



MATLAB[®] Programming for Engineers

Third Edition

Stephen J. Chapman

13



Bookware Companion Series

For sale in India, Pakistan, Bangladesh,
Nepal and Sri Lanka only.
Circulation of this edition
outside of these
countries is
STRICTLY
PROHIBITED

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

3rd Edition

Stephen J. Chapman

SYSTEMS Australia

THOMSON
—★—
LEARNING

Australia • Canada • Mexico • Singapore • Spain • United Kingdom • United States



MATLAB[®] Programming for Engineers, Third Edition
by Stephen J. Chapman

**Associate Vice-President
and Editorial Director:**
Evelyn Veitch

Publisher:
Bill Stenquist

Sales and Marketing Manager:
John More

Developmental Editor:
Kamilah Reid Burrell

Production Service:
RPK Editorial Services

Copy Editor/Proofing:
Harlan James

Indexer:
Shelly Gerger-Knechtl

Production Manager:
Renate McCloy

Creative Director:
Angela Cluer

Cover Design:
Peter Papayanakis

Compositor:
PV & M Publishing Solutions

COPYRIGHT © 2004 by Thomson Learning
A division of Thomson Learning, Inc.,
Thomson Learning[™] is a trademark used herein under license.

First Reprint 2004 by Thomson Asia Pte Ltd., Singapore

Second Reprint 2005

ISBN: 981-254-893-9

Printed and bound at Eastern Press (Bangalore) Pvt. Ltd., 100% EOU,
110B, Bommasandra Industrial Area, Hosur Road, Anekal Taluk,
Bangalore 560 099 (INDIA).

2 3 4 5

08 07 06 05

ALL RIGHTS RESERVED.No part of this work covered by the copyright herein may be reproduced, transcribed or used in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, Web distribution or information storage and retrieval systems—without the written permission of the publisher.

Library of Congress Control
Number: 2004095122

ISBN: 0-534-42417-1

For sale in the Indian Subcontinent only.

*This book is dedicated to my wife Rosa,
the mother of our eight children,
after almost 30 wonderful years together.*

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 predefined 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 Fortran or C. This book introduces the MATLAB language and shows how to use it to solve typical technical problems.

This book teaches MATLAB as a technical programming language, showing students how to write clean, efficient, and documented programs. It 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, and how to locate any desired function with MATLAB's extensive on-line help facilities.

The first six chapters of the text are designed to serve as the text for an "Introduction to Programming/Problem Solving" course for freshman engineering students. This material should fit comfortably into a 9-week, 3-hour course. The remaining chapters cover advanced topics such as I/O and graphical user interfaces. These chapters may be covered in a longer course or used as a reference by engineering students or practicing engineers who use MATLAB as a part of their coursework or employment.

Changes in the Third Edition

The third edition of this book is specifically devoted to MATLAB 7. MATLAB 7 contains many language and tool changes, and this book had to be revised extensively for the new version. Some of the major changes include:

- Case-sensitive function and directory names on all platforms.
- Function handles.
- The use of an `end` statement at the end of functions.
- Nested functions.
- Math operations with single and integer data types.
- Major revision of the GUI code, including the addition of panels, button groups, and toolbars. The code auto-generated by `guide` has been totally changed. Frames have been deprecated.
- Major revisions to programming tools, such as the addition of conditional breakpoints and the `mlint` tool to check for poor programming practices within an M-file.

The changes in MATLAB are so major that many MATLAB 7 programs (especially GUIs) may not be backward compatible with earlier versions of MATLAB. If you are working with an earlier version of MATLAB, do not use this book—get the Second Edition instead. Both the Second Edition and the Third Edition of this book will remain available until the transition to MATLAB 7 is essentially complete.

The Advantages of MATLAB for Technical Programming

MATLAB has many advantages compared to conventional computer languages for technical problem solving. These include:

1. Ease of Use

MATLAB is an interpreted language, like many versions of Basic. Like Basic, it 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 prewritten 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 NT/2000/XP, 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 predefined functions that provide tested and prepackaged 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 subroutines or functions to implement calculations such as the arithmetic mean, standard deviation, median, among others. These and hundreds of other functions are built right into the MATLAB language, making your job much easier.

In addition to the large library of 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.

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.

6. MATLAB Compiler

MATLAB's flexibility and platform independence are achieved by compiling MATLAB programs into a device-independent p-code and then interpreting the p-code instructions at run-time. This approach is similar to that used by Microsoft's Visual Basic language. Unfortunately, the resulting programs can sometimes execute slowly because the MATLAB code is interpreted rather than compiled. We will point out features that tend to slow program execution when we encounter them.

