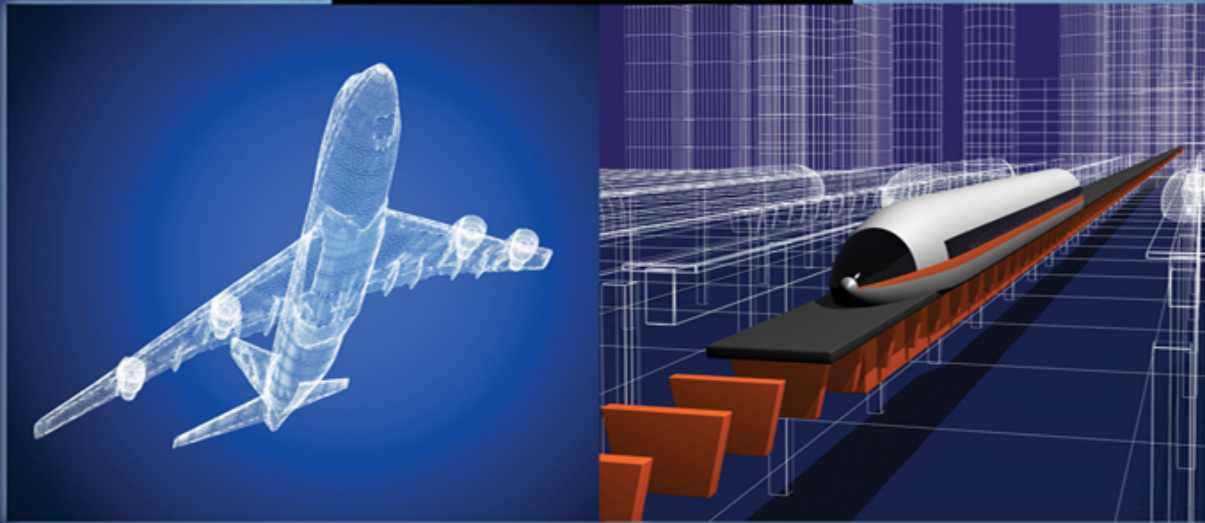
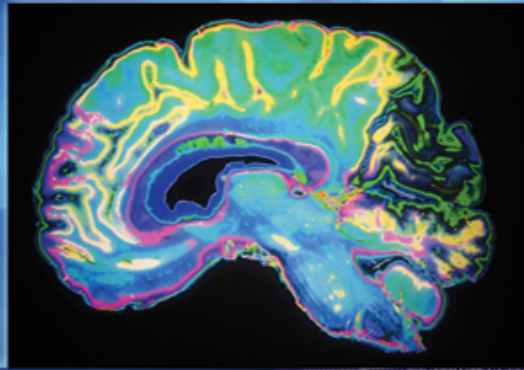


MATLAB[®]

PROGRAMMING

with Applications for Engineers



STEPHEN J. CHAPMAN

MATLAB[®] **Programming** **with Applications** **for Engineers**

MATLAB[®]

Programming

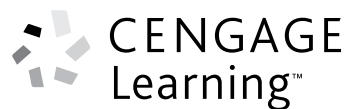
with Applications

for Engineers

First Edition

Stephen J. Chapman

BAE Systems Australia



Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

**MATLAB® Programming with Applications
for Engineers****Stephen J. Chapman**Publisher, Global Engineering: Christopher
M. Shortt

Senior Acquisitions Editor: Randall Adams

Senior Developmental Editor: Hilda Gowans

Editorial Assistant: Tanya Altieri

Team Assistant: Carly Rizzo

Marketing Manager: Lauren Betsos

Media Editor: Chris Valentine

Content Project Manager: D. Jean Buttrum

Production Service: RPK Editorial Services, Inc.

Copyeditor: Shelly Gerger-Knechtl

Proofreader: Harlan James

Indexer: Shelly Gerger-Knechtl

Compositor: Integra Software Solutions

Senior Art Director: Michelle Kunkler

Internal Designer: Carmela Periera

Cover Designer: Andrew Adams/4065042
Canada Inc.

