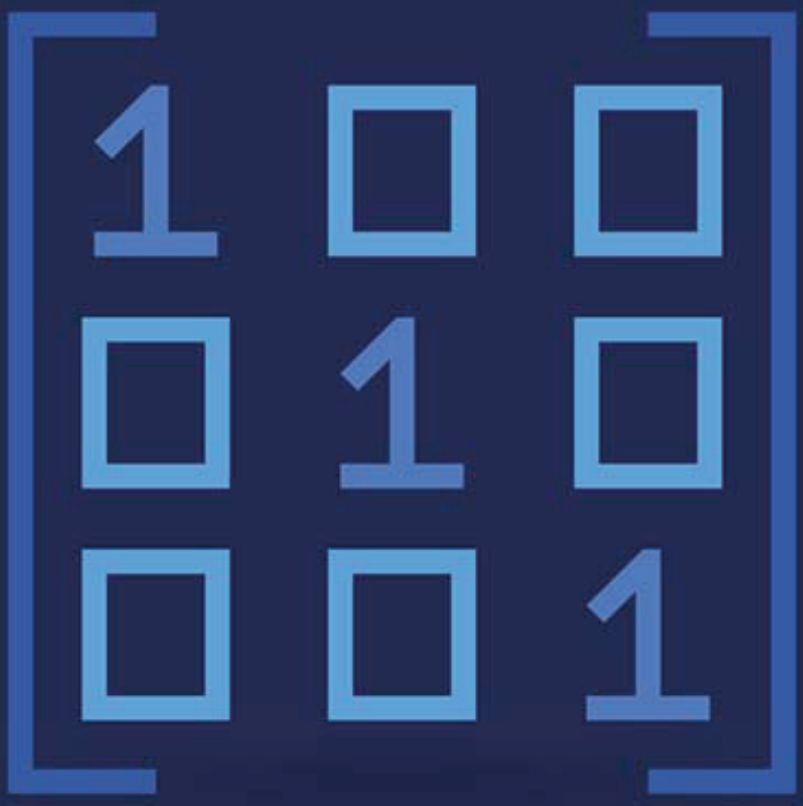
GLOBAL EDITION



MATLAB® for Engineers

FIFTH EDITION

Holly Moore

Pearson

MATLAB® for Engineers

This page intentionally left blank

MATLAB® for Engineers

Fifth Edition
Global Edition

Holly Moore
Salt Lake Community College
Salt Lake City, Utah

Pearson

330 Hudson Street, NY NY 10013

Director, Portfolio Management: Engineering, Computer Science & Global Editions: *Julian Partridge*
Specialist, Higher Ed Portfolio Management: *Holly Stark*
Portfolio Management Assistant: *Amanda Brands*
Acquisitions Editor, Global Edition: *Moasenla Jamir*
Managing Content Producer: *Scott Disanno*
Content Producer: *Carole Snyder*
Assistant Project Editor, Global Edition: *Aman Kumar*
Web Developer: *Steve Wright*
Manager, Media Production, Global Edition: *Vikram Kumar*
Rights and Permissions Manager: *Ben Ferrini*
Manufacturing Buyer, Higher Ed, Lake Side Communications Inc (LSC): *Maura Zaldivar-Garcia*
Senior Manufacturing Controller, Global Edition: *Angela Hawksbee*
Inventory Manager: *Ann Lam*
Product Marketing Manager: *Yvonne Vannatta*
Field Marketing Manager: *Demetrius Hall*
Marketing Assistant: *Jon Bryant*
Cover Designer: *Lumina Datamatics, Inc.*
Full-Service Project Management: *SPi Global*
Cover Image: *Venomous Vector / Shutterstock*

Pearson Education Limited
KAO Two
KAO Park
Harlow
CM17 9NA
United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsonglobaleditions.com

Authorized adaptation from the United States edition, entitled MATLAB® for Engineers, *5th Edition, ISBN 978-0-13-458964-0 by Holly Moore, published by Pearson Education © 2018.*

ISBN 10: 1-292-23120-3
ISBN 13: 978-1-292-23120-4

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

Typeset by SPi Global
Printed and bound by Vivar in Malaysia

Contents

ABOUT THIS BOOK 11
DEDICATION AND ACKNOWLEDGMENTS 15

1 • ABOUT MATLAB® 17

1.1 What is MATLAB®? 17
1.2 Student Edition of MATLAB® 18
1.3 How is MATLAB® Used in Industry? 19
1.4 Problem Solving in Engineering and Science 21

2 • MATLAB® ENVIRONMENT 25

2.1 Getting Started 25
2.2 MATLAB® Windows 27
2.3 Solving Problems With MATLAB® 33
2.4 Saving Your Work 57
Summary 68
MATLAB® Summary 69
Key Terms 71
Problems 71

3 • BUILT-IN MATLAB® FUNCTIONS 78

Introduction 78
3.1 Using Built-In Functions 78
3.2 Using the Help Feature 80
3.3 Elementary Math Functions 83
3.4 Trigonometric Functions 91
3.5 Data Analysis Functions 95
3.6 Random Numbers 114
3.7 Complex Numbers 119
3.8 Computational Limitations 122
3.9 Special Values and Miscellaneous Functions 124
Summary 126

MATLAB® Summary 126
Key Terms 128
Problems 128

4 • MANIPULATING MATLAB® MATRICES 135

4.1 Manipulating Matrices 135
4.2 Problems with Two Variables—Using Meshgrid 142
4.3 Special Matrices 149
Summary 155
MATLAB® Summary 155
Key Terms 156
Problems 156

5 • PLOTTING 162

Introduction 162
5.1 Two-Dimensional Plots 162
5.2 Subplots 179
5.3 Other Types of Two-Dimensional Plots 181
5.4 Three-Dimensional Plotting 198
5.5 Editing Plots From the Menu Bar 205
5.6 Creating Plots From the Workspace Window 207
5.7 Saving Your Plots 208
Summary 209
MATLAB® Summary 209
Problems 211

6 • LOGICAL FUNCTIONS AND SELECTION STRUCTURES 222

Introduction 222
6.1 Relational and Logical Operators 223
6.2 Flowcharts and Pseudocode 225
6.3 Logical Functions 227
6.4 Selection Structures 233
6.5 Debugging 250
Summary 250
MATLAB® Summary 251
Key Terms 251
Problems 252