A separate MATLAB compiler is available. This compiler can compile a MATLAB program into a true executable that runs faster than the interpreted code. It is a great way to convert a prototype MATLAB program into an executable suitable for sale and distribution to 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 3 and 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 subtasks 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.*
3. *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. 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 two or more spaces to improve the readability of the code.

5. 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.

6. Emphasis on Data Structures

Chapter 7 contains a detailed discussion of MATLAB data structures, including sparse arrays, cell arrays, and structure arrays. The proper use of these data structures is illustrated in the chapters on handle graphics and graphical user interfaces.

Pedagogical Features

The first six chapters of this book are specifically designed to be used in a freshman “Introduction to Program/Problem Solving” course. It should be possible to cover this material comfortably in a 9-week, 3 hour-per-week course. If there is insufficient time to cover all of the material in a particular engineering program, Chapter 6 may be deleted, the remaining material will still teach the fundamentals of programming and using MATLAB to solve problems. This feature should appeal to harassed engineering educators trying to cram ever more material into a finite curriculum.

The remaining chapters cover advanced material that will be useful to engineers and engineering students as they progress in their careers. This material includes advanced I/O and the design of graphical user interfaces for programs.

The book includes several features designed to aid student comprehension. A total of 15 quizzes appear scattered throughout the chapters, with answers to all questions included in Appendix B. These quizzes can serve as a useful self-test of comprehension. In addition, there are approximately 160 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 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.

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.engineering.thomsonlearning.com>. Please check that site for any updates and/or corrections.

Acknowledgments

I would like to thank Bill Stenquist and the crew at Thomson for the support they have given me in getting this book to market. It has been gratifying to see the user response to the first and second editions, which were the result of our joint efforts.

In addition, I would like to thank my wife Rosa and our children—Avi, David, Rachel, Aaron, Sarah, Naomi, Shira, and Devorah—for being such delightful people and the inspiration for my efforts.

STEPHEN J. CHAPMAN
Melbourne, Australia

Contents

Chapter 1 Introduction to MATLAB

I

- 1.1 The Advantages of MATLAB 2**
- 1.2 Disadvantages of MATLAB 3**
- 1.3 The MATLAB Environment 3**
 - The MATLAB Desktop 4
 - The Command Window 4
 - The Command History Window 6
 - The Start Button 7
 - The Edit/Debug Window 7
 - Figure Windows 8
 - Docking and Undocking Windows 10
 - The MATLAB Workspace 11
 - The Workspace Browser 12
 - Getting Help 13
 - A Few Important Commands 14
 - The MATLAB Search Path 15
- 1.4 Using MATLAB as a Scratch Pad 17**
- 1.5 Summary 18**
 - MATLAB Summary 19
- 1.6 Exercises 19**

Chapter 2 MATLAB Basics

21

- 2.1 Variables and Arrays 21**
- 2.2 Initializing Variables in MATLAB 25**
 - Initializing Variables in Assignment Statements 25
 - Initializing with Shortcut Expressions 28
 - Initializing with Built-In Functions 29
 - Initializing Variables with Keyboard Input 29
- 2.3 Multidimensional Arrays 31**
 - Storing Multidimensional Arrays in Memory 33
 - Accessing Multidimensional Arrays with One Dimension 33
- 2.4 Subarrays 35**
 - The end Function 35
 - Using Subarrays on the Left-Hand Side of an Assignment Statement 36
 - Assigning a Scalar to a Subarray 37
- 2.5 Special Values 38**
- 2.6 Displaying Output Data 40**
 - Changing the Default Format 40
 - The disp function 41
 - Formatted Output with the fprintf Function 41
- 2.7 Data Files 43**
- 2.8 Scalar and Array Operations 45**
 - Scalar Operations 46
 - Array and Matrix Operations 46
- 2.9 Hierarchy of Operations 50**
- 2.10 Built-in MATLAB Functions 53**
 - Optional Results 53
 - Using MATLAB Functions with Array Inputs 53
 - Common MATLAB Functions 54
- 2.11 Introduction to Plotting 54**
 - Using Simple xy Plots 56
 - Printing a Plot 57
 - Exporting a Plot as a Graphical Image 57
 - Multiple Plots 58
 - Line Color, Line Style, Marker Style, and Legends 60
 - Logarithmic Scales 63
- 2.12 Examples 64**
- 2.13 Debugging MATLAB Programs 71**
- 2.14 Summary 73**
 - Summary of Good Programming Practice 74
 - MATLAB Summary 74
- 2.15 Exercises 77**

