Practical MATLAB and Python

A comparative approach to problem-solving to code like a pro

Dr. Mamta Kapoor

Dr. Geeta Arora

www.bpbonline.com

First Edition 2026

Copyright © BPB Publications, India

ISBN: 978-93-65891-263

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they cannot be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

To View Complete
BPB Publications Catalogue
Scan the QR Code:

www.bpbonline.com

Dedicated to

This book is dedicated to Almighty God for giving me the strength that illuminated every step of this journey. Furthermore, it is dedicated to my beloved parents, whose blessings, love, and sacrifices have been the foundation for the creation of this work.

- Dr. Mamta Kapoor

This book is dedicated to the Divine Power that has blessed me with a family whose love, care, and support have empowered me to chase my aspirations and realize my potential.

- Dr. Geeta Arora

About the Authors

- **Dr. Mamta Kapoor** is a mathematician and researcher. Her research areas are numerical approximation of linear and non-linear partial differential equations, semi-analytical solution of fractional partial differential equations, fractional calculus, fuzzy calculus, scientific computing, numerical analysis, and fluid mechanics. She has 58 research publications in national/international journals of repute (Scopus indexed). Along with this research, she has a number of certifications in data science due to her deep interest in this field. Some of the related certifications are: Professional Certificate Course in data science from E & ICT Academy, IIT Kanpur; Advanced Certification in applied data science, Machine Learning & IoT organized by E & ICT Academy, Indian Institute of Technology, Guwahati; etc.

- **Dr. Geeta Arora** is a mathematician and researcher. She earned her Ph.D. in mathematics from IIT Roorkee in 2011. Her research focuses on the development of numerical methods and statistics, with specific interests in computational techniques, numerical analysis, partial differential equations, working with collocation methods, differential quadrature techniques, wavelets, and radial basis functions. She has a substantial publication record, including around 85 research papers (Scopus indexed) in international and national journals. Additionally, she has authored 15 book chapters, written books on Vedic Mathematics and Essential Statistics, and edited several books on numerical methods. She has also conducted various short-term courses and workshops on MATLAB programming and applications, along with Vedic Mathematics.

About the Reviewers

- **Vishi Singh Bhatia** is a seasoned IT professional with a master's in information technology with over 17 years of IT experience spanning roles such as software developer, business analyst, functional lead, and architect, primarily within the healthcare insurance and pharmacy domain.

  Recognized among the top 1% mentors on Topmate and honored with the People's Choice Award. He has served as a judge for prestigious technical awards, including the Globee Awards and Claro, as well as for international conferences such as the 8th International Conference on Intelligent Computing and Communication and the 2025 International Conference on applied artificial intelligence and innovation. Additionally, he has contributed as a reviewer for multiple books.

  He has a proven track record of leading cross-functional teams across global locations, and driving large-scale transformation programs for Fortune 500 clients. Vishi has extensive experience in legacy modernization initiatives, successfully executing projects that integrate AI and cloud technologies. He is currently working at Tata Consultancy Services Limited and is part of the healthcare unit, where he is helping Fortune 500 companies undergo their digital transformation.

  Certifications and affiliations: IEEE Senior Member, ACM, AAAI, judge in various hackathons and technical awards, PMP Certified and PMP Mentor, CSM, Azure Certified, ITIL, ISTQB, AHIP (Parts A & B).

- **Gowtham** is a seasoned AI professional with over 14 years of industry experience, transitioning from a data scientist to a lead AI engineer. He brings deep expertise across diverse domains, including healthcare, fintech, life sciences, and the automobile industry.

  He has architected and delivered a wide range of data science solutions — from health insurance claim processing and claim amount forecasting to privacy-focused applications in fintech, leveraging computer vision and NLP for KYC document verification. In life sciences, he has designed and deployed generative AI solutions using LangChain, LangGraph, MCP, and OpenAI models — integrated with modern vector databases such as Qdrant and Weaviate — to assist in selecting the right candidates for clinical trials and driving advanced research insights.

  Currently, he is working in the automobile sector as a lead AI engineer, where he applies GenAI, MCP, and vector database–driven solutions to transform warranty-related claims processing and optimize operational efficiency. His work spans developing and deploying innovative AI systems powered by Python and Rust that leverage **retrieval-augmented generation** (**RAG**) and multi-agent workflows to address complex real-world problems at scale.

  He is passionate about applying AI responsibly and at scale, helping organizations reimagine processes, unlock business value, and maximize their technological investments.

Acknowledgements

First and foremost, we are profoundly grateful to our families for their unwavering support, encouragement, and understanding throughout this journey. Their love and motivation have been a constant source of strength and inspiration.

We extend our heartfelt thanks to BPB Publications for their expert guidance and continued support in transforming this manuscript into a published work. Their professionalism and assistance were invaluable in navigating the various stages of the publication process.