7 • REPETITION STRUCTURES 264

Introduction 264
7.1 `For` Loops 265
7.2 `While` Loops 273
7.3 `Break` and `Continue` 281
7.4 Midpoint Break Loops 282

7.5 Nested Loops 286
7.6 Improving the Efficiency of Loops 287
Summary 290
MATLAB® Summary 291
Key Terms 291
Problems 291

8 • USER-CONTROLLED INPUT AND OUTPUT 297

Introduction 297
8.1 User-Defined Input 297
8.2 Output Options 302
8.3 Graphical Input 313
8.4 More Features Using Section Dividers 314
8.5 Reading and Writing Data from Files 316
8.6 Debugging Your Code 319
Summary 323
MATLAB® Summary 324
Key Terms 325
Problems 325

9 • USER-DEFINED FUNCTIONS 330

Introduction 330
9.1 Creating Function Files 330
9.2 Creating Your Own Toolbox of Functions 349
9.3 Anonymous Functions and Function Handles 350
9.4 Function Functions 352
9.5 Subfunctions 353
Summary 359
MATLAB® Summary 360
Key Terms 360
Problems 360

10 • MATRIX ALGEBRA 367

Introduction 367
10.1 Matrix Operations and Functions 367
10.2 Solutions of Systems of Linear Equations 387
10.3 Special Matrices 401
Summary 404
MATLAB® Summary 406
Key Terms 406
Problems 406

11 • OTHER KINDS OF ARRAYS 414

Introduction 414
11.1 Data Types 415

11.2 Numeric Data Types 415
11.3 Character and String Data 421
11.4 Symbolic Data 429
11.5 Logical Data 429
11.6 Sparse Arrays 430
11.7 Categorical Arrays 431
11.8 Time Arrays 431
11.9 Multidimensional Arrays 436
11.10 Cell Arrays 437
11.11 Structure Arrays 439
11.12 Table Arrays 446
Summary 447
MATLAB® Summary 448
Key Terms 449
Problems 449

12 • SYMBOLIC MATHEMATICS 457

Introduction 457
12.1 Symbolic Algebra 458
12.2 Solving Expressions and Equations 464
12.3 Symbolic Plotting 475
12.4 Calculus 483
12.5 Differential Equations 497
12.6 Converting Symbolic Expressions to Anonymous Functions 501
Summary 502
MATLAB® Summary 503
Problems 504

13 • NUMERICAL TECHNIQUES 513

13.1 Interpolation 513
13.2 Curve Fitting 523
13.3 Using the Interactive Fitting Tools 536
13.4 Differences and Numerical Differentiation 539
13.5 Numerical Integration 548
13.6 Solving Differential Equations Numerically 554
Summary 561
MATLAB® Summary 563
Key Terms 563
Problems 564

14 • ADVANCED GRAPHICS 573

Introduction 573
14.1 Images 573
14.2 Graphics Objects 588
14.3 Animation 594

14.4 Other Visualization Techniques 601
14.5 Introduction to Volume Visualization 603
Summary 606
MATLAB® Summary 607
Key Terms 608
Problems 608

15 • CREATING GRAPHICAL USER INTERFACES 611

Introduction 611
15.1 A Simple GUI with One User Interaction 612
15.2 A Graphical User Interface with Multiple User Interactions—`ready_aim_fire` 620
15.3 An Improved `ready_aim_fire` Program 623
15.4 A Much Better `ready_aim_fire` Program 625
15.5 Built-In GUI Templates 629
Summary 632
Key Terms 632
Problems 632

16 • SIMULINK®—A BRIEF INTRODUCTION 634

Introduction 634
16.1 Applications 634
16.2 Getting Started 635
16.3 Solving Differential Equations with Simulink® 643
Summary 649
Key Terms 650
Problems 650

APPENDIX A • SPECIAL CHARACTERS, COMMANDS, AND FUNCTIONS 654

APPENDIX B • SCALING TECHNIQUES 669

APPENDIX C • THE READY_AIM_FIRE GUI 672

APPENDIX D 677

INDEX 679

This page intentionally left blank

About This Book

This book grew out of my experience teaching MATLAB® and other computing languages to freshmen engineering students at Salt Lake Community College. I was frustrated by the lack of a text that "started at the beginning." Although there were many comprehensive reference books, they assumed a level of both mathematical and computer sophistication that my students did not possess. Also, because MATLAB® was originally adopted by practitioners in the fields of signal processing and electrical engineering, most of these texts provided examples primarily from those areas, an approach that didn't fit with a general engineering curriculum. This text starts with basic algebra and shows how MATLAB® can be used to solve engineering problems from a wide range of disciplines. The examples are drawn from concepts introduced in early chemistry and physics classes and freshman and sophomore engineering classes. A standard problem-solving methodology is used consistently.

The text assumes that the student has a basic understanding of college algebra and has been introduced to trigonometric concepts; students who are mathematically more advanced generally progress through the material more rapidly. Although the text is not intended to teach subjects such as statistics or matrix algebra, when the MATLAB® techniques related to these subjects are introduced, a brief background is included. In addition, sections describing MATLAB® techniques for solving problems by means of calculus and differential equations are introduced near the end of appropriate chapters. These sections can be assigned for additional study to students with a more advanced mathematics background, or they may be useful as reference material as students progress through an engineering curriculum.

The book is intended to be a "hands-on" manual. My students have been most successful when they read the book while sitting beside a computer and typing in the examples as they go. Numerous examples are embedded in the text, with more complicated numbered examples included in each chapter to reinforce the concepts introduced. Practice exercises are included in each chapter to give students an immediate opportunity to use their new skills.

The material is grouped into three sections. The first, ***An Introduction to Basic MATLAB® Skills***, gets the student started and contains the following chapters:

- Chapter 1 shows how MATLAB® is used in engineering and introduces a standard problem-solving methodology.
- Chapter 2 introduces the MATLAB® environment and the skills required to perform basic computations. It also introduces MATLAB program files (sometimes called M-files) , and the concept of organizing code into sections. Doing so early in the text makes it easier for students to save their work and develop a consistent programming strategy.
- Chapter 3 details the wide variety of problems that can be solved with built-in MATLAB® functions. Background material on many of the functions is provided to help the student understand how they might be used. For example, the difference between Gaussian random numbers and uniform random numbers is described, and examples of each are presented.