Chapter 3 Branching Statements and Program Design 85

- 3.1 Introduction to Top-Down Design Techniques 85
- 3.2 Use of Pseudocode 91
- 3.3 The Logical Data Type 91
 - Relational Operators 92
 - A Caution About the == and ~= Operators 95
 - Logic Operators 96
 - Logical Functions 100
- 3.4 Branches 102
 - The if Construct 102
 - Examples Using if Constructs 104
 - Notes Concerning the Use of if Constructs 111
 - The switch Construct 113
 - The try/catch Construct 114
- 3.5 Additional Plotting Features 117
 - Controlling x- and y-axis Plotting Limits 117
 - Command/Function Duality 118
 - Plotting Multiple Plots on the Same Axes 120
 - Creating Multiple Figures 121
 - Subplots 121
 - Enhanced Control of Plotted Lines 123
 - Enhanced Control of Text Strings 123
 - Polar Plots 125
 - Annotating and Saving Plots 132
- 3.6 More on Debugging MATLAB Programs 134
- 3.7 Summary 139
 - Summary of Good Programming Practice 139
 - MATLAB Summary 140
- 3.8 Exercises 141

Chapter 4 Loops

147

- 4.1 The while Loop 147
- 4.2 The for Loop 153
 - Details of Operation 161
 - The MATLAB Just-in-Time (JIT) Compiler 163
 - The break and continue Statements 167
 - Nesting Loops 168
- 4.3 Logical Arrays and Vectorization 170
 - Creating the Equivalent of if/else Constructs with Logical Arrays 173
- 4.4 Additional Examples 175

- 4.5 **Summary 190**
 - Summary of Good Programming Practice 190
 - MATLAB Summary 191
- 4.6 **Exercises 191**

Chapter 5 User-Defined Functions 199

- 5.1 **Introduction to MATLAB Functions 201**
- 5.2 **Variable Passing in MATLAB: The Pass-By-Value Scheme 207**
- 5.3 **Optional Arguments 218**
- 5.4 **Sharing Data Using Global Memory 223**
- 5.5 **Preserving Data Between Calls to a Function 231**
- 5.6 **Function Functions 236**
- 5.7 **Subfunctions, Private Functions, and Nested Functions 240**
 - Subfunctions 241
 - Private Functions 242
 - Nested Functions 243
 - Order of Function Evaluation 245
- 5.8 **Summary 246**
 - Summary of Good Programming Practice 247
 - MATLAB Summary 247
- 5.9 **Exercises 248**

Chapter 6 Additional Data Types and Plot Types 261

- 6.1 **Complex Data 261**
 - Complex Variables 264
 - Using Complex Numbers with Relational Operators 264
 - Complex Functions 265
 - Plotting Complex Data 269
- 6.2 **String Functions 272**
 - String Conversion Functions 273
 - Creating Two-Dimensional Character Arrays 273
 - Concatenating Strings 274
 - Comparing Strings 275
 - Searching and Replacing Characters Within a String 278
 - Uppercase and Lowercase Conversion 280
 - Trimming Whitespace from Strings 280
 - Numeric-to-String Conversions 281
 - String-to-Numeric Conversions 282
 - Summary 283
- 6.3 **Multidimensional Arrays 290**
- 6.4 **Additional Data Types 292**

- The single Data Type 292
- Integer Data Types 293
- Limitations of the single and Integer Data Types 295
- 6.5 Additional Two-Dimensional Plots 295**
 - Additional Types of Two-Dimensional Plots 295
 - Plotting Functions 300
 - Histograms 302
- 6.6 Three-Dimensional Plots 303**
 - Three-Dimensional Line Plots 303
 - Three-Dimensional Surface, Mesh, and Contour Plots 305
- 6.7 Summary 306**
 - Summary of Good Programming Practice 309
 - MATLAB Summary 309
- 6.8 Exercises 311**

Chapter 7 Advanced Features: Sparse Arrays, Cell Arrays, Structures, and Function Handles 315