We also wish to acknowledge the contributions of the technical reviewers and editors whose thoughtful feedback and insights played a crucial role in refining and enhancing the quality of this book.

Finally, we are thankful to our readers for their interest and trust in our work. Your encouragement is deeply valued and continues to inspire us.

To all who have contributed in any way to the completion of this book—your efforts and support are sincerely appreciated.

Preface

Writing efficient and structured code has emerged as an essential skill across different domains such as engineering, scientific computing, data analysis, data science, machine learning, mathematical modeling, and research fields. Among several tools available, MATLAB and Python are noted as the two widely adopted programming languages, each with its own strengths and applications.

This book is created to develop a common understanding of these two languages among readers. It provides a learning guide as well as a practical reference for the students, educators, researchers, data analysts, data scientists, and professionals who wish to gain proficiency in these two languages and to enhance their coding skills as well.

Initiating with the basic concepts, this book covers a wide range of topics, including variables, data types, control structures, functions, data handling, plotting, and advanced topics such as signal and image processing. Each chapter provides the key functionalities in MATLAB and Python with comparative examples. This side-by-side comparison will help readers identify the syntax differences, conceptual similarities, and distinct strengths. Real-world applications and practice exercises are incorporated to reinforce the learning process and to bridge the gap between theory and practice.

Whether you are a beginner who is exploring the programming scope in scientific domains or an experienced professional, this book is designed to enhance your coding and problem-solving skills. By the end of this journey, readers will gain the ability to make suitable choices in tool selection and apply their coding skills in different domains, including data analysis, data science, scientific computing, etc.

Chapter 1: Introduction to MATLAB and Python- This chapter provides a basic introduction to the languages MATLAB and Python. By the end of this chapter, you will have an understanding of the key differences and strengths of MATLAB and Python. Moreover, you will learn how to set up and navigate the respective programming environments. Apart from this, you will gain familiarity with fundamental operations, data structures, and functions in both languages.

Chapter 2: MATLAB and Python Variables and Data Types- A thorough explanation of the variables and data types in MATLAB and Python, is given this chapter, respectively. This chapter covers a number of topics, including how to define variables in MATLAB, how to use arrays and matrices in MATLAB, and how to use strings and structures in MATLAB. Several subjects are introduced in relation to Python, such as declaring variables, data types, and strings, lists, tuples, dictionaries, and sets. Additionally, a comparison of Python and MATLAB is given through a number of examples. You may practice the topics they have learnt by completing the assignment at the end of the chapter.

Chapter 3: Basic Operations in MATLAB and Python Languages- We will examine the basic functions and operations of two potent programming languages—Python and MATLAB—in this chapter. Engineers and scientists like MATLAB because of its well-known prowess in numerical computations, especially in arithmetic and matrix operations. We will examine how well it can execute logical operations, sophisticated matrix manipulations, and fundamental arithmetic operations—all of which are critical for data analysis and algorithm creation. However, Python, which is well-known for its ease of use and adaptability, has a wide range of features, such as list operations, string manipulation, arithmetic operations, and several built-in functions that make it appropriate for data processing and general-purpose programming. By comprehending these fundamental ideas in Python and MATLAB, readers will acquire a deep knowledge of the concept.

Chapter 4: Control Flow and Structures in MATLAB and Python- The fundamental ideas of control flow and structures in MATLAB and Python, two popular programming languages in data analysis, engineering, and scientific computing, will be covered in this chapter. Readers who understand control flow will be able to write more effective programs that can repeat tasks and make judgments under certain circumstances.

This chapter is organized into three primary sections, the first two of which concentrate on the Python and MATLAB languages, respectively, while the third portion deals with popular instances of each. You will study loops and conditional expressions in the MATLAB portion. You may examine related ideas in the Python portion, but with some syntactic variations. Python implements decision-making logic via conditional expressions.

Chapter 5: Functions and Scripts in MATLAB and Python- You will learn how to create and utilize functions and scripts in MATLAB and Python, two robust programming languages that are frequently used in data analysis and scientific computing, in this chapter. You will discover how to use the function keyword in MATLAB to define reusable functions in distinct files, enabling modular and well-structured code. MATLAB scripts, which are collections of commands that are executed one after the other and are perfect for automating repetitive activities, are also covered in this chapter. You will also find anonymous functions made with the @ symbol, which offers a rapid method of defining basic, one-line functions without requiring a separate file. The def keyword will be used to define functions in the Python section, allowing for organized and reusable code blocks. Additionally, you will study lambda functions, which are concise and anonymous.