- Chapter 4 demonstrates the power of formulating problems by using matrices in MATLAB® and expanding on the techniques employed to define those matrices. The **meshgrid** function is introduced in this chapter and is used to solve problems with two variables. The difficult concept of meshing variables is revisited in Chapter 5 when surface plots are introduced.
- Chapter 5 describes the wide variety of both two-dimensional and three-dimensional plotting techniques available in MATLAB®. Creating plots via MATLAB® commands, either from the command window or from within a MATLAB program, is emphasized. However, the extremely valuable techniques of interactively editing plots and creating plots directly from the workspace window are also introduced.

  MATLAB® is a powerful programming language that includes the basic constructs common to most programming languages. Because it is a scripting language, creating programs and debugging them in MATLAB® is often easier than in traditional programming languages such as C++. This makes MATLAB® a valuable tool for introductory programming classes. The second section of the text, ***Programming in MATLAB®***, introduces students to programming and consists of the following chapters:
- Chapter 6 describes logical functions such as **find** and demonstrates how they vary from the **if** and **if/else** structures. The **switch/case** structure is also introduced. The use of logical functions over control structures is emphasized, partly because students (and teachers) who have previous programming experience often overlook the advantages of using MATLAB®'s built-in matrix functionality.
- Chapter 7 introduces repetition structures, including **for** loops, **while** loops, and midpoint break loops that utilize the **break** command. Numerous examples are included because students find these concepts particularly challenging.
- Chapter 8 introduces functions that interact with the program user, including user-defined input, formatted output, and graphical input techniques. The use of MATLAB®'s debugging tools is also introduced.
- Chapter 9 describes how to create and use user-defined functions. It also teaches students how to create a "toolbox" of functions to use in their own programming projects.

Chapters 1 through 9 taken together are essential for a basic understanding of MATLAB®, but the chapters in Section 3, ***Advanced MATLAB® Concepts,*** do not depend upon each other. Any or all of these chapters could be used in an introductory course or could serve as reference material for self-study. Most of the material is appropriate for freshmen. A two-credit course might include Chapters 1 through 9 plus Chapter 10, while a three-credit course might include Chapters 1 through 14, but eliminate Sections 12.4, 12.5, 13.4, 13.5, and 13.6, which describe differentiation techniques, integration techniques, and solution techniques for differential equations. Chapters 15 and 16 will be interesting to more advanced students, and might be included in a course delivered to sophomore or junior students instead of to freshmen. The skills developed in these chapters will be especially useful as students become more involved in solving engineering problems:

- Chapter 10 discusses problem solving with matrix algebra, including dot products, cross products, and the solution of linear systems of equations. Although matrix algebra is widely used in all engineering fields, it finds early application in the statics and dynamics classes taken by most engineering majors.

- Chapter 11 is an introduction to the wide variety of data types available in MATLAB®. This chapter is especially useful for electrical engineering and computer engineering students.
- Chapter 12 introduces MATLAB®'s symbolic mathematics package, built on the MuPad engine. Students will find this material especially valuable in mathematics classes. My students tell me that the package is one of the most valuable sets of techniques introduced in the course. It is something they start using immediately.
- Chapter 13 presents numerical techniques used in a wide variety of applications, especially curve fitting and statistics. Students value these techniques when they take laboratory classes such as chemistry or physics or when they take the labs associated with engineering classes such as heat transfer, fluid dynamics, or strengths of materials.
- Chapter 14 examines graphical techniques used to visualize data. These techniques are especially useful for analyzing the results of numerical analysis calculations, including results from structural analysis, fluid dynamics, and heat transfer codes.
- Chapter 15 introduces MATLAB®'s graphical user interface capability, using the GUIDE application. Creating their own graphical user interfaces gives students insight into how the graphical user interfaces they use daily on other computer platforms are created.
- Chapter 16 introduces Simulink®, which is a simulation package built on top of the MATLAB® platform. Simulink® uses a graphical user interface that allows programmers to build models of dynamic systems. It has found significant acceptance in the field of electrical engineering but has wide application across the engineering spectrum.

Appendix A lists all of the functions and special symbols (or characters) introduced in the text. Appendix B describes strategies for scaling data, so that the resulting plots are linear. Appendix C includes the complete MATLAB® code to create the `Ready_Aim_Fire` graphical user interface described in Chapter 15. Appendix D includes the Asheville, North Carolina weather data used in a number of the example problems.

The following materials are provided on the Instructor's Resource Center:

- M-files containing solutions to practice exercises (These files are also available on the student version of the website.)
- M-files containing solutions to example problems
- M-files containing solutions to homework problems
- PowerPoint slides for each chapter
- All of the figures used in the text, suitable for inclusion in your own PowerPoint presentations

Appendix E Solutions to Practice Exercises can be found at the following website:
www.pearsonglobaleditions.com/moore

WHAT'S NEW IN THIS EDITION

New versions of MATLAB® are rolled out every six months, which makes keeping any text up-to-date a challenge. Significant changes were introduced in version 2014b to

the graphics package. Another major change occurred in 2016 with the addition of major changes to the symbolic algebra functionality. Multiple new data types were introduced in both the 2016 updates. The changes in this edition reflect these software updates up through R2016b, which include the following:

- Screen shots shown in the book were updated to reflect the 2016b release.
- The use of subfunctions in MATLAB programs was updated, since functions no longer need to be stored in separate files.
- New functionality and behavior associated with the 2014 graphics update is included.
- The behavior of the symbolic algebra package in MATLAB has changed dramatically, and the impacts are reflected in changes to Chapter 12. The use of implicit symbolic variables designated with single quotes has largely been eliminated as an acceptable programming technique. Symbolic plotting functions have been replaced with newer functions that will accept both symbolic and function input.
- New data types, such as table, datetime and strings are introduced.
- A number of new functions were introduced throughout the book, largely related to the new data types introduced in 2016.
- Additional problems were added and some problems were modified, based on the feedback from both instructors and students who have used the book. Historic data used in the problems has been updated to current values – for example ACE hurricane information now includes data through 2016.

Dedication and Acknowledgments

This project would not have been possible without the support of both my family and colleagues. Thanks to Mike, Heidi, Meagan, and David, and to my husband, Dr. Steven Purcell. I also benefited greatly from the suggestions for problems related to electricity from Lee Brinton and Gene Riggs of the SLCC Electrical Engineering Department. Their cheerful efforts to educate me on the mysteries of electricity are much appreciated. I'd also like to thank Quentin McRae, also at SLCC, who made numerous suggestions that improved the homework problems. And finally, Art Fox has been my tireless colleague and collaborator for almost 20 years and is responsible in large part for the success of our MATLAB computing courses at SLCC – especially the online versions.

