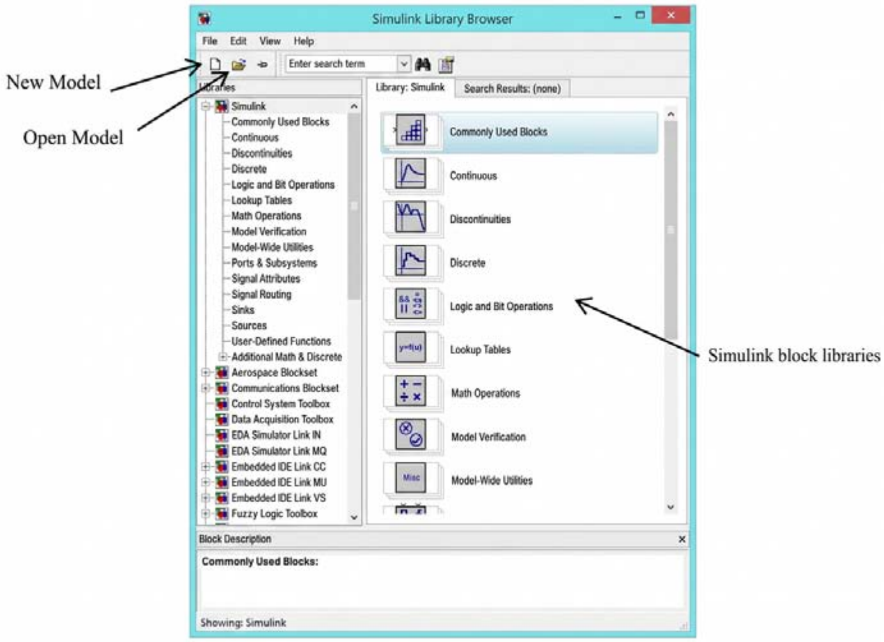


Fig. (2). Simulink Library Browser

File →New →Model

or simply by clicking the '*New Model*' icon in the *Library Browser Window* toolbar. An existing model may be opened with:

File →Open

or by clicking on the '*Open Model*' icon in the toolbar.

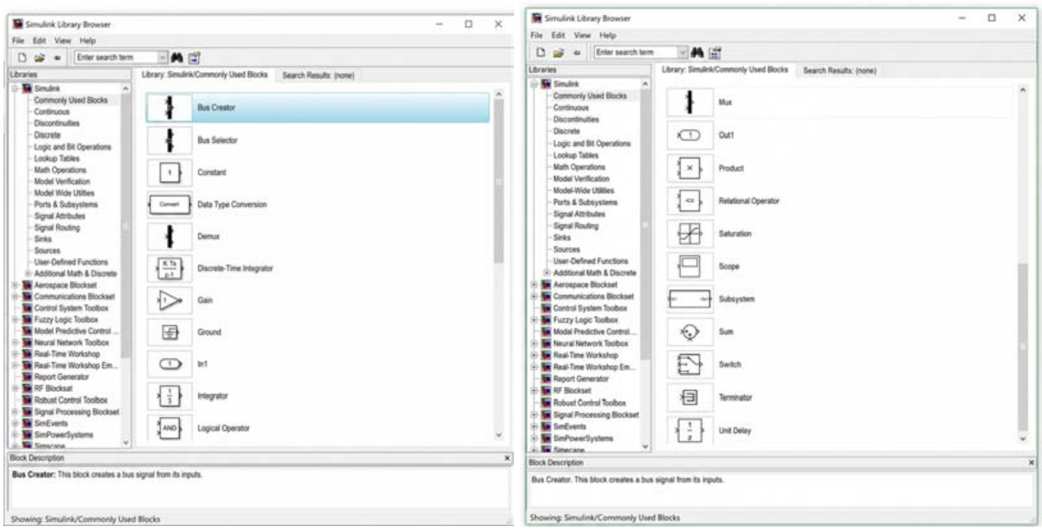
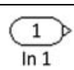
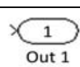


Fig. (3). Commonly used Simulink blocks

Commonly Used Blocks

The Simulink blocks that are commonly used can be found under the library '*Commonly Used Blocks*', shown in Fig. (**3**). Some of these blocks are explained [4, 5] in Table **1**.

Table 1. Commonly used blocks of the Simulink library.

Blocks	Explanation
	It is a port that serves as a link from outside a system into the system.
	It is a block that links from a system to a destination outside the system.