Cover Image: © ivn3da/Shutterstock;

© Martin Trajkovski/Shutterstock;

© Daisy Daisy/Shutterstock;

© PhotoStocker/Shutterstock

Rights Acquisitions Specialist: John Hill

Text and Image Permissions Researcher:
Kristiina Paul

Senior First Print Buyer: Doug Wilke

© 2013 Cengage Learning

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transmitted, stored, or used in any form or by any means graphic, electronic, or mechanical, including but not limited to photocopying, recording, scanning, digitizing, taping, web distribution, information networks, or information storage and retrieval systems, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the publisher.

For product information and technology assistance, contact us at
Cengage Learning Customer & Sales Support, 1-800-354-9706.

For permission to use material from this text or product,
submit all requests online at www.cengage.com/permissions.
Further permissions questions can be emailed to
permissionrequest@cengage.com.

Library of Congress Control Number: 2011934123

ISBN-13: 978-0-495-66807-7

ISBN-10: 0-495-66807-9

Cengage Learning200 First Stamford Place, Suite 400
Stamford, CT 06902
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at: international.cengage.com/region.

Cengage Learning products are represented in Canada by
Nelson Education Ltd.

For your course and learning solutions, visit www.cengage.com/engineering.

Purchase any of our products at your local college store or at our preferred
online store www.cengagebrain.com.

MATLAB is a registered trademark of The MathWorks, Inc., 3 Apple Hill
Drive, Natick, MA.

Printed in the United States of America

1 2 3 4 5 6 7 13 12 11

This is an electronic version of the print textbook. Due to electronic rights restrictions, some third party content may be suppressed. Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. The publisher reserves the right to remove content from this title at any time if subsequent rights restrictions require it. For valuable information on pricing, previous editions, changes to current editions, and alternate formats, please visit www.cengage.com/highered to search by ISBN#, author, title, or keyword for materials in your areas of interest.

This book is dedicated with love to my daughter Sarah Rivkah Chapman. As a student at Swinburne University in Melbourne, she may actually wind up using it!

About the Author

Stephen J. Chapman received a BS in Electrical Engineering from Louisiana State University (1975), an MSE in Electrical Engineering from the University of Central Florida (1979), and pursued further graduate studies at Rice University.

From 1975 to 1980, he served as an officer in the U.S. Navy, assigned to teach Electrical Engineering at the U.S. Naval Nuclear Power School in Orlando, Florida. From 1980 to 1982, he was affiliated with the University of Houston, where he ran the power systems program in the College of Technology.

From 1982 to 1988 and from 1991 to 1995, he served as a Member of the Technical Staff of the Massachusetts Institute of Technology's Lincoln Laboratory, both at the main facility in Lexington, Massachusetts, and at the field site on Kwajalein Atoll in the Republic of the Marshall Islands. While there, he did research in radar signal processing systems. He ultimately became the leader of four large operational range instrumentation radars at the Kwajalein field site (TRADEX, ALTAIR, ALCOR, and MMW).

From 1988 to 1991, Mr. Chapman was a research engineer in Shell Development Company in Houston, Texas, where he did seismic signal processing research. He was also affiliated with the University of Houston, where he continued to teach on a part-time basis.

Mr. Chapman is currently Manager of Systems Modeling and Operational Analysis for BAE Systems Australia, in Melbourne, Australia. He is the leader of a team that has developed a model of how naval ships defend themselves against antiship missile attacks. This model contains more than 400,000 lines of MATLAB™ code written over more than a decade, so he has extensive practical experience applying MATLAB to real-world problems.

Mr. Chapman is a Senior Member of the Institute of Electrical and Electronic Engineers (and several of its component societies). He is also a member of the Association for Computing Machinery and the Institution of Engineers (Australia).