Chapter 6: Data Handling in MATLAB- In order to assist users in managing data for analysis, calculation, and visualization, this chapter provides a thorough overview of MATLAB's data handling features. Text files, spreadsheets, and binary files are just a few of the many sources of data that may be read and written using the powerful array of tools that MATLAB offers. Additionally, it allows import/export procedures for picture and audio files as well as formats like .csv, .xls, and .mat. Anyone working with experimental measurements, simulation findings, or system inputs and outputs has to understand these procedures. Beginning with basic I/O methods like fopen, fprintf, and fread, this chapter progresses to high-level functions like readtable and writetable that are utilized for tabular data. Additionally, it offers information on best practices, data cleaning methods, and real-world use cases.

Chapter 7: Data Handling in MATLAB and Python- You will study fundamental Python file handling strategies in this chapter, with an emphasis on using built-in functions like open, read, write, and close to carry out file operations. Data persistence and manipulation are made possible by these functions, which make it possible to read from and write to files efficiently. This chapter also explores working with common data types, including CSV and JSON. The CSV module in Python will teach readers how to read from and write to CSV files, which are frequently used to store tabular data. To manage JSON files, a popular format for data transmission, the json module will be presented. You will have the ability to handle files and process data in a variety of formats at the end of this chapter.

Chapter 8: Plotting and Visualization in MATLAB- This chapter aims to give readers the hands-on skills they need to design and modify different kinds of plots in MATLAB for practical uses. You may efficiently visualize data by mastering fundamental 2D charting functions like plot, bar, and scatter. In order to improve plot clarity and presentation, this chapter discusses customization approaches, such as adding titles, labels, and legends, and changing line styles. Additionally, you will learn how to display complex data in three dimensions by utilizing functions like plot3, surf, and mesh in 3D plotting. The chapter concludes by introducing specialized plots like polar plots, heatmaps, and histograms, which expand your capacity to evaluate and present data in a variety of fields, including science, engineering, finance, and data analytics.

Chapter 9: Plotting and Visualization in Python- In this chapter, MATLAB's plotting features are systematically compared with a thorough tutorial on data visualization approaches in Python. Beginning with an overview of Python's visualization ecosystem and its major libraries—Matplotlib for simple plotting, Seaborn for statistical graphics, and Plotly for interactive visualizations—the content is organized to guide users from fundamental ideas to sophisticated applications. After that, this chapter moves on to more fundamental charting methods, showing how to make and modify a variety of chart types, such as line plots, bar charts, scatter plots, and histograms. Advanced modification options to improve plot clarity and visual appeal are covered in detail in a separate section. A significant part of this chapter is the comparison analysis, in which we will compare how Python and MATLAB implement visualization tasks side by side.

Chapter 10: Working with Data in MATLAB and Python- With an emphasis on practical applications, this chapter aims to give readers the fundamental knowledge and abilities they need to handle, analyze, and preprocess data in MATLAB and Python. Readers will discover how to effectively index, slice, and reshape datasets by investigating data manipulation techniques. These abilities are essential for jobs like processing financial records or cleaning sensor data in engineering. Additionally, statistical functions are covered in the chapter, allowing users to calculate metrics like mean, standard deviation, and correlation—all of which are essential in domains like market trend analysis and biological research (e.g., evaluating data from clinical trials).

Chapter 11: Signal and Image Processing in MATLAB and Python- In this chapter, methods for image and signal processing in MATLAB and Python environments are examined. MATLAB and Python are important in domains like computer vision, audio analysis, and telecommunications because they offer powerful tools and frameworks for processing signals and images. A comparative analysis of MATLAB and Python programs with examples, MATLAB-based material with examples, and Python-based content with examples comprise this chapter.

Chapter 12: Case Studies in MATLAB and Python- This chapter aims to present practical, real-world case studies that demonstrate how MATLAB and Python can be applied to solve complex problems across various domains, including engineering, finance, signal processing, and data science. Through interactive, hands-on examples, readers will learn to perform numerical computations, data analysis, and visualization in both programming environments. The chapter is designed to build problem-solving capabilities by walking through industry-relevant scenarios such as signal filtering, image processing, financial forecasting, and statistical evaluation. By engaging with these examples, readers will enhance their ability to convert theoretical knowledge into working code, streamline workflows, and make informed decisions when choosing between MATLAB and Python for specific applications. The chapter also highlights best practices in algorithm design, debugging strategies, and performance assessment, empowering readers with the skills needed to effectively address real-world challenges.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/4a1c95

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Practical-MATLAB-and-Python.
In case there's an update to the code, it will be updated on the existing GitHub repository.
We have code bundles from our rich catalogue of books and videos available at
https://github.com/bpbpublications. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at: **errata@bpbonline.com**

Your support, suggestions and feedback are highly appreciated by the BPB Publications' Family.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks. You can check our social media handles below:


Instagram


Facebook


Linkedin


YouTube

Get in touch with us at: **business@bpbonline.com** for more details.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

https://discord.bpbonline.com

Table of Contents

1. Introduction to MATLAB and Python....1
Introduction....1
Structure....1
Objectives....1
1.1 MATLAB....2
1.1.1 MATLAB environment....2
1.1.2 Basic syntax and operations....3
Arithmetic operations....5
Functions and scripts....5
1.2 Python....7
1.2.1 Setting up the environment....8
1.2.2 Basic syntax and operations....10
Variables and data types....11
Arithmetic operations....11
Lists, tuples, and dictionaries....11
Functions....12
NumPy for numerical computing....12
1.3 Comparison between MATLAB and Python....12
Conclusion....13
Exercises....13

2. MATLAB and Python Variables and Data Types....15
Introduction....15
Structure....15
Objectives....15
2.1 MATLAB....15
2.1.1 Defining variables in MATLAB....16
2.1.2 Creating variables....16
2.1.3 Displaying variables....16
2.1.4 Data types in MATLAB....17
2.1.5 Arrays and matrices....17
Matrix operations....18
2.1.6 Strings....18
String operations....19
2.1.7 Cell arrays....20
2.1.8 Structures....20
2.2 Python....22
2.2.1 Defining variables in Python....22
2.2.2 Data types in Python....23

2.2.3 Integers *23*
2.2.4 Arithmetic operations *23*
2.2.5 Strings *24*
Common string methods *25*
Escape characters *25*
2.2.6 Lists *25*
2.2.7 Use cases and importance in data handling *26*
Creating lists *26*
Accessing list elements *27*
Modifying lists *28*
List operations *30*
Types of lists (2D lists) *32*
2.2.8 Matrix representation and basic operations *33*
2.2.9 Tuples *33*
Creating tuples *34*
Accessing tuple elements *35*
Tuple operations *35*
Tuple methods and built-in functions *37*
2.2.10 Dictionaries *38*
Creating dictionaries *38*
Accessing and modifying dictionary elements *40*
Dictionary methods and functions *41*
Working with nested dictionaries *43*
2.2.11 Sets *45*
Creating sets *45*
Accessing and modifying set elements *47*
Set operations *48*
Set methods *50*
2.3 Comparison of examples via MATLAB and Python 54
Conclusion 59
Exercises 59

3. Basic Operations in MATLAB and Python Languages 63
Introduction 63
Structure 63
Objectives 63
3.1 MATLAB 64
3.1.1 Arithmetic operations in MATLAB *64*
3.1.2 Matrix operations in MATLAB *67*
3.1.3 Logical operations in MATLAB *71*
3.2 Python 75
3.2.1 Arithmetic operations in Python *75*
3.2.2 String manipulation in Python *76*

3.2.3 List operations in Python *78*
3.2.4 Basic built-in functions in Python *86*
3.3 Comparison of examples via MATLAB and Python 87
Conclusion 95
Exercises 95

4. Control Flow and Structures in MATLAB and Python 99
Introduction 99
Structure 99
Objectives 100
4.1. Control flow in MATLAB 100
4.1.1 Conditional statements in MATLAB *100*
4.1.2 Loops in MATLAB *101*
For loop *101*
While Loop *102*
4.2. Control flow in Python 102
4.2.1 Conditional statements in Python *102*
4.2.2 Loops in Python *103*
For loop *103*
While Loop *104*
4.3. Common examples in MATLAB and Python 104
Conclusion 128
Exercises 129

5. Functions and Scripts in MATLAB and Python 133
Introduction 133
Structure 133
Objectives 134
5.1 Functions and scripts in MATLAB 134
5.1.1 Functions *134*
5.1.2 Creating functions in MATLAB *135*
5.1.3 Function with multiple outputs *137*
5.1.4 Inline functions *141*
5.1.5 Short note on scripts in MATLAB *141*
Referencing a script inside another script *142*
5.2 Functions and scripts in Python 142
5.2.1 Understanding functions *143*
5.2.2 Functions with return values *144*
5.2.3 Scope of variables in functions *145*
Understanding variable scope in Python *145*
Python using scope *145*
Python managing scope *146*
5.2.4 Recursive functions *146*

Recursion using the call stack LIFO *147*
5.2.5 Lambda functions and anonymous functions *148*
5.2.6 Writing Python scripts *150*
5.3 Comparative study in MATLAB and Python 150
Conclusion 156
Exercises 156
MATLAB *156*
Python *157*
Common practice questions in MATLAB and Python *158*

6. Data Handling in MATLAB **159**
Introduction 159
Structure 159
Objectives 159
6.1 Introduction to data handling in MATLAB 160
6.2 Reading from and writing to files 160
6.2.1 Basic file operations *160*
6.2.2 Working with binary files *162*
6.3 Importing and exporting data 163
6.3.1 Using readtable and writetable *164*
6.3.2 Working with .mat files *165*
6.3.3 Importing Excel files *166*
6.3.4 Importing text and delimited files *167*
6.4 Handling different data formats 168
6.4.1 Supported formats *168*
6.4.2 Using the file import tool *169*
6.4.3 Data cleaning after import *169*
Conclusion 171
Exercises 171

