GLOBAL EDITION

Introduction to MATLAB®

THIRD EDITION

Delores M. Etter

ALWAYS LEARNING

PEARSON

Introduction to MATLAB®

Introduction to MATLAB®

Third Edition
Global Edition

Delores M. Etter
Southern Methodist University
Dallas, Texas

Global Edition contributions by

Anju Mishra
Amity University
Uttar Pradesh, India

PEARSON

Hoboken • Boston • Columbus
San Francisco • New York • Indianapolis • London
Toronto • Sydney • Singapore • Tokyo • Montreal
Dubai • Madrid • Hong Kong • Mexico City
Munich • Paris • Amsterdam • Cape Town

Vice President and Editorial Director, ECS: *Marcia Horton*
Executive Editor: *Holly Stark*
Editorial Assistant: *Carlin Heinle*
Head of Learning Asset Acquisition, Global Editions: *Laura Dent*
Acquisition Editor, Global Editions: *Subhasree Patra*
Acquisition Editor, Global Editions: *Aditee Agarwal*
Assistant Project Editor, Global Editions: *Paromita Banerjee*
Director of Marketing: *Margaret Waples*
Marketing Manager: *Tim Galligan*
Marketing Assistant: *Jon Bryant*

Program Management Team Lead: *Scott Disanno*
Program Manager: *Clare Romeo*
Project Manager: *Priyadharshini Dhanagopal*
Senior Production Manufacturing Controller, Global Editions: *Trudy Kimber*
Senior Operations Specialist: *Nick Sklitsis*
Operations Specialist: *Linda Sager*
Permissions Project Manager: *Karen Sanatar*
Full-Service Project Management: *Jouve North America*
Cover Image: © *PinnacleAnimates/Shutterstock*

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

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

© Pearson Education Limited 2015

The rights of Delores M. Etter to be identified as the author of this work have been asserted by her in accordance with the Copyright, Designs and Patents Act 1988.

Authorized adaptation from the United States edition, entitled Introduction to MATLAB®, 3rd Edition, ISBN 978-0-13-377001-8 by Delores M. Etter, *published by Pearson Education © 2015.*

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

ISBN 10: 1-292-01939-5
ISBN 13: 978-1-292-01939-0

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 in New Baskerville Std by Jouve North America
Printed and bound by Courier Kendallville in The United States of America

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

MATLAB is a registered trademark of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098.

Contents

About This Book

Engineers and scientists use computers to solve a variety of problems, ranging from the evaluation of a simple function to solving a system of equations. MATLAB® has become the **technical computing environment** of choice for many engineers and scientists because it is a single interactive system that incorporates **numeric computations**, **scientific visualization**, and **symbolic computation**.

Because the MATLAB computing environment is one that a new engineer is likely to encounter in a job, it is a good choice for an introduction to computing for engineers. This book is appropriate for use as an introductory engineering text or as a supplemental text in an advanced course. It is also useful as a professional reference.

This text was written to introduce engineering problem solving with the following objectives:

- Present a consistent **methodology for solving engineering problems**.
- Describe the exceptional **computational and visualization capabilities of MATLAB**.
- Illustrate the problem-solving process through a variety of **engineering examples and applications**.

To accomplish these objectives, Chapter 1 presents a five-step process that was developed by the author and is used to consistently solve engineering problems throughout the text. The rest of the chapters present the capabilities of MATLAB for solving engineering problems using specific examples from many different engineering disciplines.

TEXT ORGANIZATION

This book is designed for use in a variety of engineering and science course settings as a **primary text** for introductory students and as a **supplement for intermediate or advanced courses**. It is feasible to cover Chapters 1 through 8 in a one-semester course for a complete introduction to MATLAB's capabilities. If a briefer introduction to MATLAB is desired, we suggest that Chapters 1 through 4 be covered along with selected topics from Chapters 5 through 8.

PREREQUISITES

No prior experience with the computer is assumed. The mathematical background needed for Chapters 1 through 6 is **college algebra** and **trigonometry**. More advanced mathematics is needed for some of the material in later chapters.

PROBLEM-SOLVING METHODOLOGY

The emphasis on engineering and scientific problem solving is an important part of this text. Chapter 1 introduces a **five-step process for solving engineering problems** using the computer:

1. State the problem clearly.
2. Describe the input and output information.

3. Work a simple example by hand.
4. Develop an algorithm and convert it to MATLAB.
5. Test the solution with a variety of data.

To reinforce the development of problem-solving skills, each of these steps is identified every time a complete solution to an engineering problem is developed.