Contents

Chapter I Introduction to MATLAB

I

- I.1 The Advantages of MATLAB 2**
- I.2 Disadvantages of MATLAB 4**
- I.3 The MATLAB Environment 4**
 - 1.3.1 The MATLAB Desktop 4
 - 1.3.2 The Command Window 6
 - 1.3.3 The Command History Window 7
 - 1.3.4 The Start Button 7
 - 1.3.5 The Edit/Debug Window 9
 - 1.3.6 Figure Windows 9
 - 1.3.7 Docking and Undocking Windows 11
 - 1.3.8 The MATLAB Workspace 11
 - 1.3.9 The Workspace Browser 12
 - 1.3.10 Getting Help 13
 - 1.3.11 A Few Important Commands 15
 - 1.3.12 The MATLAB Search Path 17
- I.4 Using MATLAB as a Calculator 19**
- I.5 Summary 21**
 - 1.5.1 MATLAB Summary 22
- I.6 Exercises 22**

vii

- 2.1 Variables and Arrays 25**
- 2.2 Creating and Initializing Variables in MATLAB 29**
 - 2.2.1 Initializing Variables in Assignment Statements 29
 - 2.2.2 Initializing with Shortcut Expressions 32
 - 2.2.3 Initializing with Built-in Functions 33
 - 2.2.4 Initializing Variables with Keyboard Input 33
- 2.3 Multidimensional Arrays 35**
 - 2.3.1 Storing Multidimensional Arrays in Memory 37
 - 2.3.2 Accessing Multidimensional Arrays with One Dimension 37
- 2.4 Subarrays 39**
 - 2.4.1 The `end` Function 39
 - 2.4.2 Using Subarrays on the Left-hand Side of an Assignment Statement 40
 - 2.4.3 Assigning a Scalar to a Subarray 41
- 2.5 Special Values 42**
- 2.6 Displaying Output Data 44**
 - 2.6.1 Changing the Default Format 44
 - 2.6.2 The `disp` function 46
 - 2.6.3 Formatted output with the `fprintf` function 46
- 2.7 Data Files 48**
- 2.8 Scalar and Array Operations 50**
 - 2.8.1 Scalar Operations 51
 - 2.8.2 Array and Matrix Operations 51
- 2.9 Hierarchy of Operations 54**
- 2.10 Built-in MATLAB Functions 57**
 - 2.10.1 Optional Results 58
 - 2.10.2 Using MATLAB Functions with Array Inputs 58
 - 2.10.3 Common MATLAB Functions 58
- 2.11 Introduction to Plotting 60**
 - 2.11.1 Using Simple *xy* Plots 61
 - 2.11.2 Printing a Plot 62
 - 2.11.3 Exporting a Plot as a Graphical Image 62
 - 2.11.4 Saving a Plot in a Figure File 63
 - 2.11.5 Multiple Plots 63
 - 2.11.6 Line Color, Line Style, Marker Style, and Legends 64
- 2.12 Examples 68**
- 2.13 MATLAB Applications: Vector Mathematics 74**
 - 2.13.1 Vector Addition and Subtraction 76
 - 2.13.2 Vector Multiplication 77
- 2.14 MATLAB Applications: Matrix Operations and Simultaneous Equations 81**
 - 2.14.1 The Matrix Inverse 82

- 2.15 Debugging MATLAB Programs 84**
- 2.16 Summary 86**
 - 2.16.1 Summary of Good Programming Practice 86
 - 2.16.2 MATLAB Summary 87
- 2.17 Exercises 90**

Chapter 3 Two-Dimensional Plots 103

- 3.1 Additional Plotting Features for Two-Dimensional Plots 103**
 - 3.1.1 Logarithmic Scales 104
 - 3.1.2 Controlling *x*- and *y*-axis Plotting Limits 107
 - 3.1.3 Plotting Multiple Plots on the Same Axes 110
 - 3.1.4 Creating Multiple Figures 111
 - 3.1.5 Subplots 111
 - 3.1.6 Controlling the Spacing Between Points on a Plot 114
 - 3.1.7 Enhanced Control of Plotted Lines 117
 - 3.1.8 Enhanced Control of Text Strings 118
- 3.2 Polar Plots 121**
- 3.3 Annotating and Saving Plots 123**
- 3.4 Additional Types of Two-Dimensional Plots 126**
- 3.5 Using the `plot` function with Two-Dimensional Arrays 131**
- 3.6 Summary 133**
 - 3.6.1 Summary of Good Programming Practice 134
 - 3.6.2 MATLAB Summary 134
- 3.7 Exercises 135**