7. Data Handling in MATLAB and Python **175**
Introduction 175
Structure 175
Objectives 175
7.1 File handling and data formats in Python 176
7.1.1 File handling in Python *176*
7.1.2 Working with CSV files *178*
Using pandas for CSV handling *179*
7.1.3 Working with JSON files *180*
7.1.4 Working with other data formats *180*
7.1.5 Practical examples and use cases *182*
7.2 Comparative study of MATLAB and Python via examples 184
Conclusion 193

Exercises....193
Basic file operations....193
Working with different file modes....193
Reading and writing CSV files....194
Reading and writing JSON files....194
Handling binary files....194
Advanced file handling operations....195

8. Plotting and Visualization in MATLAB....197
Introduction....197
Structure....197
Objectives....197
8.1 plot function as foundation of MATLAB visualization....198
8.2 Bar function and visualizing categorical data....201
8.3 Exploring variable relationship through the scatter function....203
8.4 Customization of plots in MATLAB....205
8.4.1 Using titles to give context to plot....205
8.4.2 Using labels to identify axes....207
8.4.3 Using legends for multiple data series....208
8.4.4 Using line styles to enhance plot readability....209
8.4.5 A customized plot....211
8.4.6 Additional customization tips....212
8.5 Introduction to 3D plotting....213
8.6 Specialized plots in MATLAB....219
8.6.1 Using histograms to visualize data distributions....219
8.6.2 Using heatmaps to visualize matrix data....221
8.6.3 Using polar plots to visualize angular data....221
8.6.4 Error bar plots and representing variability....222
8.6.5 Stacked bar plots and comparing grouped data....223
8.6.6 Pie charts and proportional data visualization....223
8.6.7 Stem plots and visualizing discrete data....224
8.6.8 Contour plots and level curves....225
8.6.9 Box plots and statistical distribution....226
8.6.10 Logarithmic plots and visualizing exponential data....226
8.6.11 Quiver plots and visualizing vector fields....227
8.6.12 Waterfall plots and sequential surface representation....228
8.6.13 Filled contour plots and enhanced contours....228
8.6.14 Surface plot with contours....229
8.6.15 Stair plot in MATLAB....230
Conclusion....232
Exercises....232

9. Plotting and Visualization in Python **233**
Introduction 233
Structure 233
Objectives 234
9.1 Data visualization and libraries in Python 234
9.1.1 Importance of data visualization *234*
9.1.2 Libraries for data visualization in Python *234*
9.1.3 Matplotlib as the foundational library *234*
9.1.4 Seaborn and statistical data visualization *236*
9.1.5 Plotly and interactive visualizations *238*
9.2 Basic plotting in Python with Matplotlib 240
9.2.1 Introduction to Matplotlib *240*
9.3 Customizing plots in Python 243
9.3.1 Specialized plots in Python *249*
9.3.2 Introduction to specialized plots *249*
9.3.2.1 Histograms for visualizing distributions *250*
9.3.2.2 Heatmaps for visualizing matrix relationships *251*
9.3.2.3 Polar plots using visualizing angular data *254*
9.4 Comparison of examples via MATLAB and Python 255
Conclusion 269
Exercises 269

10. Working with Data in MATLAB and Python **271**
Introduction 271
Structure 271
Objectives 272
10.1 MATLAB-based concepts 272
10.1.1 Data manipulation in MATLAB *272*
10.1.2 Statistical functions in MATLAB *276*
10.1.3 Tables in MATLAB *278*
10.2 Python-based concepts 280
10.2.1 Data manipulation with Pandas and NumPy *281*
10.2.2 Statistical analysis in Python *284*
10.3 Comparative study via MATLAB and Python codes 287
10.3.1 Data manipulation *287*
10.3.2 Statistical functions *292*
10.3.3 Working with tables/DataFrames *295*
10.3.4 Advanced topics *297*
10.3.5 Visualization *300*
10.3.6 Miscellaneous *303*
10.3.7 Additional activities *303*
Conclusion 311
Exercises 311

MATLAB *311*
Python *313*

11. Signal and Image Processing in MATLAB and Python **315**
Introduction 315
Structure 315
Objectives 315
11.1 MATLAB-based content 316
11.1.1 Signal processing in MATLAB *316*
11.1.2 Image processing in MATLAB *317*
11.1.3 Advanced applications *319*
11.2 Python-based content 321
11.2.1 Signal processing in Python *321*
11.2.2 Image processing in Python *322*
11.2.3 Advanced applications *325*
11.3 Comparative study of MATLAB and Python codes 326
11.3.1 Signal processing examples *326*
11.3.2 Image processing examples *337*
11.3.3 Advanced applications *342*
Conclusion 344
Exercises 344