ENGINEERING AND SCIENTIFIC APPLICATIONS

Throughout the text, emphasis is placed on incorporating **real-world engineering and scientific examples** with solutions and usable code. Each chapter begins with a discussion of a significant engineering achievement. The chapter then includes examples related to this achievement. These examples include analysis of data from the following applications:

- temperatures from a sensor
- mass of air in a wind tunnel
- velocity and acceleration for unducted fan engine
- saturation vapor pressure for water at different temperatures
- Great Circle distances using GPS coordinates
- wind speeds from the Mount Washington Observatory
- wind speeds generated for a flight simulator
- ocean wave interaction
- performance quality scores
- mass calculations for a spacecraft
- current values for electrical circuits
- projectile range and impact
- interpolation using steam tables
- flow model for water in a culvert
- population models

VISUALIZATION

The visualization of the information related to a problem is a key advantage of using MATLAB for developing and understanding solutions. Therefore, it is important to learn to generate **plots** in a variety of formats to use when analyzing, interpreting, and evaluating data. We begin using plots with the first MATLAB program presented in Chapter 1 and continually expand **plotting capabilities** within the remaining chapters. Chapter 4 covers all the main types of plots and graphs.

SOFTWARE ENGINEERING CONCEPTS

Engineers and scientists are also expected to develop and implement **user-friendly** and **reusable** computer solutions. Therefore, learning software engineering techniques is crucial to successfully develop these computer solutions. Readability and documentation are stressed in the development of programs. Through MATLAB, users are able to write portable code that can be transferred from one computer platform to another. Additional topics that relate to software engineering issues are discussed in Chapter 1 and include the **software life cycle**, **maintenance**, and **software prototypes**.

PROBLEM SETS

Learning any new skill requires practice at a number of **different levels of difficulty**. Each chapter ends with a set of problems. These are problems that relate to a variety of engineering applications with the level of difficulty ranging from straightforward to longer assignments. **Engineering data sets** are included, for many of the problems, to use in testing.

STUDENT AIDS

Each chapter ends with a **Summary** that reviews the topics covered and includes a list of **Key Terms**. A **MATLAB Summary** lists all the special symbols, commands, and functions defined in the chapter. **Hints** are provided to help the student avoid some of the common errors.

WHAT'S NEW IN THIS EDITION?

- The discussions, screen captures, examples, and problem solutions have been updated to reflect MATLAB Version 8.2, R2013b.
- A discussion of the new Help browser is included along with screen captures to illustrate using this feature.
- The section on random number generation has been rewritten to reflect changes relative to the random number seed and to include the new function for generating random integers.
- The section on numerical integration has been rewritten to support the new integration function.
- Updated examples and discussion for current hardware and software are included throughout the text.
- Updated discussions and examples of importing and exporting data with other applications, such as Excel.

Acknowledgments

I want to acknowledge the outstanding work of the publishing team at Prentice Hall. My first MATLAB text was published in 1993, so some of us have worked together for many years. I would like to especially acknowledge the support of Marcia Horton, Holly Stark, Clare Romeo, Scott Disanno, and Greg Dulles. I would also like to express my gratitude to my husband, a mechanical/aerospace engineer, for his help in developing some of the engineering applications. Finally, I want to recognize the important contributions of the many students in my introductory courses for their feedback on the explanations, the examples, and the problems.

Delores M. Etter
Texas Instruments Distinguished Chair in Engineering Education
Professor, Department of Electrical Engineering
The Bobby B. Lyle School of Engineering
Southern Methodist University
Dallas, Texas

Pearson would also like to thank and acknowledge B. R. Chandavarkar (NITK Surathkal), Debaprasad Das (Assam University), and Rohit P. Tahiliani (NMAM Institute of Technology) for reviewing the Global Edition.

CHAPTER

1 An Introduction to Engineering Problem Solving

Objectives

After reading this chapter, you should be able to

- describe some important engineering achievements,
- understand the relationship of MATLAB with computer hardware and software, and
- describe a five-step process for solving engineering problems.

ENGINEERING ACHIEVEMENTS

Engineers solve real-world problems using scientific principles from disciplines that include computer science, mathematics, physics, biology, and chemistry. It is this variety of subjects and the challenge of real problems that have a positive impact on our world, which makes engineering so interesting and rewarding. For example, engineers are working to develop techniques to provide access to clean water to people around the world. Engineers are working to make solar energy more economical so that it can give more than the 1 percent of our energy that it provides today. Engineers work to reduce pollution through developing ways to capture and store excess carbon dioxide in our manufacturing plants. Engineers restore and improve our urban and transportation infrastructure. At the beginning of each chapter, we will present a short discussion on a significant engineering achievement, and in that chapter, we will solve small problems related to that application.