This book is dedicated to my father, Professor George E. Moore, who taught in the Department of Electrical Engineering at the South Dakota School of Mines and Technology for almost 20 years. Professor Moore earned his college degree at the age of 54 after a successful career as a pilot in the United States Air Force and was a living reminder that you are never too old to learn. My mother, Jean Moore, encouraged both him and her two daughters to explore outside the box. Her loving support made it possible for both my sister and I to enjoy careers in engineering—something few women attempted in the early 1970s. I hope that readers of this text will take a minute to thank those people in their lives who've helped them make their dreams come true. Thanks Mom and Dad!

Acknowledgments for the Global Edition

Pearson would like to acknowledge and thank Somitra Kumar Sanadhya, Indraprastha Institute of Information Technology, Delhi, for contributing to the Global Edition, and Nikhil Marriwala, University Institute of Engineering and Technology, Kurukshetra, for reviewing the Global Edition.

This page intentionally left blank

CHAPTER

1 About MATLAB®

Objectives

After reading this chapter, you should be able to:

- Understand what MATLAB® is and why it is widely used in engineering and science.
- Understand the advantages and limitations of the student edition of MATLAB®.
- Formulate problems using a structured problem-solving approach.

1.1 WHAT IS MATLAB®?

MATLAB® is one of a number of commercially available mathematical computation tools, which also include Maple, Mathematica, and MathCad. Despite what proponents may claim, no single one of these tools is "the best." Each has strengths and weaknesses. Each allows you to perform basic mathematical computations. They differ in the way they handle symbolic calculations and more complicated mathematical processes, such as matrix manipulation. For example, MATLAB® (short for **Mat**rix **Lab**oratory) excels at computations involving matrices, whereas Maple excels at symbolic calculations. At a fundamental level, you can think of these applications as sophisticated computer-based calculators. They can perform the same functions as your scientific calculator—and many more. If you have a computer on your desk, you may find yourself using MATLAB® instead of your calculator for even the simplest mathematical applications such as balancing your checkbook. In many engineering classes, the use of applications such as MATLAB® to perform computations is replacing more traditional computer programming. Although applications such as MATLAB® have become standards tool for engineers and scientists, this doesn't mean that you shouldn't learn a high-level language such as C++, Java, or Fortran.

Because MATLAB® is so easy to use, you can perform many programming tasks with it, but it isn't always the best tool for a programming task. It excels at numerical calculations—especially matrix calculations—and graphics, but you wouldn't want to

use it to write a word-processing program. For large applications, such as operating systems or design software, C++, Java, or Fortran would be the applications of choice. (In fact, MATLAB®, which *is* a large application, was originally written in Fortran and later rewritten in C, a precursor of C++.) Usually, high-level applications do not offer easy access to graphing—a task at which MATLAB® excels. The primary area of overlap between MATLAB® and high-level applications is "number crunching"—repetitive calculations or the processing of large quantities of data. Both MATLAB® and high-level applications are good at processing numbers. A "number-crunching" program is generally easier to write in MATLAB®, but usually it will execute faster in C++ or Fortran. The one exception to this rule is calculations involving matrices. MATLAB® is optimized for matrices. Thus, if a problem can be formulated with a matrix solution, MATLAB® executes substantially faster than a similar program in a high-level language.

KEY IDEA
MATLAB® is optimized for matrix calculations.

MATLAB® is available as both professional and student versions. The professional version is probably installed in your college or university computer laboratory, but you may enjoy having the student version at home. MATLAB® is updated regularly; this textbook is based on MATLAB® 7.1. If you are using earlier versions, you will notice a significant difference in the layout of the graphical user interface; however, the differences in coding approaches are minor. There are substantial differences in versions that predate MATLAB® 5.5.

The standard installation of the professional version of MATLAB® is capable of solving various technical problems. Additional capability is available in the form of function toolboxes. These toolboxes are purchased separately, and they may or may not be available to you. You can find a complete list of the MATLAB® product family at The MathWorks website, www.mathworks.com.

1.2 STUDENT EDITION OF MATLAB®

KEY IDEA
MATLAB® is regularly updated.

The professional and student editions of MATLAB® are very similar. Beginning students probably won't be able to tell the difference. Student editions are available for Microsoft Windows, Mac, and Linux operating systems, and can be purchased from college bookstores or online from MathWorks at www.mathworks.com.

MathWorks packages its software in groups called *releases,* and MATLAB® 7.1 is featured, along with other products, such as Simulink, in Release R2016b®. New versions are released every six months. Students may purchase just MATLAB, or a bundle that includes the following products:

- Full MATLAB®
- Simulink, with the ability to build models with up to 1000 blocks (the professional version allows an unlimited number of blocks)
- Symbolic Math Toolbox
- Control Systems Toolbox
- Data Acquisition Toolbox
- Instrument Control Toolbox
- Simulink Control Design
- Signal Processing Toolbox
- DSP System Toolbox
- Statistics and Machine Learning Toolbox
- Optimization Toolbox
- Image Processing Toolbox
- A single-user license, limited to students for use in their classwork (The professional version is licensed either singly or to a group).

Toolboxes other than those included with the student edition may be purchased separately. You should be aware that if you are using a professional installation of MATLAB®, all of the toolboxes available in the student edition may not be available to you.

The biggest difference you should notice between the professional and student editions is the command prompt, which is

```
>>
```

in the professional version, and

```
EDU>>
```

in the student edition.

1.3 HOW IS MATLAB® USED IN INDUSTRY?

The ability to use tools such as MATLAB® is quickly becoming a requirement for many engineering positions. A recent job search on Monster.com found the following advertisement:

> . . . is looking for a System Test Engineer with Avionics experience. Responsibilities include modification of MATLAB® scripts, execution of Simulink simulations, and analysis of the results data. Candidate MUST be very familiar with MATLAB®, Simulink, and C++ . . .

This advertisement isn't unusual. The same search turned up 771 different companies that specifically required MATLAB® skills for entry-level engineers. Widely used in engineering and science fields, MATLAB® is particularly popular for electrical engineering applications. The sections that follow outline a few of the many applications currently using MATLAB®.

KEY IDEA
MATLAB® is widely used in engineering.