Chapter 4 Branching Statements and Program Design 139

- 4.1 Introduction to Top-Down Design Techniques 140**
- 4.2 Use of Pseudocode 143**
- 4.3 Relational and Logic Operators 144**
 - 4.3.1 Relational Operators 144
 - 4.3.2 A Caution About The `==` And `~=` Operators 146
 - 4.3.3 Logic Operators 147
 - 4.3.4 Logical Functions 151
- 4.4 Branches 153**
 - 4.4.1 The `if` Construct 154
 - 4.4.2 Examples Using `if` Constructs 156
 - 4.4.3 Notes Concerning the Use of `if` Constructs 162
 - 4.4.4 The `switch` Construct 164
 - 4.4.5 The `try/catch` Construct 166
- 4.5 More on Debugging MATLAB Programs 173**
- 4.6 MATLAB Applications: Roots of Polynomials 178**

- 4.7 Summary 181**
 - 4.7.1 Summary of Good Programming Practice 181
 - 4.7.2 MATLAB Summary 182
- 4.8 Exercises 182**

Chapter 5 Loops and Vectorization 189

- 5.1 The while Loop 189**
- 5.2 The for Loop 195**
 - 5.2.1 Details of Operation 202
 - 5.2.2 Vectorization: A Faster Alternative to Loops 204
 - 5.2.3 The MATLAB Just-In-Time (JIT) Compiler 205
 - 5.2.4 The break and continue Statements 208
 - 5.2.5 Nesting Loops 210
- 5.3 Logical Arrays and Vectorization 212**
 - 5.3.1 Creating the Equivalent of if/else Constructs with Logical Arrays 213
- 5.4 The MATLAB Profiler 215**
- 5.5 Additional Examples 217**
- 5.6 The textread Function 232**
- 5.7 MATLAB Applications: Statistical Functions 234**
- 5.8 MATLAB Applications: Curve Fitting and Interpolation 237**
 - 5.8.1 General Least-Squares Fits 237
 - 5.8.2 Cubic Spline Interpolation 244
 - 5.8.3 Interactive Curve-Fitting Tools 250
- 5.9 Summary 253**
 - 5.9.1 Summary of Good Programming Practice 254
 - 5.9.2 MATLAB Summary 254
- 5.10 Exercises 255**

Chapter 6 Basic User-Defined Functions 267

- 6.1 Introduction to MATLAB Functions 269**
- 6.2 Variable Passing in MATLAB: The Pass-By-Value Scheme 274**
- 6.3 Optional Arguments 285**
- 6.4 Sharing Data Using Global Memory 290**
- 6.5 Preserving Data Between Calls to a Function 298**
- 6.6 MATLAB Applications: Sorting Functions 303**
- 6.7 MATLAB Applications: Random Number Functions 305**
- 6.8 Summary 306**
 - 6.8.1 Summary of Good Programming Practice 306
 - 6.8.2 MATLAB Summary 306
- 6.9 Exercises 307**

- 9.1.4 Extending Cell Arrays 380
- 9.1.5 Deleting Cells in Arrays 382
- 9.1.6 Using Data in Cell Arrays 383
- 9.1.7 Cell Arrays of Strings 383
- 9.1.8 The Significance of Cell Arrays 384
- 9.1.9 Summary of `cell` Functions 388
- 9.2 Structure Arrays 388**
 - 9.2.1 Creating Structure Arrays 390
 - 9.2.2 Adding Fields to Structures 392
 - 9.2.3 Removing Fields from Structures 392
 - 9.2.4 Using Data in Structure Arrays 393
 - 9.2.5 The `getfield` and `setfield` Functions 394
 - 9.2.6 Dynamic Field Names 395
 - 9.2.7 Using the `size` Function with Structure Arrays 397
 - 9.2.8 Nesting Structure Arrays 397
 - 9.2.9 Summary of `structure` Functions 398
- 9.3 Importing Data into MATLAB 403**
- 9.4 Summary 405**
 - 9.4.1 Summary of Good Programming Practice 406
 - 9.4.2 MATLAB Summary 406
- 9.5 Exercises 406**