1.1 ENGINEERING ENVIRONMENT

Engineers work in an environment that requires a **strong technical background**, and the computer will be the primary computational tool of most engineers. The focus of this text is to teach you the fundamentals of one of the most widely used engineering tools—MATLAB. However, engineers in the twenty-first century must also have many nontechnical skills and capabilities. The computer is also useful in developing additional nontechnical abilities.

Engineers need **strong communication skills** both for oral presentations and for the preparation of written material. Computers provide the software to assist in writing outlines and developing materials, such as graphs, for presentations and technical reports.

The **design/process/manufacture path**, which consists of taking an idea from a concept to a product, is one that engineers must understand firsthand. Computers are used in every step of this process, from design analysis, machine control, robotic assembly, quality assurance to market analysis.

Engineering teams today are **interdisciplinary teams**. Learning to interact in teams and to develop organizational structures for effective team communication is important for engineers. You are likely to be part of a diverse engineering team in which members are located around the globe, and thus, there are many additional challenges for teams that are not geographically located.

The engineering world is a **global** one. To be effective, you need to understand different cultures, political systems, and business environments. Courses in these topics and in foreign languages help provide some understanding, but exchange programs with international experiences provide invaluable knowledge in developing a broader understanding of the world.

Engineers are **problem solvers**, but problems are not always formulated carefully in the real world. An engineer must be able to extract a problem statement from a problem discussion and then determine the important issues. This involves not only developing order, but also learning to correlate chaos. It means not only analyzing the data, but also synthesizing a solution using many pieces of information. The integration of ideas can be as important as the decomposition of the problem into manageable pieces. A problem solution may involve not only abstract thinking about the problem, but also experimental learning from the problem environment.

Problem solutions must also be considered in their **societal context**. Environmental concerns should be addressed as alternative solutions to problems are being considered. Engineers must also be conscious of ethical issues in providing test results, quality verifications, and design limitations. Ethical issues are never easy to resolve, and some of the exciting new technological achievements bring ethical issues with them.

The material presented in this text is only one step in building the knowledge, confidence, and understanding needed by engineers today. We begin the process with a brief discussion of computing systems and an introduction to a problem-solving methodology that will be used throughout this text as we use MATLAB to solve engineering problems.

1.2 ENGINEERING COMPUTING AND MATLAB

Before we begin discussing MATLAB, a brief discussion on computing is useful, especially for those who have not had lots of experience with computers. A computer is a machine that is designed to perform operations that are specified with

a set of instructions called a program. Computer ***hardware*** refers to the computer equipment, such as a notebook computer, a thumb drive, a keyboard, a flat-screen monitor, or a printer. Computer ***software*** refers to the programs that describe the steps we want the computer to perform. This can be software that we have written, or it can be programs that we download or purchase, such as computer games. Our computer hardware/software can be self-contained, as in a notebook computer. A computer can also access both hardware and software through a computer network, and through access to the Internet. In fact, cloud computing provides access to hardware, software, and large data sets through remote networks.

1.2.1 Computer Hardware

All computers have a common internal organization as shown in Figure 1.1. The processor is the part of the computer that controls all the other parts. It accepts input values (from a device such as a keyboard or a data file) and stores them in memory. It also interprets the instructions in a computer program. If we want to add two values, the processor will retrieve the values from memory and send them to the arithmetic logic unit (ALU). The ALU performs the addition, and the processor then stores the result in memory. The processing unit and the ALU use internal memory composed of read-only memory (ROM) and random access memory (RAM); data can also be stored in external storage devices such as external drives or thumb drives. The processor and the ALU together are called the ***central processing unit*** (CPU). A ***microprocessor*** is a CPU that is contained in a single integrated-circuit chip, which contains millions of components in an area much smaller than a postage stamp.

Many inexpensive printers today use ink-jet technology to print both color copies and black-and-white copies. We can also store information on a variety of digital memory devices, including CDs and DVDs. A printed copy of information is called a ***hard copy***, and a digital copy of information is called an ***electronic copy*** or a ***soft copy***. Many printers today can also perform other functions such as copying, faxing, and scanning.

Computers come in all sizes, shapes, and forms. In fact, most of our phones today contain CPUs and store programs that they can execute. Smartphones also contain a graphics processing unit, a significant amount of RAM, and are trending to multicore (or multiprocessor), low-power CPUs. Many homes today have