12. Case Studies in MATLAB and Python **347**
Introduction 347
Structure 347
Objectives 347
Chapter-wise exercises 347
Conclusion 359

Index **361-365**

Chapter 1
Introduction to MATLAB and Python

Introduction

In today's data-driven world, computational tools play a crucial role in scientific research, engineering, and data analysis. Two of the most widely used programming environments for numerical computing and algorithm development are **Matrix Laboratory** (**MATLAB**) and Python. While MATLAB has long been a primary tool in engineering and academia due to its powerful matrix operations and specialized toolboxes, Python has emerged as a versatile, open-source alternative with extensive libraries for scientific computing, machine learning, and automation. This chapter provides a comprehensive introduction to both MATLAB and Python, covering their core features, environments, and basic syntax.

Structure

This chapter contains the following topics:

- 1.1.MATLAB
- 1.2 Python
- 1.3 Comparison between MATLAB and Python

Objectives

By the end of this chapter, you will understand the key differences and strengths of MATLAB and Python and learn how to set up and navigate their respective programming environments.

You will also gain familiarity with fundamental operations, data structures, and functions in both languages and be prepared to apply these tools in mathematical modeling, data analysis, and algorithm development.

Whether you are an engineer, scientist, or programmer, mastering these languages will enhance your ability to solve complex computational problems efficiently.

1.1 MATLAB

Matrix Laboratory (**MATLAB**) is a high-performance numerical computing environment developed by MathWorks. It provides an interactive platform for algorithm development, data visualization, data analysis, and numerical computation. MATLAB is widely used in academia and industries such as engineering, physics, finance, and bioinformatics due to its powerful toolboxes and ease of use.

Some key features of MATLAB are:

- **Matrix-based computing**: Optimized for vector and matrix operations.
- **Rich library of functions**: Built-in mathematical, statistical, and engineering functions.
- **Toolboxes**: Specialized add-ons for signal processing, control systems, deep learning, and more.
- **Interactive graphics**: High-quality 2D/3D plotting and visualization tools.
- **Integration capabilities**: Supports interfacing with C/C++, Java, Python, and Fortran.

1.1.1 MATLAB environment

When you launch MATLAB, you interact with the following key components:

- **Command Window**: The Command Window in MATLAB is the primary interface where users can execute commands, perform calculations, and interact with the MATLAB environment in real time. It functions like an interactive shell, allowing users to enter expressions and immediately see the output. This is particularly useful for quick computations, debugging, and testing small code snippets without creating a script or function file. The Command Window also displays error messages, warnings, and outputs from scripts or functions. It supports command history, so previous commands can be accessed and reused easily. Overall, the Command Window is an essential component of MATLAB's workflow, enabling rapid experimentation and immediate feedback during numerical computations and programming.
    - It is used to enter commands and execute scripts.
    - Example: Typing 5 + 3 and pressing *Enter* displays ans = 8.
- **Workspace**: The Workspace in MATLAB is a dynamic area that displays all the variables currently in memory during a MATLAB session. It provides a convenient way to view, inspect, and manage variables, including their names, sizes, types, and values. Users can interact with the Workspace through the graphical interface or programmatically using commands like who, whos, and clear. This feature is especially useful for monitoring data during computation, debugging, and understanding how variables change over time. The Workspace complements the Command Window by allowing users to keep track of their data and results visually, making it an integral part of MATLAB's environment for efficient data handling and analysis.
    - It lists all variables currently stored in memory.
    - It shows variable names, values, and data types.
- **Current Folder**: The Current Folder panel in MATLAB displays the contents of the directory (folder) that MATLAB is currently accessing. It allows users to easily navigate the file system, open files, run scripts, and manage data files directly within the MATLAB environment. The Current Folder is important because MATLAB only has direct access to files located in this directory or on its path. Users can change the current folder using the navigation bar or commands like cd. Having quick access to project files, scripts, functions, and data sets makes the Current Folder panel a vital part of the MATLAB workflow, enhancing productivity and file organization.

  - It displays files and scripts in the working directory.
  - MATLAB executes files from this location.

- **Editor**: The Editor in MATLAB is a built-in text editor designed specifically for writing, editing, and debugging scripts, functions, and other code files. It offers features such as syntax highlighting, automatic indentation, code folding, and error checking, which help streamline the coding process. The Editor also provides tools for setting breakpoints, running sections of code, and stepping through code during debugging. Unlike the Command Window, which is used for executing individual commands interactively, the Editor is ideal for writing longer and more structured programs that can be saved and reused. It supports multiple tabs and integration with version control systems, making it a powerful tool for developing and maintaining complex MATLAB applications.
  - It is used to write, debug, and save MATLAB scripts (`.m` files).
  - It supports syntax highlighting and automatic indentation.
- **Toolboxes**: Toolboxes in MATLAB are specialized collections of functions, classes, and Simulink blocks that extend MATLAB's core capabilities to specific application areas. Each toolbox is designed to support tasks in a particular domain, such as signal processing, image processing, machine learning, control systems, optimization, and more. These toolboxes are developed and maintained by MathWorks and provide professionally developed algorithms, ready-to-use functions, and extensive documentation and examples. Toolboxes make it easier for users to perform complex operations without having to build everything from scratch. Since they are modular, users can install only the toolboxes relevant to their work, making MATLAB a flexible and scalable environment for both academic and industrial applications.
  - Extend MATLAB's functionality (e.g., Image Processing Toolbox, Simulink).