Chapter 10 Handle Graphics and Animation

411

- 10.1 Handle Graphics 411**
 - 10.1.1 The MATLAB Graphics System 411
 - 10.1.2 Object Handles 413
 - 10.1.3 Examining and Changing Object Properties 413
 - 10.1.4 Using `set` to List Possible Property Values 420
 - 10.1.5 Finding Objects 422
 - 10.1.6 Selecting Objects with the Mouse 424
- 10.2 Position and Units 426**
 - 10.2.1 Positions of `figure` Objects 427
 - 10.2.2 Positions of `axes` Objects 428
 - 10.2.3 Positions of `text` Objects 428
- 10.3 Printer Positions 431**
- 10.4 Default and Factory Properties 431**
- 10.5 Graphics Object Properties 434**
- 10.6 Animations and Movies 434**
 - 10.6.1 Erasing and Redrawing 434
 - 10.6.2 Creating a Movie 439
- 10.7 Summary 441**
 - 10.7.1 Summary of Good Programming Practice 441
 - 10.7.2 MATLAB Summary 442
- 10.8 Exercises 442**

Chapter 11 More MATLAB Applications 447

- 11.1 Solving Systems of Simultaneous Equations 447**
 - 11.1.1 Possible Solutions of Simultaneous Equations 449
 - 11.1.2 Determining the Existence and Uniqueness of Solutions 451
 - 11.1.3 Well-Conditioned Versus Ill-Conditioned Systems of Equations 452
 - 11.1.4 Solving Systems of Equations with Unique Solutions 454
 - 11.1.5 Solving Systems of Equations with an Infinite Number of Solutions 456
 - 11.1.6 Solving Overdetermined Systems of Equations 460
- 11.2 Differences and Numerical Differentiation 463**
- 11.3 Numerical Integration—Finding the Area Under a Curve 466**
- 11.4 Differential Equations 472**
 - 11.4.1 Deriving Differential Equations for a System 473
 - 11.4.2 Solving Ordinary Differential Equations in MATLAB 476
 - 11.4.3 Applying ode45 to Solve for the Voltage in a Circuit 480
 - 11.4.4 Solving Systems of Differential Equations 482
 - 11.4.5 Solving Higher Order Differential Equations 486
 - 11.4.6 Stiff Differential Equations 489
- 11.5 Summary 490**
 - 11.5.1 Summary of Good Programming Practice 491
 - 11.5.2 MATLAB Summary 492
- 11.6 Exercises 492**

Appendix A ASCII Character Set 499

Appendix B Additional MATLAB Input/Output Functions 501

- B.1 MATLAB File Processing 501**
- B.2 File Opening and Closing 503**
 - B.2.1 The fopen Function 503
 - B.2.2 The fclose Function 505
- B.3 Binary I/O Functions 506**
 - B.3.1 The fwrite Function 506
 - B.3.2 The fread Function 507
- B.4 Formatted I/O Functions 510**
 - B.4.1 The fprintf Function 510
 - B.4.2 Understanding Format Conversion Specifiers 512
 - B.4.3 The fscanf Function 514
 - B.4.4 The fgetl Function 516
 - B.4.5 The fgets Function 516
- B.5 The textscan Function 516**

Appendix C Working with Character Strings 519

- C.1 String Functions 519**
 - C.1.1 String Conversion Functions 520
 - C.1.2 Creating Two-Dimensional Character Arrays 520
 - C.1.3 Concatenating Strings 521
 - C.1.4 Comparing Strings 521
 - C.1.5 Searching and Replacing Characters within a String 525
 - C.1.6 Uppercase and Lowercase Conversion 526
 - C.1.7 Trimming Whitespace from Strings 527
 - C.1.8 Numeric-to-String Conversions 527
 - C.1.9 String-to-Numeric Conversions 529
 - C.1.10 Summary 530
- C.2 Summary 536**
 - C.2.1 Summary of Good Programming Practice 536
 - C.2.2 MATLAB Summary 537
- C.3 Exercises 538**