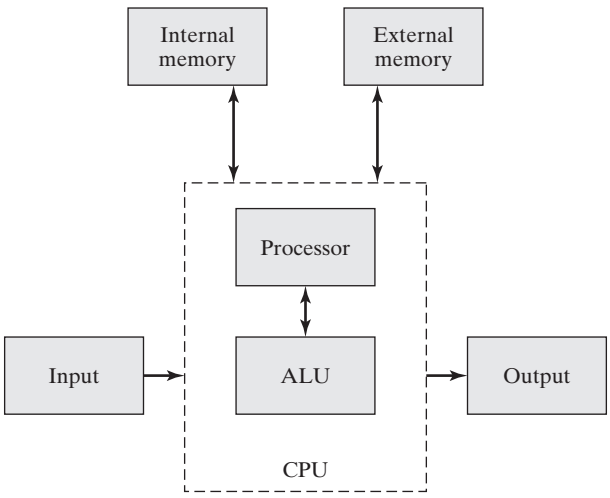


Figure 1.1
Internal organization of a computer.

personal computers that are used for a variety of applications, including e-mail, financial budgeting, and games; these computers are typically desktop computers with separate monitors and keyboards. Notebook computers contain all their hardware in a small footprint, and thus become very convenient. For some people, tablet computers (such as the iPad) and smartphones are even replacing the use of the desktop and notebook computers.

1.2.2 Computer Software

Computer software contains the instructions or commands that we want the computer to perform. There are several important categories of software, including operating systems, software tools (MATLAB is a software tool), and language compilers. Figure 1.2 illustrates the interaction among these categories of software and the computer hardware. We now discuss each of these software categories in more detail.

Operating Systems

Some software, such as an operating system, typically comes with the computer hardware when it is purchased. The ***operating system*** provides an interface between you (the user) and the hardware by providing a convenient and efficient environment in which you can select and execute the software application on your system. The component of the operating system that manages the interface between the hardware and software applications is called a ***kernel***. Examples of desktop operating systems include Windows, Mac OS, Unix, and Linux. Operating systems for smartphones include Android (a Linux variant) and iOS (a Unix variant).

Operating systems also contain a group of programs called ***utilities*** that allow you to perform functions such as printing files, copying files from one folder to another, and listing the files in a folder. Most operating systems today simplify using these utilities through icons and menus.

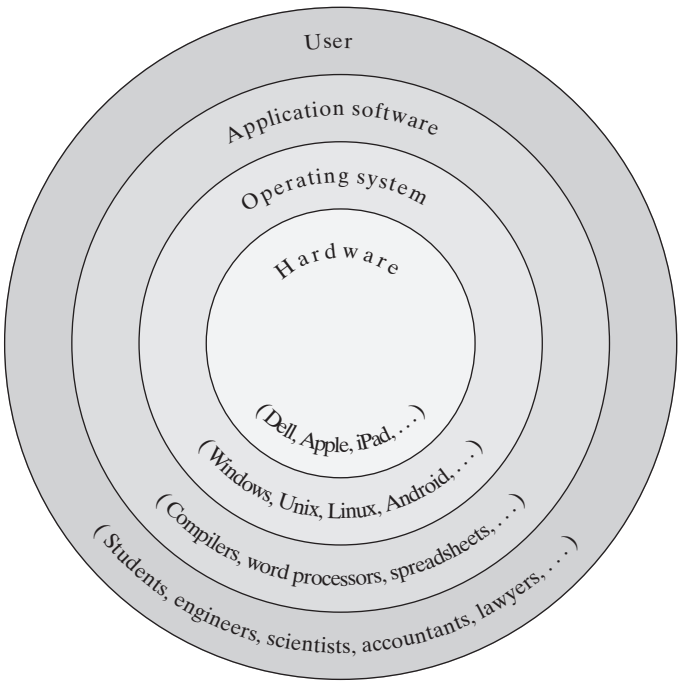


Figure 1.2
Interactions between software and hardware.

Software Tools

Software tools are programs that have been written to perform common operations. For example, **word processors** like Microsoft Word are programs that allow you to enter and format text. They allow you to download information from the Internet into the file, and allow you to enter mathematical equations. They also can check your grammar and spelling. Most word processors also allow you to produce documents that have figures, images, and can print in two columns. These capabilities allow you to perform desktop publishing from a notebook computer.

Spreadsheet programs like Excel are software tools that allow you to easily work with data that can be displayed in a grid of rows and columns. Spreadsheets were initially developed to be used for financial and accounting applications, but many science and engineering problems can be easily solved with spreadsheets. Most spreadsheet packages include plotting capabilities, so they are especially useful in analyzing and displaying information in charts. Database management tools allow you to analyze and "mine" information from large data sets.

Another important category of software tools is **mathematical computation tools**. This category includes MATLAB and Mathematica. Not only do these tools have very powerful mathematical commands, but they are also graphics tools that provide extensive capabilities for generating graphs. This combination of computational and visualization power make them particularly useful tools for engineers.

If an engineering problem can be solved using a software tool, it is usually more efficient to use the software tool than to write a program in a computer language. The distinction between a software tool and a computer language is becoming less clear as some of the more powerful software tools include their own language in addition to having specialized operations. (MATLAB is both a software tool and a programming language.)