    Different types of toolboxes can be explored at the following link: **https://www.mathworks.com/products.html**

1.1.2 Basic syntax and operations

Variables and data types: In MATLAB, variables are used to store data values, and they are created automatically when a value is assigned using the equal sign (`=`). MATLAB is designed for matrix and numerical computation, so all variables are, by default, stored as matrices or arrays, even if they contain a single number. MATLAB is dynamically typed, meaning that you do not need to declare a variable's type before using it. Common data types in MATLAB include numeric types (`double`, `single`, `int8`, `int16`, etc.), character arrays and strings (`char`, `string`), logicals (`true`, `false`), cell arrays, structures (`struct`), and tables. MATLAB's powerful handling of arrays and data types allows users to perform complex mathematical and data manipulation tasks with simple and concise syntax. Understanding variables and data types is essential for writing efficient and error-free MATLAB programs. MATLAB is dynamically typed (no explicit declaration needed).

Some common data types are as follows:

- **Numeric**: `double`, `int8`, `single`
- **Logical**: `true/false`
- **Character**: `'Hello'`
- **Cell arrays**: `{1, 'text', [3 4]}`

Example 1.1: [Numeric (Default double)]:

```
a = 5.25;          % 'a' is stored as a double by default
class(a)           % Returns: 'double'
```

Example 1.2: [Integer data types (int8, int16, etc.)]

```
b = int8(127);      % Assign an 8-bit signed integer
class(b)            % Returns: 'int8'
```

Example 1.3: [Single precision floating point]:

```
c = single(3.14);   % Converts to single precision
class(c)            % Returns: 'single'
```

Example 1.4: [Logical values (true/false)]:

```
d = true;           % Logical variable
e = (5 > 10);       % Evaluates to false
class(e)            % Returns: 'logical'
```

Example 1.5: [Character array (String using single quotes)]:

```
greeting = 'Hello, MATLAB';   % Character array
class(greeting)               % Returns: 'char'
```

Example 1.6: String data type (introduced in newer MATLAB versions):

```
name = "Quantum";             % String scalar
class(name)                   % Returns: 'string'
```

Example 1.7: Cell array with mixed data types:

```
myCell = {1, 'MATLAB', [2 3 4]};    % Cell array containing number, string, and array
class(myCell)                       % Returns: 'cell'
```

Example 1.8: Accessing elements of a cell array:

```
element = myCell{2};         % Accesses 'MATLAB'
class(element)               % Returns: 'char'
```

Example 1.9: Creating logical array from condition:

```
A = [1, 2, 3, 4, 5];
logicalA = A > 3;            % Returns [0 0 0 1 1]
class(logicalA)              % Returns: 'logical'
```

Example 1.10: Combining types in a structure:

```
student.name = 'Alice';
student.age = int8(22);
student.passed = true;
student.grades = [85, 90, 78];
% Use 'class' function on a field
class(student.age)           % Returns: 'int8'
```

Example 1.11: MATLAB's array structures for row vectors, column vectors, and matrices:

```
a = 10;          % Scalar
b = [1 2 3];     % Row vector
c = [1; 2; 3];   % Column vector
d = rand(3,3);   % 3x3 random matrix
```

Arithmetic operations

In MATLAB, arithmetic operations are fundamental and are performed using standard operators such as addition (+), subtraction (-), multiplication (*), division (/ for right division and \ for left division), element-wise multiplication (.*), element-wise division (./, .\), and exponentiation (^ for matrix power and .^ for element-wise power), which are detailed in the following table. MATLAB is inherently designed for matrix and vector computations, so these operations can be applied to scalars, vectors, matrices, and higher-dimensional arrays. Element-wise operators are particularly important when performing operations on corresponding elements of arrays. MATLAB follows standard operator precedence rules, and parentheses can be used to change the order of evaluation. Mastery of arithmetic operations in MATLAB is essential for performing calculations, implementing algorithms, and developing simulations in engineering, science, and applied mathematics.