Appendix D Answers to Quizzes 539

- Index 555**

Preface

MATLAB[®] (short for MATrix LABoratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It started life as a program designed to perform matrix mathematics, but over the years it has grown into a flexible computing system capable of solving essentially any technical problem.

The MATLAB program implements the MATLAB language and provides a very extensive library of pre-defined functions to make technical programming tasks easier and more efficient. This extremely wide variety of functions makes it much easier to solve technical problems in MATLAB than in other languages such as Java, Fortran, or C++. This book introduces the MATLAB language, and shows how to use it to solve typical technical problems.

This book seeks to simultaneously teach MATLAB as a technical programming language and also to introduce the student to many of the practical functions that make solving problems in MATLAB so much easier than in other languages. The book provides a complete introduction to the fundamentals of good procedural programming, developing good design habits that will serve a student well in any other language that he or she may pick up later. There is a very strong emphasis on proper program design and structure. A standard program design process is introduced at the beginning of Chapter 4 and then followed regularly throughout the remainder of the text.

In addition, the book uses the programming topics and examples as a jumping off point for exploring the rich set of highly optimized application functions that are built directly into MATLAB. For example, in Chapter 4 we present a programming example that finds the roots of a quadratic equation. This serves as a jumping off point for exploring the MATLAB function `roots`, which can efficiently find the

roots of polynomials of any order. In Chapter 5, we present a programming example that calculates the mean and standard deviation of a data set. This serves as a jumping off point for exploring the MATLAB functions `mean`, `median`, and `std`. There is also a programming example showing how to do a least-squares fit to a straight-line. This serves as a jumping off point for exploring MATLAB curve fitting functions such as `polyfit`, `polyval`, `spline`, and `ppval`. There are similar ties to MATLAB applications in many other chapters as well. In all cases, there are end of chapter exercises to reinforce the applications lessons learned in that chapter.

In addition, Chapter 11 is devoted totally to practical MATLAB applications, including solving systems of simultaneous equations, numerical differentiation, numerical integration (quadrature), and solving ordinary differential equations.

This book makes no pretense at being a complete description of all of MATLAB's hundreds of functions. Instead, it teaches the student how to use MATLAB as a language to solve problems, and how to locate any desired function with MATLAB's extensive on-line help facilities. It highlights quite a few of the key engineering applications, but there are far more good ones built into the language than can be covered in any course of reasonable length. With the skills developed here, students will be able to continue discovering features on their own.

The Advantages of MATLAB for Problem Solving

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

1. **Ease of Use.** MATLAB is very easy to use. The program can be used as a scratch pad to evaluate expressions typed at the command line, or it can be used to execute large pre-written programs. Programs may be easily written and modified with the built-in integrated development environment, and debugged with the MATLAB debugger. Because the language is so easy to use, it is ideal for educational use, and for the rapid prototyping of new programs.

Many program development tools are provided to make the program easy to use. They include an integrated editor / debugger, on-line documentation and manuals, a workspace browser, and extensive demos.

2. **Platform Independence.** MATLAB is supported on many different computer systems, providing a large measure of platform independence. At the time of this writing, the language is supported on Windows XP/Vista/7, Linux, Unix, and the Macintosh. Programs written on any platform will run on all of the other platforms, and data files written on any platform may be read transparently on any other platform. As a result, programs written in MATLAB can migrate to new platforms when the needs of the user change.

3. **Pre-defined Functions.** MATLAB comes complete with an extensive library of pre-defined functions that provide tested and pre-packaged solutions to many basic technical tasks. For example, suppose that you are writing a program that must calculate the statistics associated with an input data set. In most languages, you would need to write your own sub-routines or functions to implement calculations such as the arithmetic mean, standard deviation, median, etc. These and hundreds of other functions are built right into the MATLAB language, making your job much easier.