Computer Languages

Computer languages can be described in terms of generations. The first generation of computer languages is machine languages. ***Machine languages*** are tied closely to the design of the computer hardware, and are often written in binary strings consisting of 0s and 1s. Therefore, machine language is also called binary language.

An ***assembly language*** is also unique to a specific computer design, but its instructions are written in symbolic statements instead of binary. Assembly languages usually do not have many statements; thus, writing programs in assembly language can be tedious. In addition, to use an assembly language you must also know information that relates to the specific hardware. Instrumentation that contains microprocessors often requires that the programs operate very fast; thus, the programs are called real-time programs. These real-time programs are usually written in assembly language to take advantage of the specific computer hardware in order to perform the steps faster. Assembly languages are second generation languages.

Third generation languages use English-like commands. These languages include C, C++, C#, and Java. Writing programs in a ***high-level language*** is certainly easier than writing programs in machine language or in assembly language. However, a high-level language contains a large number of commands and an extensive set of ***syntax*** (or grammar) rules for using these commands.

MATLAB is considered a fourth generation programming language because of its powerful commands, user interfaces, and its ability to interface to other languages. Higher-level languages are still primarily in research phases and tend to be domain specific.

1.2.3 Executing a Computer Program

A program written in a high-level language such as C must be translated into machine language before the instructions can be executed by the computer. A special program called a ***compiler*** is used to perform this translation. Thus, in order to write and execute C programs, we must have a C compiler. The C compilers are available as separate software packages for use with specific operating systems.

If any errors (often called ***bugs***) are detected by the compiler during compilation, corresponding error messages are printed. We must correct our program statements and then perform the compilation step again. The errors identified during this stage are called ***compiler errors*** or compile-time errors. For example, if we want to divide the value stored in a variable called sum by 3, the correct expression in C is sum/3; if we incorrectly write the expression using the backslash, as in sum\3, we will get a compiler error. The process of compiling, correcting statements (or ***debugging***), and recompiling is often repeated several times before the program compiles without compiler errors. When there are no compiler errors, the compiler generates a program in machine language that performs the steps specified by the original C program. The original C program is referred to as the source program, and the machine language version is called an object program. Thus, the source program and the object program specify the same steps; but the source program is written in a high-level language, and the object program is specified in machine language.

Once the program has compiled correctly, additional steps are necessary to prepare the object program for execution. This preparation involves linking other machine language statements to the object program and then loading the program into memory. After this linking/loading, the program steps are executed by the computer. New errors called execution errors, run-time errors, or ***logic errors*** may be identified in this stage; they are also called program bugs. Execution errors often cause the termination of a program. For example, the program statements may attempt to perform a division by zero, which generates an execution error. Some execution errors do not stop the program from executing, but they cause incorrect results to be computed. These types of errors can be caused by programmer errors in determining the correct steps in the solutions and by errors in the data processed by the program. When execution errors occur due to errors in the program statements, we must correct the errors in the source program and then begin again with the compilation step. Even when a program appears to execute properly, we must check the answers carefully to be sure that they are correct. The computer will perform the steps precisely as we specify, and if we specify the wrong steps, the computer will execute these wrong (but syntactically legal) steps and thus present us with an answer that is incorrect.

The process of compilation, linking/loading, and execution is outlined in Figure 1.3. The process of converting an assembly language program to binary is performed by an assembler program, and the corresponding processes are called assembly, linking/loading, and execution.

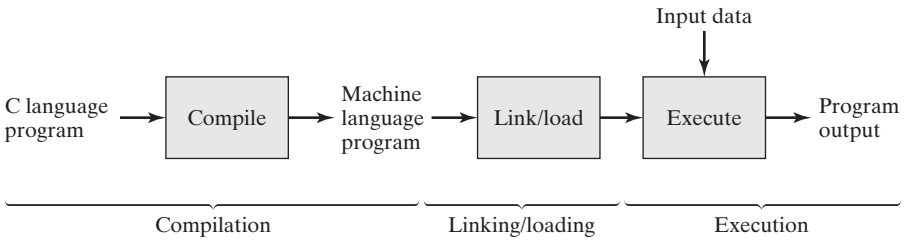


Figure 1.3
Program compilation/loading, linking, and execution.

1.2.4 Software Life Cycle

The cost of a computer solution to a problem can be estimated in terms of the cost of the hardware and the cost of the software. The majority of the cost in a computer solution today is in the cost of the software, and thus, a great deal of attention has been given to understanding the development of a software solution.