CHAPTER 8

Applications of Simulink in Electrical Engineering

Abstract: Simulink, a powerful simulation tool within the MATLAB environment, has become an essential platform for modeling and analyzing electrical engineering systems. This chapter explores the diverse applications of Simulink in electrical engineering, covering various domains such as electrical machines, control systems, power systems, power electronics, indirect Field-Oriented Control (FOC) in mechatronic systems, and renewable energy sources. Each section provides theoretical insights, step-by-step simulations, and practical implementation techniques with key results. The chapter also includes solved examples to enhance conceptual understanding and practice questions to reinforce learning. The comprehensive coverage of applications makes this chapter a valuable resource for readers in the field of electrical engineering.

Keywords: Control system, Electric motor, Field-oriented control, Power system, Power electronics, Simulink, Solar PV.

INTRODUCTION

The growing complexity of modern electrical engineering systems demands the use of advanced simulation tools to effectively analyze, design, and optimize these systems. Simulink, a powerful tool developed by MathWorks®, addresses this need by offering a user-friendly graphical interface for modeling and simulating dynamic systems [1]. With its extensive library of pre-built blocks and toolboxes specifically designed for electrical engineering applications, Simulink simplifies the process of creating accurate and efficient models. This chapter is designed to provide a comprehensive introduction to the use of Simulink in key areas of electrical engineering. It covers a wide range of topics, including the modeling and analysis of electrical machines, the design and analysis of control systems, power systems, power electronic converters, the implementation of indirect Field-Oriented Control (FOC) in mechatronic systems, and renewable energy (*e.g.*, solar PV) systems. By exploring these applications, a practical insight will be gained, paving the way to solve real-world engineering challenges using Simulink as an indispensable tool for both learning and professional development in the field of electrical engineering.

The chapter begins with an exploration of electrical machine simulations, *i.e.*, Permanent Magnet DC Motors and Three-Phase Induction Motors, supported by

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

their mathematical modelling. Next, it delves into control systems, where state-space modeling and state feedback controller design are implemented. The power system section highlights simulations related to Load Frequency Control (LFC) and Two-Area Automatic Generation Control (AGC), which are crucial for system stability. Further, in the power electronics section, different converter topologies such as controlled rectifiers, choppers, and three-phase PWM inverters are analyzed through simulation. The chapter also introduces Field-Oriented Control (FOC) techniques in mechatronics, covering axis transformation with the complete implementation of indirect FOC. Finally, the renewable energy segment explores the Photovoltaic (PV) system modeling and presents detailed simulation results. The chapter concludes with tutorials that reinforce the concepts discussed, ensuring that readers gain practical expertise in Simulink-based electrical engineering simulations.

ELECTRICAL MACHINES

The domain of electrical machines deals with a wide range of machines, including AC and DC types, which are used both in motoring and generating modes in different applications. In the following section, the simulation of some important machines is worked out using Simulink.

Simulation of Permanent Magnet DC Motor (PMDC Motor)

Simulation of a PMDC motor [2, 3] in Simulink has been carried out using the parameters given in Table **1**.

Table 1: PMDC Motor parameters.

Armature resistance $r_a = 1\Omega$	Friction coefficient $B = 0.000005$ *N-m-sec/rad*
Armature inductance $L_a = 0.02\ H$	Constant $k_a = 0.3\ V - sec/rad$
Moment of inertia $J = 0.0001\ kg - m^2$	

The applied armature voltage is *40 V*, and the load torque is 0.5*N-m*. Initial conditions must be used.

For the parameters mentioned above, the following are the two differential equations used for the simulation of the motor, whose *s*-domain block diagram is shown in Fig. (**1**).

$$\frac{di_a}{dt} = -\frac{r_a}{L_a} i_a - \frac{k_a}{L_a}\omega_r + \frac{1}{L_a} v_a, \qquad \textbf{(8.2.1)}$$

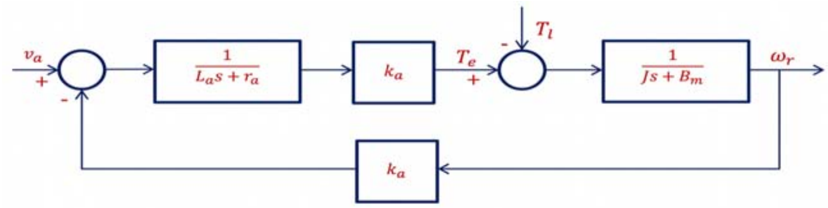


Fig. (1). Block diagram of a Permanent-Magnet DC motor.