1.3.1 Electrical Engineering

MATLAB® is used in electrical engineering for a wide variety of applications. For example, Figure 1.1 includes several images created to help visualize the arrangements of electromagnetic fields in space and time. These images represent real physical situations with practical applications. Cellular communications, medical diagnostics, and home computers are just a few of the technologies that exist thanks to our understanding of this beautiful phenomenon.

1.3.2 Biomedical Engineering

Medical images are usually saved as dicom files (the Digital Imaging and Communications in Medicine standard). Dicom files use the file extension .dcm. The MathWorks

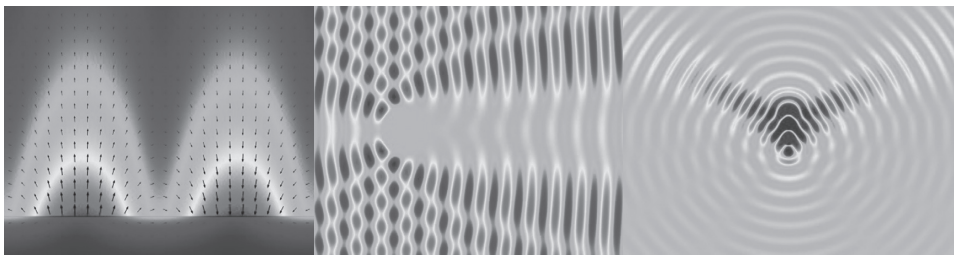


Figure 1.1
Arrangements of electromagnetic fields. (a) surface plasmon polariton; (b) light scattering by a circular metal cylinder; (c) beam forming by a six-element dipole array. (Used with permission of Dr. James R. Nagel, University of Utah Department of Electrical and Computer Engineering.)

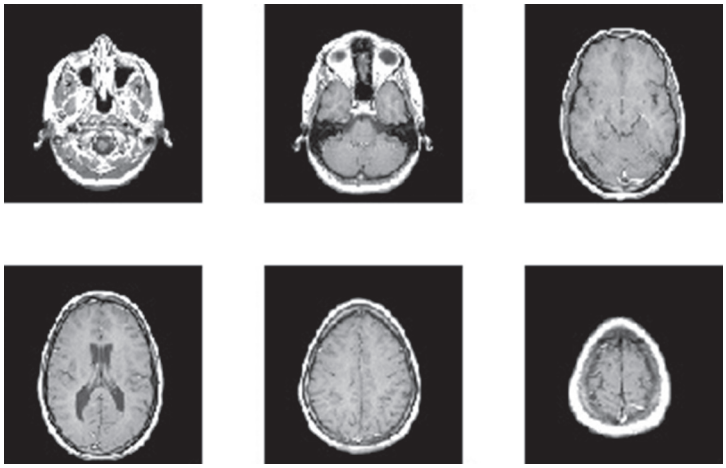


Figure 1.2
Horizontal slices through the brain, based on the sample data file included with MATLAB®.

offers an Image Processing Toolbox that can read these files, making their data available to MATLAB®. (The Image Processing Toolbox is included with the student edition, and is optional with the professional edition.) The Image Processing Toolbox also includes a wide range of functions, many of them especially appropriate for medical imaging. A limited MRI data set that has already been converted to a format compatible with MATLAB® ships with the standard MATLAB® program. This data set allows you to try out some of the imaging functions available both with the standard MATLAB® installation and with the expanded imaging toolbox, if you have it installed on your computer. Figure 1.2 shows six images of horizontal slices through the brain based on the MRI data set.

The same data set can be used to construct a three-dimensional image, such as either of those shown in Figure 1.3. Detailed instructions on how to create these images are included in the MATLAB® tutorial, accessed from the help button on the MATLAB® toolbar.

1.3.3 Fluid Dynamics

Calculations describing fluid velocities (speeds and directions) are important in a number of different fields. Aerospace engineers in particular are interested in the

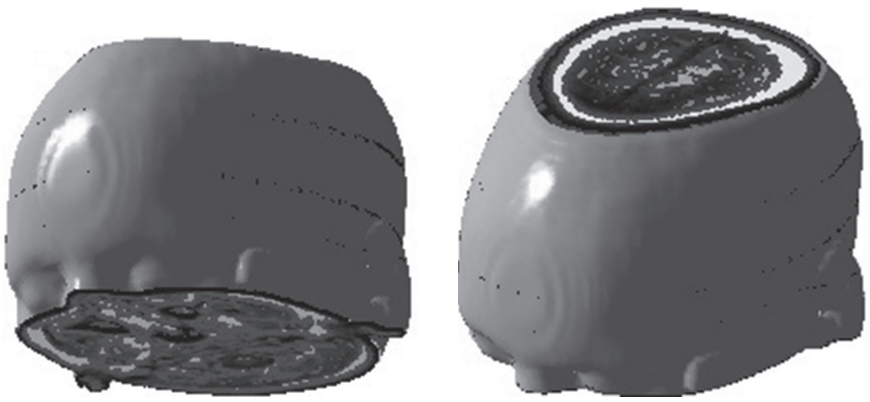


Figure 1.3
Three-dimensional visualization of MRI data, based on the sample data set included with MATLAB®.

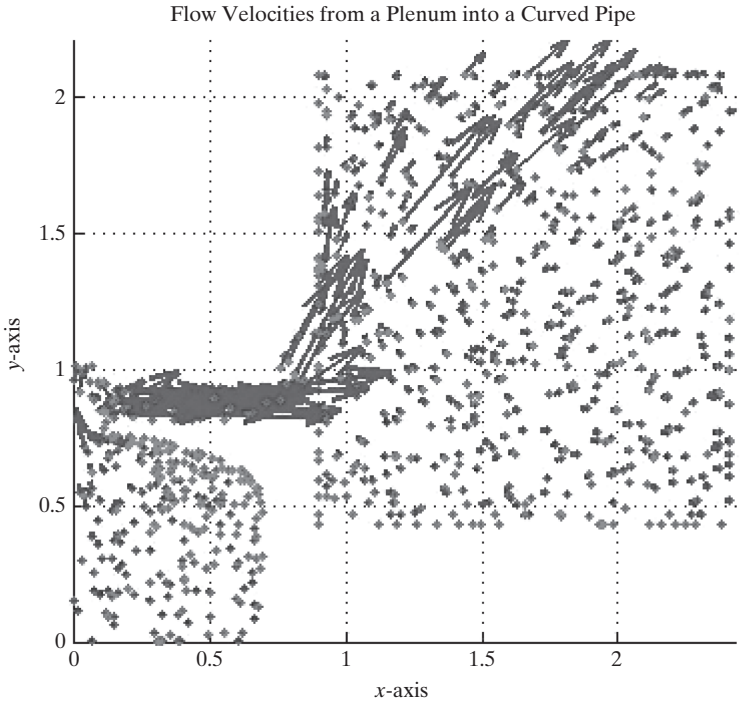


Figure 1.4
Quiver plot of gas behavior in a thrust-vector control device.