The development of a software project generally follows definite steps or cycles, which are collectively called the ***software life cycle***. These steps typically include project definition, detailed specification, coding and modular testing, integrated testing, and maintenance. (These steps will be explained in more detail in later chapters.) Software maintenance is a significant part of the cost of a software system. This maintenance includes adding enhancements to the software, fixing errors identified as the software is used, and adapting the software to work with new hardware and software. The ease of providing maintenance is directly related to the original definition and specification of the solution because these steps lay the foundation for the rest of the project. The problem-solving process that we present in the next section emphasizes the need to define and specify the solution carefully before beginning to code or test it.

One of the techniques that has been successful in reducing the cost of software development both in time and cost is the development of ***software prototypes***. Instead of waiting until the software system is developed and then letting the users work with it, a prototype of the system is developed early in the life cycle. This prototype does not have all the functions required of the final software, but it allows the user to use it early in the life cycle, and to make desired modifications to the specifications. Making changes earlier in the life cycle is both cost- and time-effective. It is not uncommon for a software prototype to be developed in MATLAB, and then for the final system to be developed in another language.

As an engineer, it is very likely that you will need to modify or add additional capabilities to existing software that has been developed using a software tool or a high-level language. These modifications will be much simpler if the existing software is well-structured and readable and if the documentation that accompanies the software is up-to-date and clearly written. For these reasons, we stress developing good habits that make programs more readable and self-documenting.

1.3 AN ENGINEERING PROBLEM-SOLVING METHODOLOGY

Problem solving is a key part not only of engineering courses, but also of courses in computer science, mathematics, physics, and chemistry. Therefore, it is important to have a consistent approach to solving problems. It is also helpful if the approach is general enough to work for all these different areas, so that we do not have to learn one technique for solving mathematics problems, a different technique for solving physics problems, and so on. The problem-solving process that we present works for engineering problems and can be tailored to solve problems in other areas as well. However, it does assume that we are using a computer to help solve the problem.

The process, or methodology, for problem solving that we will use throughout this text has five steps:

1. State the problem clearly.
2. Describe the input and output information.
3. Work the problem by hand (or with a calculator) for a simple set of data.
4. Develop a MATLAB solution.
5. Test the solution with a variety of data.

We now discuss each of these steps using data collected from a physics laboratory experiment as an example.

EXAMPLE 1.1

TEMPERATURE ANALYSIS AND PLOT

Assume that we have collected a set of temperatures from a sensor on a piece of equipment that is being used in an experiment. The temperature measurements shown in Table 1.1 are taken every 30 seconds, for 5 minutes, during the experiment. We want to compute the average temperature, and we also want to plot the temperature values.

Table 1.1 Experimental Temperature Data

Time, minutes	Temperature, F
0.0	105
0.5	126
1.0	119
1.5	129
2.0	132
2.5	128
3.0	131
3.5	135
4.0	136
4.5	132
5.0	137

SOLUTION

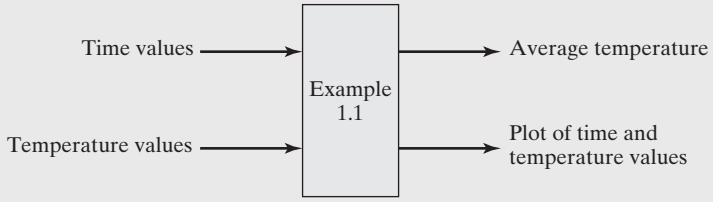
1. Problem Statement

The first step is to state the problem clearly. It is extremely important to give a clear, concise statement of the problem, in order to avoid any misunderstandings. For this example, the statement of the problem is as follows:

Compute the average of a set of temperatures. Then plot the time and temperature values.

2. Input/Output Description

The second step is to describe carefully the information that is given to solve the problem and then to identify the values to be computed. These items represent the input and the output for the problem and collectively can be called input/output, or I/O. For many problems, it is useful to create a diagram that shows the input and output. At this point, the program is called an abstraction because we are not defining the steps to determine the output; instead, we are only showing the information that is used to compute the output. The I/O diagram for this example is as follows:



3. Hand Example

The third step is to work the problem by hand or with a calculator, using a simple set of data. This step is very important and should not be skipped, even for simple problems. This is the step in which you work out the details of the solution to the problem. If you cannot take a simple set of numbers and compute the output (either by hand or with a calculator), you are not ready to move on to the next step. You should reread the problem and perhaps consult reference material. For this problem, the only calculation is computing the average, or mean value, of a set of temperature values. Assume that we use the first three sets of data for the hand example. By hand, we compute the average to be $(105 + 126 + 119)/3$, or 116.6667.