$$\frac{d\omega_r}{dt} = -\frac{k_a}{J} i_a - \frac{B_m}{J}\omega_r - \frac{1}{J}T_l \quad \textbf{(8.2.2)}$$

By using s-domain modeling, the motor equations can be easily used to develop the Simulink model shown in Fig. (**2**). It is to be noted that the value of the initial condition was set as $x_0 = \begin{bmatrix} x_{10} \\ x_{20} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ for *integrator 1* and *integrator 2*, as shown in Fig. (**3**). DC input value of 40 V was initialized at time t = 0.5 sec., and load torque was applied at time *t = 1* sec. from *Source Block Parameters* dialog block for $v_a$and T_l, shown in Fig. (**4a** and **4b**), respectively. After running the developed Simulink model with the stop time set to 1.5 sec., and selecting the solved as: Fixed step (with a step size of 0.0001 sec.) and ode4 (Runge-Kutta), the motor response for current i_a and speed ω_r can be easily viewed, as shown in Fig. (**5a** and **5b**), respectively. The motor responses can be also plotted in figure file from the exported output data using the '*To Workspace*' block, as shown in Fig. (**6a** and **6 b**).

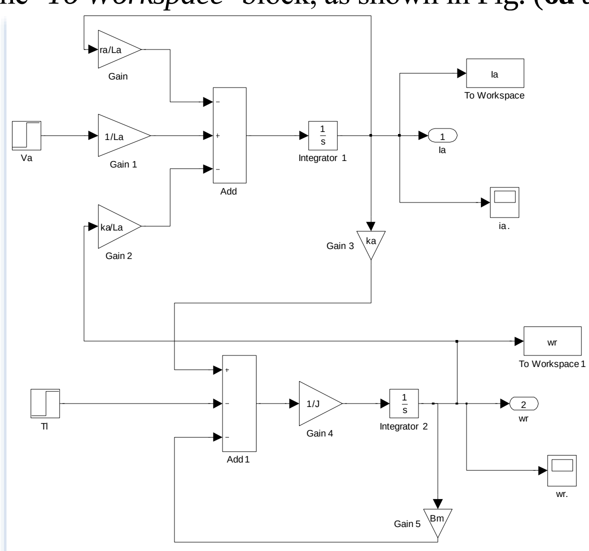


Fig. (2). Developed Simulink model of a Permanent-Magnet DC motor.

SUBJECT INDEX

Arif Iqbal, S. P. Singh & Mohammed Aslam Husain

E

F

G

H

I

K

L

M

N

O

P

R

S

T

V

W

Z

Arif Iqbal

Dr. Arif Iqbal received his B.Tech. and M.Tech. degrees in Electrical Engineering from Aligarh Muslim University, India, in 2005 and 2007, respectively, and earned his Ph.D. from IIT Roorkee in 2015. He has more than a decade of combined experience in industry and academia. Currently, he serves as an Assistant Professor in the Department of Electrical Engineering at Rajkiya Engineering College, Ambedkar Nagar, India. His research interests include multiphase AC machines and drives, power electronics, and renewable energy systems. He has published extensively in reputed journals and authored a book on virtual experimentation in electrical engineering.

S.P. Singh

Dr. S.P. Singh is an Associate Professor and Head of the Department of Electrical Engineering at Rajkiya Engineering College, Ambedkar Nagar, Uttar Pradesh, India. He holds a B.E. in Electrical and Electronics Engineering, an M.Tech. in Control and Instrumentation, and a Ph.D. in Power Electronics from MNNIT Allahabad. With over 17 years of academic experience, he has supervised 35 M.Tech. dissertations, two Ph.D. scholars, and is currently guiding three doctoral candidates. His research interests include Power Electronics and Drives, Renewable Energy Systems, and Power Quality Improvement. He received the Best Teacher Award–2024 from AKTU, Lucknow.

Mohammed Aslam Husain

Mohammed Aslam Husain received his B.Tech., M.Tech. in Power Systems and Drives, and Ph.D. in Power Electronics and Renewable Energy from Zakir Husain College of Engineering and Technology, Aligarh Muslim University, India. Since December 2017, he has been serving as an Assistant Professor in Electrical Engineering Department at Rajkiya Engineering College, Ambedkar Nagar, Uttar Pradesh. He is a recipient of the IEI Young Engineers Award 2022–23, the Young Scientist Award from the Council of Science and Technology, Uttar Pradesh, and the CSIR Senior Research Fellowship (2016). His research interests include power electronics, renewable energy systems, and sensorless control.