behavior of gases, both outside an aircraft or space vehicle and inside the combustion chambers. Visualizing the three-dimensional behavior of fluids is tricky, but MATLAB® offers a number of tools that make it easier. Figure 1.4 represents the flow-field calculation results for a thrust-vector control device as a quiver plot. Thrust-vector control is the process of changing the direction in which a nozzle points (and hence the direction a rocket travels) by pushing on an actuator (a piston-cylinder device). The model in the figure represents a high-pressure reservoir of gas (a plenum) that eventually feeds into the piston and thus controls the length of the actuator.

1.4 PROBLEM SOLVING IN ENGINEERING AND SCIENCE

KEY IDEA
Always use a systematic problem-solving strategy.

A consistent approach to solving technical problems is important throughout engineering, science, and computer programming disciplines. The approach we outline here is useful in courses as diverse as chemistry, physics, thermodynamics, and engineering design. It also applies to the social sciences, such as economics and sociology. Different authors may formulate their problem-solving schemes differently, but they all have the same basic format:

- **State the problem.**
  - Drawing a picture is often helpful in this step.
  - If you do not have a clear understanding of the problem, you are not likely to be able to solve it.
- **Describe the input** values (knowns) and the required **outputs** (unknowns).
  - Be careful to include units as you describe the input and output values. Sloppy handling of units often leads to wrong answers.
  - Identify constants you may need in the calculation, such as the ideal gas constant and the acceleration due to gravity.

- If appropriate, label a sketch with the values you have identified, or group them into a table.

- **Develop an algorithm** to solve the problem. In computer applications, this can often be accomplished with a **hand example**. You'll need to:
  - Identify any equations relating the knowns and unknowns.
  - Work through a simplified version of the problem by hand or with a calculator.
- **Solve** the problem. In this book, this step involves creating a **MATLAB® solution**.
- **Test the solution.**
  - Do your results make sense physically?
  - Do they match your sample calculations?
  - Is your answer really what was asked for?
  - Graphs are often useful ways to check your calculations for reasonableness.

If you consistently use a structured problem-solving approach, such as the one just outlined, you'll find that "story" problems become much easier to solve. Example 1.1 illustrates this problem-solving strategy.

EXAMPLE 1.1

THE CONVERSION OF MATTER TO ENERGY

Albert Einstein (Figure 1.5) is arguably the most famous physicist of the 20th century. He was born in Germany in 1879 and attended school in both Germany and Switzerland. While working as a patent clerk in Bern, he developed his famous theory of relativity. Perhaps the best-known physics equation today is his

$$E = mc^2.$$

This astonishingly simple equation links the previously separate worlds of matter and energy, and can be used to find the amount of energy released as matter is changed in form in both natural and human-made nuclear reactions.

The sun radiates 385×10^{24} J/s of energy, all of which is generated by nuclear reactions converting matter to energy. Use MATLAB® and Einstein's equation to determine how much matter must be converted to energy to produce this much radiation in one day.

1. State the problem
   Find the amount of matter necessary to produce the amount of energy radiated by the sun every day.
2. Describe the input and output

   Input

   Energy: $E = 385 \times 10^{24}$ J/s, which must be converted into the total energy radiated during one day

   Speed of light: $c = 3.0 \times 10^8$ m/s

   Output

   Mass m in kg
3. Develop a hand example
   The energy radiated in one day is

$$385 \times 10^{24}\ \text{J/s} \times 3600\ \text{s/h} \times 24\ \text{h/day} \times 1\ \text{day} = 3.33 \times 10^{31}\ \text{J}.$$

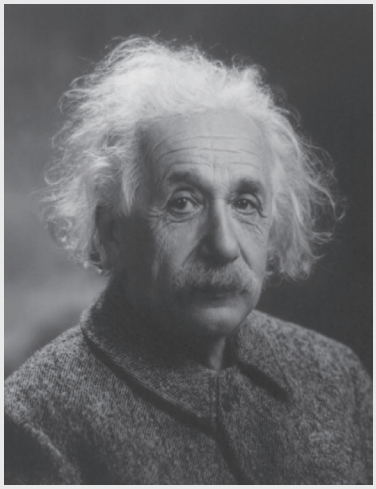


Figure 1.5
Albert Einstein.
(GL Archive/Alamy)

The equation $E = mc^2$ must be solved for m and the values for E and c substituted. Thus,

$$m = \frac{E}{c^2}$$

$$m = \frac{3.33 \times 10^{31}\,\text{J}}{(3.0 \times 10^8\,\text{m/s})^2}$$

$$= 3.7 \times 10^{14}\,\text{J/m}^2\text{s}^2.$$

We can see from the output criteria that we want the mass in kg, so what went wrong? We need to do one more unit conversion:

$$1\text{J} = 1\ \text{kg m}^2/\text{s}^2$$

$$= 3.7 \times 10^{14} \frac{J}{\text{m}^2/\text{s}^2} \times \frac{\text{kg m}^2/\text{s}^2}{J} = 3.7 \times 10^{14}\ \text{kg}$$

4. Develop a MATLAB® solution

At this point, you have not learned how to create MATLAB® code. However, you should be able to see from the following sample code that MATLAB® syntax is similar to that used in most algebraic scientific calculators. MATLAB® commands are entered at the prompt (**>>**), and the results are reported on the next line. The code is as follows:

```
>> E=385e24  The user types in this information
E =

   3.8500e+026    This is the computer's response
>> E=E*3600*24
E =
   3.3264e+031
>> c=3e8
```

```
c =
   300000000
>> m=E/c^2
m =
   3.6960e+014
```

From this point on, we will not show the prompt when describing interactions in the command window.