- 7.1 Sparse Arrays 315**
 - The `sparse` Attribute 317
- 7.2 Cell Arrays 323**
 - Creating Cell Arrays 325
 - Using Braces `{}` as Cell Constructors 326
 - Viewing the Contents of Cell Arrays 326
 - Extending Cell Arrays 327
 - Deleting Cells in Arrays 330
 - Using Data in Cell Arrays 330
 - Cell Arrays of Strings 331
 - The Significance of Cell Arrays 332
 - Summary of `cell` Functions 336
- 7.3 Structure Arrays 336**
 - Creating Structure Arrays 336
 - Adding Fields to Structures 339
 - Removing Fields from Structures 340
 - Using Data in Structure Arrays 340
 - The `getfield` and `setfield` Functions 342
 - Dynamic Field Names 343
 - Using the `size` Function with Structure Arrays 344
 - Nesting Structure Arrays 345
 - Summary of `structure` Functions 346
- 7.4 Function Handles 346**
 - Creating and Using Function Handles 346
 - The Significance of Function Handles 348
 - Function Handles and Nested Functions 350

- 7.5 **Summary 353**
 - Summary of Good Programming Practice 355
 - MATLAB Summary 355
- 7.6 **Exercises 355**

Chapter 8 Input/Output Functions 359

- 8.1 **The `textread` Function 359**
- 8.2 **More about the `load` and `save` Commands 361**
- 8.3 **An Introduction to MATLAB File Processing 363**
- 8.4 **File Opening and Closing 365**
 - The `fopen` Function 365
 - The `fclose` Function 368
- 8.5 **Binary I/O Functions 368**
 - The `fwrite` Function 368
 - The `fread` Function 369
- 8.6 **Formatted I/O Functions 373**
 - The `fprintf` Function 373
 - Understanding Format Conversion Specifiers 374
 - How Format Strings Are Used 377
 - The `sprintf` Function 379
 - The `fscanf` Function 380
 - The `fgetl` Function 382
 - The `fgets` Function 383
- 8.7 **Comparing Formatted and Binary I/O Functions 383**
- 8.8 **File Positioning and Status Functions 388**
 - The `exist` Function 389
 - The `ferror` Function 392
 - The `feof` Function 392
 - The `ftell` Function 392
 - The `frewind` Function 392
 - The `fseek` Function 393
 - The `textscan` Function 399
 - Function `uiimport` 401
- 8.9 **Summary 401**
 - Summary of Good Programming Practice 404
 - MATLAB Summary 404
- 8.10 **Exercises 405**

Chapter 9 Handle Graphics 409

- 9.1 **The MATLAB Graphics System 409**
- 9.2 **Object Handles 411**
- 9.3 **Examining and Changing Object Properties 411**

- Changing Object Properties at Creation Time 411
- Changing Object Properties After Creation Time 412
- 9.4 Using set to List Possible Property Values 418**
- 9.5 User-Defined Data 420**
- 9.6 Finding Objects 422**
- 9.7 Selecting Objects with the Mouse 423**
- 9.8 Position and Units 426**
 - Positions of figure Objects 426
 - Positions of axes and uicontrol Objects 427
 - Positions of text Objects 428
- 9.9 Printer Positions 431**
- 9.10 Default and Factory Properties 431**
- 9.11 Graphics Object Properties 434**
- 9.12 Summary 434**
 - Summary of Good Programming Practice 434
 - MATLAB Summary 435
- 9.13 Exercises 435**

Chapter 10 Graphical User Interfaces 439

- 10.1 How a Graphical User Interface Works 439**
- 10.2 Creating and Displaying a Graphical User Interface 440**
 - A Look Under the Hood 451
 - The Structure of a Callback Subfunction 454
 - Adding Application Data to a Figure 454
 - A Few Useful Functions 456
- 10.3 Object Properties 457**
- 10.4 Graphical User Interface Components 459**
 - Static Text Fields 460
 - Edit Boxes 460
 - Pushbuttons 462
 - Toggle Buttons 463
 - Checkboxes and Radio Buttons 463
 - Popup Menus 466
 - List Boxes 466
 - Sliders 469
- 10.5 Additional Containers: Panels and Button Groups 475**
 - Panels 475
 - Button Groups 476
- 10.6 Dialog Boxes 478**
 - Error and Warning Dialog Boxes 479
 - Input Dialog Boxes 480
 - The uigetfile, uisetfile, and uigetdir Dialog Boxes 481
 - The uisetcolor and uisetfont Dialog Boxes 482
- 10.7 Menus 483**

Suppressing the Default Menu	484
Creating Your Own Menus	485
Accelerator Keys and Keyboard Mnemonics	487
Creating Context Menus	488
10.8 Tips for Creating Efficient GUIs	494
Tool Tips	494
Pcode	495
Toolbars	496
Additional Enhancements	497
10.9 Summary	502
Summary of Good Programming Practice	503
MATLAB Summary	504
10.10 Exercises	505
A ASCII Character Set	509
B Answers to Quizzes	511
Index	529