Operation	Syntax	Example
Addition	`+`	`5 + 3 → 8`
Subtraction	`-`	`7 - 2 → 5`
Multiplication	`*`	`4 * 6 → 24`
Division	`/`	`10 / 2 → 5`
Exponentiation	`^`	`2^3 → 8`

***Table 1.1**: Basic arithmetic operations in MATLAB*

Vectorized addition:

```
A = [1, 2, 3];
B = [4, 5, 6];
C = A + B;        % Element-wise addition → [5, 7, 9]
```

Matrix multiplication:

```
M1 = [1 2; 3 4];
M2 = [5; 6];
result = M1 * M2;  % Matrix multiplication → [17; 39]
```

Scalar division:

```
totalMarks = 450;
subjects = 5;
average = totalMarks / subjects;   % → 90
```

Element-wise operations (for arrays):

```
A = [1 2; 3 4];
B = [5 6; 7 8];
C = A .* B;  % Element-wise multiplication → [5 12; 21 32]
```

Functions and scripts

In MATLAB, both scripts and functions are types of program files that contain sequences of MATLAB commands. While they may appear similar at first instance, they serve distinct purposes and differ in terms of how they handle input/output, variable scope, and reusability.

Functions

A function in MATLAB is a more structured and modular type of program file that allows for input arguments and output results. Functions operate in their own local workspace, which means variables inside the function do not interfere with variables in the base workspace unless explicitly passed in or out. This makes functions ideal for performing specific tasks repeatedly or with varying data.

Characteristics of functions:

- It is defined using the function keyword.
- It accepts input arguments and returns output values.
- It has their own isolated variable workspace.
- It supports modular, reusable programming practices.

Syntax:

```
function [output1, output2, ...] = functionName(input1, input2, ...)
    % Function body
end
```

Let us look at the following types of functions:

- **Built-in functions**: MATLAB provides a wide range of built-in functions that simplify mathematical, statistical, and engineering computations. These functions are optimized, pre-defined operations that can be applied directly to scalars, vectors, matrices, and higher-dimensional arrays. Whether you are performing basic arithmetic or complex scientific analysis, built-in functions in MATLAB help streamline your code and enhance performance. Functions like **`sin()`**, **`cos()`**, and **`sqrt()`** are used for mathematical operations, while **`mean()`** and **`max()`** are commonly used for data analysis:
  - **sin() – Sine of an angle (in radians):**

    ```
    angle = pi / 6;              % 30 degrees in radians
    sineValue = sin(angle);      % Returns 0.5
    ```

  - **cos() – Cosine of an angle (in radians):**

    ```
    theta = pi / 3;              % 60 degrees in radians
    cosValue = cos(theta);       % Returns 0.5
    ```

  - **sqrt() – Square root calculation:**

    ```
    distance = 25;
    rootValue = sqrt(distance);  % Returns 5
    ```

  - **mean() – Average of an array**:

    ```
    scores = [88, 92, 79, 85, 90];
    averageScore = mean(scores);  % Returns 86.8
    ```

  - **max() – Maximum element in an array:**

    ```
    temperatures = [22.5, 27.8, 25.1, 30.0, 28.4];
    maxTemp = max(temperatures);  % Returns 30.0
    ```

- **User-defined functions**:

  ```
  function y = square(x)
      y = x^2;
  end
  ```

Scripts

A sequence of commands saved in **`.m`** files. A script is a simple program file that contains a sequence of MATLAB commands. It operates in the base workspace, meaning any variables created or modified in the script are accessible after the script finishes running. Scripts are typically used for performing a series of calculations, visualizations, or simulations where data is already available in the workspace or defined within the script itself.

Characteristics of scripts:

- It does not accept input or return output explicitly.
- It shares the same workspace as the base MATLAB environment.
- It is ideal for quick analysis or when working with existing data.

Key differences between scripts and functions in MATLAB:

Feature	Script	Function
Input arguments	No	Yes
Output arguments	No	Yes
Workspace	Base workspace	Local workspace
Reusability	Limited	High
Best used for	Simple tasks, quick computations	Modular, repeatable tasks

***Table 1.2**: Main differences between scripts and functions in MATLAB*

A function or a script in MATLAB can be used in the following ways:

- Use scripts when working interactively or when prototyping with known data.
- Use functions when you need to encapsulate a task that might be reused, or when you want to pass specific inputs and get well-defined outputs.

Example 1.12: Steps to run a script in MATLAB:

1. **Script name**: **`circle_area.m`**

   ```
   radius = 5;
   area = pi * radius^2;
   disp(['The area of the circle is: ', num2str(area)]);
   ```

2. **How to run**:
   a. Save the file as circle_area.m in your MATLAB working directory.
   b. In the Command Window, type:
      `circle_area`
3. **Output**:

   `The area of the circle is: 78.54`

1.2 Python

Python is a general-purpose, interpreted, high-level programming language known for its simplicity and readability. It is widely used in:

- Scientific computing (NumPy, SciPy)
- Data analysis (Pandas)
- Machine learning (Scikit-learn, TensorFlow)
- Web development (Django, Flask)