5. Test the solution

The MATLAB® solution matches the hand calculation, but do the numbers make sense? Anything times 10^{14} is a really large number. Consider, however, that the mass of the sun is 2×10^3 kg. We can calculate how long it would take to consume the mass of the sun completely at a rate of 3.7×10^{14} kg/day. We have

$$\text{Time} = \frac{\text{Mass of the sun}}{\text{Rate of consumption}}$$

$$\text{Time} = \frac{2 \times 10^{30}\ \text{kg}}{3.7 \times 10^{14}\ \text{kg/day}} \times \frac{\text{year}}{365\ \text{days}} = 1.5 \times 10^{13}\ \text{years}.$$

That's 15 trillion years! We don't need to worry about the sun running out of matter to convert to energy in our lifetimes.

CHAPTER

2 MATLAB® Environment

Objectives

After reading this chapter, you should be able to:

- Start the MATLAB® program and solve simple problems in the command window.
- Understand MATLAB®'s use of matrices.
- Identify and use the various MATLAB® windows.
- Define and use simple matrices.
- Name and use variables.
- Understand the order of operation used in MATLAB®.
- Understand the difference between scalar, array, and matrix calculations in MATLAB®.
- Express numbers in either floating-point or scientific notation.
- Adjust the format used to display numbers in the command window.
- Save the value of variables used in a MATLAB® session.
- Save a series of commands in a script M-file.
- Use Section mode.

2.1 GETTING STARTED

Using MATLAB® for the first time is easy; mastering it can take years. In this chapter, we will introduce you to the MATLAB® environment and show you how to perform basic mathematical computations. After reading this chapter, you should be able to start using MATLAB® for homework assignments or on the job. Of course, you will be able to do more as you complete the rest of the chapters.

Because the procedure for installing MATLAB® depends upon your operating system and your computing environment, we will assume that you have already installed MATLAB® on your computer, or that you are working in a computing laboratory with MATLAB® already installed. To start MATLAB® in either the Windows or Mac environment, double-click on the icon on the desktop, or use the start menu to find the

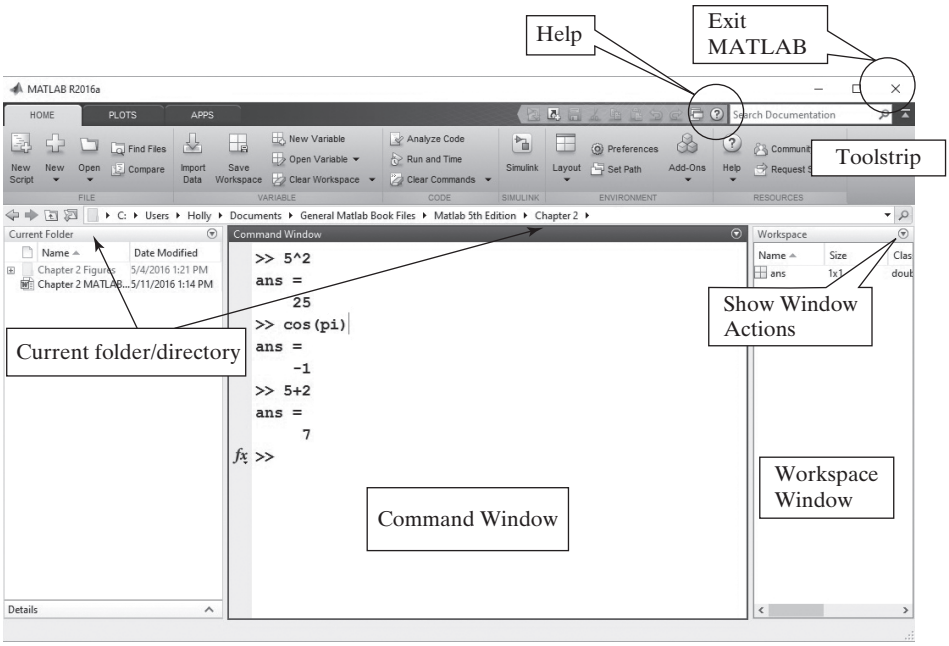


Figure 2.1
MATLAB® 2016a opening window. The MATLAB® environment consists of a number of windows, three of which open in the default view. Others open as needed during a MATLAB® session.

program. In the UNIX environment, type `Matlab` at the shell prompt. No matter how you start it, once MATLAB® opens, you should see the MATLAB® prompt (>> or `EDU>>`), which tells you MATLAB® is ready for you to enter a command. When you have finished your MATLAB® session, you can exit MATLAB® by typing `quit` or `exit` at the MATLAB® prompt. MATLAB® also uses the standard Windows menu bar, so you can exit the program by selecting the close icon (**x**) at the upper right-hand corner of the screen. The default MATLAB® screen, which opens each time you start the program, is shown in Figure 2.1. (There are minor differences in the MATLAB® desktop, depending on the release.)

To start using MATLAB®, you need be concerned only with the command window (in the center of the screen). You can perform calculations in the command window in a manner similar to the way you perform calculations on a scientific calculator. Even most of the syntax is the same. For example, to compute the value of 5 squared, type the command

```
5^2.
```

The following output will be displayed:

```
ans =
                25.
```

Or, to find the value of $\cos(\pi)$, type

```
cos(pi)
```

which results in the output

```
ans =
       -1.
```

KEY IDEA
MATLAB® uses the standard algebraic rules for order of operation.

MATLAB® uses the standard algebraic rules for order of operation, which becomes important when you chain calculations together. These rules will be discussed in Section 2.3.2. Notice that the value of pi is built into MATLAB®, so you do not have to enter it yourself.

HINT

You may think some of the examples are too simple to type in yourself—that just reading the material is sufficient. However, you will remember the material better if you both read it and type it!

Before going any further, try Practice Exercise 2.1.

PRACTICE EXERCISE 2.1

Type the following expressions into MATLAB® at the command prompt, and observe the results. The correct answers can be found on the Pearson website.

1. 5 + 2
2. 5*2
3. 5/2
4. 3 + 2*(4 + 3)
5. 2.54*8/2.6
6. 6.3 − 2.1045
7. 3.6^2
8. 1+2^2
9. sqrt(5)
10. cos(pi)

HINT

You may find it frustrating to learn that when you make a mistake, you cannot just overwrite your command after you have executed it. This occurs because the command window is creating a list of all the commands you have entered. You cannot "un-execute" a command, or "un-create" it. What you can do is enter the command correctly, then execute your new version. MATLAB® offers several ways to make this easier for you. One way is to use the arrow keys, usually located on the right-hand side of your keyboard. The up arrow, ↑, allows you to move through the list of commands you have executed. Once you find the appropriate command, you can edit it, then execute your new version.