The built-in functions can solve an astonishing range of problems, such as solving systems of simultaneous equations, sorting, plotting, finding roots of equations, numerical integration, curve fitting, solving ordinary and partial differential equations, and much, much more.

In addition to the large library of functions built into the basic MATLAB language, there are many special-purpose toolboxes available to help solve complex problems in specific areas. For example, a user can buy standard toolboxes to solve problems in Signal Processing, Control Systems, Communications, Image Processing, and Neural Networks, among many others.

4. **Device-Independent Plotting.** Unlike other computer languages, MATLAB has many integral plotting and imaging commands. The plots and images can be displayed on any graphical output device supported by the computer on which MATLAB is running. This capability makes MATLAB an outstanding tool for visualizing technical data. Plotting is introduced in Chapter 2, and covered extensively in Chapters 3 and 8. Advanced features such as animations and movies are covered in Chapter 10.
5. **Graphical User Interface.** MATLAB includes tools that allow a program to interactively construct a Graphical User Interface (GUI) for his or her program. With this capability, the programmer can design sophisticated data analysis programs that can be operated by relatively-inexperienced users.

Features of this Book

Many features of this book are designed to emphasize the proper way to write reliable MATLAB programs. These features should serve a student well as he or she is first learning MATLAB, and should also be useful to the practitioner on the job. They include:

1. **Emphasis on Top-Down Design Methodology.** The book introduces a top-down design methodology in Chapter 4, and then uses it consistently throughout the rest of the book. This methodology encourages a student

to think about the proper design of a program *before* beginning to code. It emphasizes the importance of clearly defining the problem to be solved and the required inputs and outputs before any other work is begun. Once the problem is properly defined, it teaches the student to employ stepwise refinement to break the task down into successively smaller sub-tasks, and to implement the subtasks as separate subroutines or functions. Finally, it teaches the importance of testing at all stages of the process, both unit testing of the component routines and exhaustive testing of the final product.

The formal design process taught by the book may be summarized as follows:


1. *Clearly state the problem that you are trying to solve.*
 2. *Define the inputs required by the program and the outputs to be produced by the program.*
 3. *Describe the algorithm that you intend to implement in the program.* This step involves top-down design and stepwise decomposition, using pseudocode or flow charts.
 4. *Turn the algorithm into MATLAB statements.*
 5. *Test the MATLAB program.* This step includes unit testing of specific functions, and also exhaustive testing of the final program with many different data sets.
2. **Emphasis on Functions.** The book emphasizes the use of functions to logically decompose tasks into smaller subtasks. It teaches the advantages of functions for data hiding. It also emphasizes the importance of unit testing functions before they are combined into the final program. In addition, the book teaches about the common mistakes made with functions, and how to avoid them.
 3. **Emphasis on MATLAB Tools.** The book teaches the proper use of MATLAB's built-in tools to make programming and debugging easier. The tools covered include the Editor / Debugger, Workspace Browser, Help Browser, and GUI design tools.
 4. **Emphasis on MATLAB applications.** The book teaches how to harness the power of MATLAB's rich set of functions to solve a wide variety of practical engineering problems. This introduction to MATLAB functions is spread throughout the book, and is generally tied to the topics and examples being discussed in a particular chapter.
 5. **Good Programming Practice Boxes.** These boxes highlight good programming practices when they are introduced for the convenience of the student. In addition, the good programming practices introduced in a chapter are summarized at the end of the chapter. An example Good Programming Practice Box is shown below.

*** Good Programming Practice:**

Always indent the body of an `if` construct by 2 or more spaces to improve the readability of the code.

6. Programming Pitfalls Boxes

These boxes highlight common errors so that they can be avoided. An example Programming Pitfalls Box is shown below.

 Programming Pitfalls:

Make sure that your variable names are unique in the first 63 characters. Otherwise, MATLAB will not be able to tell the difference between them.