4. MATLAB Solution

Once you can work the problem for a simple set of data, you are ready to develop an ***algorithm***, which is a step-by-step outline of the solution to the problem. For simple problems such as this one, the algorithm can be written immediately using MATLAB commands. For more complicated problems, it may be necessary to write an outline of the steps and then decompose the steps into smaller steps that can be translated into MATLAB commands. One of the strengths of MATLAB is that its commands match very closely to the steps that we use to solve engineering problems. Thus, the process of determining the steps to solve the problem also determines the MATLAB commands. At this point, we know that you do not yet understand the MATLAB commands. However, we present the solution so you can observe that the MATLAB steps match closely to the solution steps from the hand example:

```
%--------------------------------------------------------------
%  Example 1_1 This program computes the average
%  temperature and plots the temperature data.
%
time = [0.0,0.5,1.0];
temps = [105,126,119];
average = mean(temps)
plot(time,temps),title('Temperature Measurements'),
    xlabel('Time, minutes'),
    ylabel('Temperature, degrees F'),grid
%--------------------------------------------------------------
```

The words that follow percent signs are comments to help us in reading the MATLAB statements. If a MATLAB statement assigns or computes a value, it will also print the value on the screen if the statement does not end in a semicolon. Thus, the values of **`time`** and **`temps`** will not be printed, because the statements that assign

(*continued*)

them values end with semicolons. The value of the average will be computed and printed on the screen, because the statement that computes it does not end with a semicolon. Finally, a plot of the time and temperature data will be generated.

5 Testing

The final step in our problem-solving process is testing the solution. We should first test the solution with the data from the hand example, because we have already computed the solution to it. When the previous statements are executed, the computer displays the following output:

```
average =
         116.6667
```

A plot of the data points is also shown on the screen. Because the value of the average computed by the program matches the value from the hand example, we now replace the data from the hand example with the data from the physics experiment using these replacement statements:

```
time = [0.0,0.5,1.0,1.5,2.0,2.5,3.0,3.5,4.0,4.5,5.0];
temps = [105,126,119,129,132,128,131,135,136,132,137];
```

When the commands in the program are executed with the complete set of data, the computer displays the following output:

```
average =
         128.1818
```

The plot in Figure 1.4 is also shown on the screen.

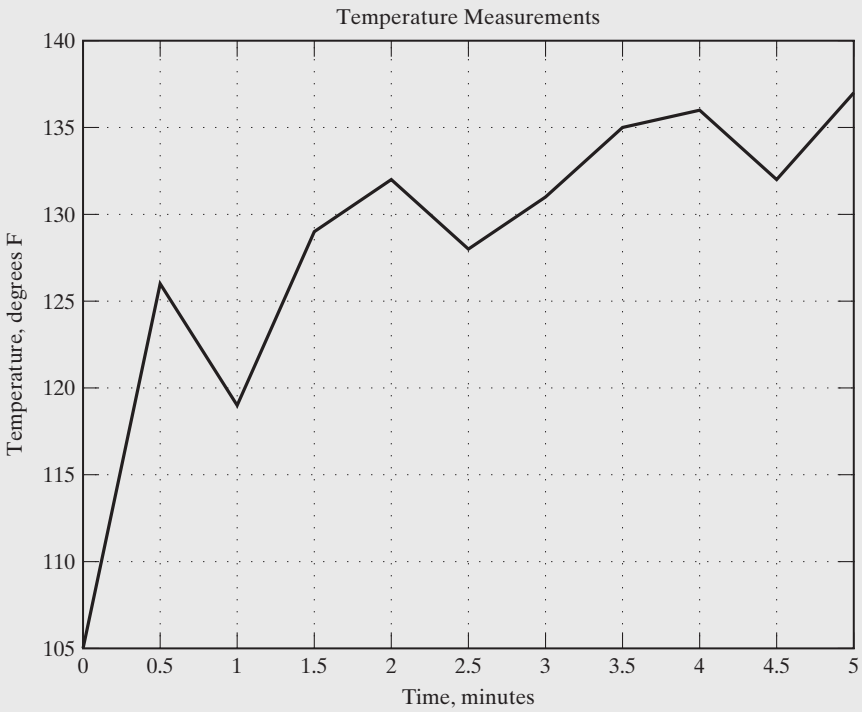


Figure 1.4
Temperatures collected in the physics experiment.

SUMMARY

Engineers solve real-world problems, and most of these solutions require the use of computing to help develop good solutions. We presented a summary of the components of a computer system from the computer hardware to computer software, and reviewed the different types of computer languages and software tools that help us develop problem solutions. We then introduced a five-step problem-solving process that we have used throughout this text. These five steps are:

1. State the problem clearly.
2. Describe the input and output information.
3. Work the problem by hand (or with a calculator) for a simple set of data.
4. Develop a MATLAB solution.
5. Test the solution with a variety of data.