2.2 MATLAB® WINDOWS

MATLAB® uses several display windows. The default view, shown in Figure 2.1, includes in the middle, a large ***command window***; located on the right, the ***workspace*** window, and located on the left the ***current folder window***. In addition, the ***command history window***; ***document windows***, ***graphics windows***, and ***editing windows*** will automatically open when needed. Each will be described in the sections that follow. MATLAB® also includes a built-in help tutorial that can be accessed from the toolstrip by selecting the question mark icon, as shown in Figure 2.1. To personalize your desktop, you can resize any of these windows, stack them on top of each other, close the ones you are not using, or "undock" them from the desktop by using the "Show Workspace Actions" menu located in the upper right-hand corner of each window.

You can restore the default configuration by selecting Layout on the toolstrip, then selecting Default, or you can add additional windows by selecting Layout and choosing the windows you would like to see.

2.2.1 Command Window

The command window is located in the center pane of the default view of the MATLAB® screen, as shown in Figure 2.1. The command window offers an environment similar to a scratch pad. Using it allows you to save the values you calculate, but not the ***commands*** used to generate those values. If you want to save the command sequence, you will need to use the editing window to create an a script stored as an *M-file*. Scripts are will be described in Section 2.4.2. Both approaches are valuable. Before we introduce scripts, we will concentrate on using the command window.

KEY IDEA
The command window is similar to a scratch pad.

2.2.2 Command History

The command history window records the commands you issued in the command window. It does not open in the default view in MATLAB 2016b, but you can add it to your desktop by selecting Layout → Command History and checking "docked." The examples in this book will show the Command History in a docked configuration. When you exit MATLAB®, or when you issue the `clc` command, the command window is cleared, but the command history window retains a list of all your commands. You may clear the command history from the "Show Command History Actions" dropdown menu, located in the upper right-hand corner of the window. If you work on a public computer, as a security precaution, MATLAB®'s defaults may be set to clear the history when you exit MATLAB®. If you entered the earlier sample commands listed in this book, notice that they are repeated in the command history window. This window is valuable for a number of reasons, including allowing you to review previous MATLAB® sessions. It can also be used to transfer commands to the command window. For example, first clear the contents of the command window by typing

KEY IDEA
The command history records all of the commands issued in the command window.

```
clc.
```

This action clears the command window but leaves the data in the command history window intact. You can transfer any command from the command history window to the command window by double-clicking (which also executes the command), or by clicking and dragging the line of code into the command window. Try double-clicking

```
cos(pi)
```

in the command history window. The command is copied into the command window and executed. It should return

```
ans =
      -1.
```

Now click and drag

```
5^2
```

from the command history window into the command window. The command will not execute until you hit Enter, and then you will get the result:

```
ans =
      25
```

You will find the command history useful as you perform more complicated calculations in the command window.

KEY IDEA
The workspace window lists information describing all the variables created by the program.

2.2.3 Workspace Window

The workspace window keeps track of the ***variables*** you have defined as you execute commands in the command window. These variables represent values stored in the computer memory, which are available for you to use. If you have been doing the examples, the workspace window should show just one variable, `ans`, and indicate that it has a value of 25, and is a double array:

Name	Size	Class	Value
ans	1 × 1	double	25

(Your view of the workspace window may be slightly different, depending on how your installation of MATLAB® is configured.)

Set the workspace window to show more about the displayed variables by right-clicking on the bar with the column labels. Check `bytes`, in addition to `name`, `value`, `class` and `size`. Your workspace window should now display the following information, although you may need to resize the window to see all the columns:

Name	Size	Class	Bytes	Value
ans	1 × 1	double	8	25

KEY IDEA
The default data type is double-precision floating-point numbers stored in a matrix.

The yellow grid-like symbol indicates that the variable **ans** is an array. The size, 1 × 1, tells us that it is a single value (one row by one column) and therefore a scalar. The array uses 8 bytes of memory. MATLAB® was written in C, and the class designation tells us that in the C language, `ans` is a double-precision floating-point array. For our needs, it is enough to know that the variable `ans` can store a floating-point number (a number with a decimal point). Actually, MATLAB® considers every number you enter to be a floating-point number, whether you insert a decimal point or not.

In addition to information about the size of the arrays and type of data stored in them, you can also choose to display statistical information about the data. Once again, right click the bar in the workspace window that displays the column headings. Notice that you can select from a number of different statistical measures, such as the max, min, and standard deviation.

You can define additional variables in the command window, and they will be listed in the workspace window. For example, typing

```
A = 5
```

returns

```
A =
     5.
```

Notice that the variable **A** has been added to the workspace window, which lists variables in alphabetical order. Variables beginning with lowercase letters are listed first, followed by variables starting with capital letters.

Name	Size	Class	Bytes	Value
ans	1 × 1	double	8	25
A	1 × 1	double	8	5

In Section 2.3.2, we will discuss in detail how to enter matrices into MATLAB®. For now, you can enter a simple one-dimensional matrix by typing

```
B = [1, 2, 3, 4].
```

This command returns

```
B =
     1 2 3 4.
```

The commas are optional; you would get the same result with

```
B = [1 2 3 4]
B =
     1 2 3 4.
```

Notice that the variable **B** has been added to the workspace window, and that it is a 1×4 array:

Name	Size	Class	Bytes	Value
ans	1×1	double	8	25
A	1×1	double	8	5
B	1×4	double	32	[1, 2, 3, 4]

You can define two-dimensional matrices in a similar fashion. Semicolons are used to separate rows. For example,

```
C = [1 2 3 4; 10 20 30 40; 5 10 15 20]
```

returns

```
C =
     1    2    3    4
     10   20   30   40
     5    10   15   20.
```

Name	Size	Class	Bytes	Value
ans	1×1	double	8	25
A	1×1	double	8	5
B	1×4	double	32	[1, 2, 3, 4]
C	3×4	double	96	$<3 \times 4$ double $>$

Notice that **C** appears in the workspace window as a 3×4 matrix. To conserve space, the values stored in the matrix are not listed.

You can recall the values for any variable by typing in the variable name. For example, entering

```
A
```

returns

```
A =
     5.
```

Although the only variables we have introduced are matrices containing numbers, other types of variables are possible.