Pedagogical Features

This book includes several features designed to aid student comprehension. A total of 13 quizzes appear scattered throughout the chapters, with answers to all questions included in Appendix D. These quizzes can serve as a useful self-test of comprehension. In addition, there are approximately 215 end-of-chapter exercises. Answers to all exercises are included in the Instructor's Manual. Good programming practices are highlighted in all chapters with special Good Programming Practice boxes, and common errors are highlighted in Programming Pitfalls boxes. End of chapter materials include Summaries of Good Programming Practice and Summaries of MATLAB Commands and Functions.

The book is accompanied by an Instructor's Manual, containing the solutions to all end-of-chapter exercises. The IM, PowerPoint slides of all figures and tables in the book and the source code for all examples in the book is available from the book's Web site, and the source code for all solutions in the Instructor's Manual is available separately to instructors.

To access additional course materials [including CourseMate], please visit www.cengagebrain.com. At the cengagebrain.com home page, search for the ISBN of your title (from the back cover of your book) using the search box at the top of the page. This will take you to the product page where these resources can be found.

A Thank You to the Reviewers

I would like to offer a special thank you to the book's reviewers. Their invaluable suggestions have made this a significantly better book, and they certainly deserve thanks for the time they devoted to reviewing drafts of the text. The reviewers who were willing to be named are:

Steven A. Peralta, University of New Mexico
Jeffrey Ringenberg, University of Michigan
Lizzie Santiago, West Virginia University
John R. White, University of Massachusetts, Lowell

A Final Note to the User

No matter how hard I try to proofread a document like this book, it is inevitable that some typographical errors will slip through and appear in print. If you should spot any such errors, please drop me a note via the publisher, and I will do my best to get them eliminated from subsequent printings and editions. Thank you very much for your help in this matter.

I will maintain a complete list of errata and corrections at the book's World Wide Web site, which is <http://www.cengage.com/engineering>. Please check that site for any updates and/or corrections.

STEPHEN J. CHAPMAN
Melbourne, Australia



Introduction to MATLAB

MATLAB (short for MATrix LABoratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It started life as a program designed to perform matrix mathematics, but over the years, it has grown into a flexible computing system capable of solving essentially any technical problem.

The MATLAB program implements the MATLAB programming language and provides an extensive library of predefined functions to make technical programming tasks easier and more efficient. This book introduces the MATLAB language as it is implemented in MATLAB Version 7.9 and shows how to use it to solve typical technical problems.

MATLAB is a huge program, with an incredibly rich variety of functions. Even the basic version of MATLAB without any toolkits is much richer than other technical programming languages. There are more than 1000 functions in the basic MATLAB product alone, and the toolkits extend this capability with many more functions in various specialties. Furthermore, these functions often solve very complex problems (solving differential equations, inverting matrices, and so forth) in a *single step*, saving large amounts of time. Doing the same thing in another computer language usually involves writing complex programs yourself or buying a third-party software package (such as IMSL or the NAG software libraries) that contains the functions.

The built-in MATLAB functions are almost always better than anything that an individual engineer could write on his or her own, because many people have worked on them and they have been tested against many different data sets. These functions are also robust, producing sensible results for wide ranges of input data and gracefully handling error conditions.

- [*read Paying the Tab: The Costs and Benefits of Alcohol Control*](#)
- [**read online Iliada pdf, azw \(kindle\), epub, doc, mobi**](#)
- [Marmol Radziner + Associates: Between Architecture and Construction pdf, azw \(kindle\), epub, doc, mobi](#)
- [Mobile First pdf, azw \(kindle\), epub](#)

- <http://metromekanik.com/ebooks/The-Ides--Caesar-s-Murder-and-the-War-for-Rome.pdf>
- <http://nautickim.es/books/Ili--ada.pdf>
- <http://www.uverp.it/library/Marmol-Radziner---Associates--Between-Architecture-and-Construction.pdf>
- <http://www.rap-wallpapers.com/?library/Sloth--A-Dictionary-for-the-Lazy.pdf>