KEY TERMS

algorithm
assembly language
bugs
central processing unit
compiler
compiler errors
debugging
electronic copy
hard copy
hardware
high-level languages
kernel
logic errors
machine languages
microprocessor
operating system
soft copy
software
software life cycle
software prototypes
syntax
utilities

PROBLEMS

The solutions to these problems are at the end of this chapter.

TRUE-FALSE PROBLEMS

Indicate whether the following statements are true (T) or false (F):

1. A CPU consists of an ALU, memory, and a processor. T F

2. Linking/loading is the step that prepares the object program for execution. T F

3. Sharing of hardware and software among multiple computers is possible by interconnecting them in the form of a network. T F

4. A computer program is the implementation of an algorithm. T F

5. A utility program converts a high-level language to binary. T F

6. MATLAB is both a software tool and a programming language. T F

7. Data can be communicated between internal memory and external memory through an ALU. T F

8. Spreadsheets are useful to manipulate objects graphically. T F

9. A microprocessor is a small chip onto which millions of transistors are fabricated. T F

10. A word processor allows you to enter and edit text. T F

11. MATLAB is very powerful both in computation and in visualization. T F

12. To correct logic errors we repeat only the execution step. T F

13. The compilation step identifies all the bugs in the program. T F

14. A software prototype has all the functions specified in the requirement specification document. T F

15. A computer program is a set of instructions to solve a problem. T F

16. A program is completely tested if it works for one set of data. T F

17. A thumb drive is an example of a soft copy. T F

18. A C compiler written for Microsoft Windows can also run on Linux. T F

19. Software maintenance is an insignificant part of the cost of today's software systems. T F

20. Assembly Language is not hardware specific. T F

MULTIPLE-CHOICE PROBLEMS

Circle the letter for the best answer to complete each statement:

21. Instructions and data are stored in

(a) the arithmetic logic unit (ALU).
(b) the control unit (processor).
(c) the central processing unit (CPU).
(d) the memory.
(e) the keyboard.

22. An operating system is

(a) the software that is designed by users.
(b) a convenient and efficient interface between the user and the hardware.
(c) the set of utilities that allows us to perform common operations.
(d) a set of software tools.

23. Source code is

(a) the result of compiler operations.
(b) the process of getting information from the processor.
(c) the set of instructions in a computer language that solves a specific problem.
(d) the data stored in the computer memory.
(e) the values entered through the keyboard.

24. Object code is

(a) the result of compiler operations on the source code.
(b) the process of obtaining information from the processor.
(c) a computer program.
(d) a process involving the listing of commands required to solve a specific problem.
(e) the result of the linking/loading process.

25. An algorithm refers to

(a) a step-by-step solution to solve a specific problem.
(b) a collection of instructions that the computer can understand.

(c) a code that allows us to type in text materials.
(d) stepwise refinement.
(e) a set of math equations to derive the solution to a problem.

26. MATLAB is characterized by:

(a) its powerful commands.
(b) its user-friendly interface.
(c) its ability to interface with other languages.
(d) all the above.

27. A printed copy of a document is called a:

(a) Soft copy.
(b) Manual copy.
(c) Hard copy.
(d) Electronic copy.

28. An example of software is

(a) a printer.
(b) a screen.
(c) a computer code.
(d) the memory.
(e) all of the above.

29. High-level languages are

(a) good for real-time programming.
(b) the second generation of computer languages.
(c) written in binary.
(d) written in English-like words.

30. The difference between the source program and the object program is

(a) the source program possibly contains some bugs, and the object program does not contain any bugs.
(b) the source program is the original code, and the object program is a modified code.
(c) the source program is specified in a high-level language, and the object program is specified in machine language.
(d) the object program is also a source program.
(e) the source program can be executed, and the object program cannot be executed.

31. The place to start when solving a problem is

(a) to develop an algorithm.
(b) to write the program.
(c) to compile the source program.
(d) to link to the object program.

32. _____ occur at execution time while _____ occur during translation of user program.

(a) Compile-time errors, logic errors
(b) run-time errors, logic errors
(c) logic errors, compile-time errors
(d) execution-time errors, logic errors

33. Which of the following is a Third Generation computer language:

(a) Mathematica
(b) C++
(c) MATLAB
(d) All

MATCHING PROBLEMS

Select the correct term for each of the following definitions from this list:

algorithm	natural languages
arithmetic logic unit (ALU)	network
central processing unit (CPU)	operating systems
compilation	output devices
debugging	program
grammar	software life cycle
hardware	Software maintenance
input devices	spreadsheet
Kernel	syntax
logic errors	system software
machine language	utility
memory	word processor
microprocessor	

34. The part of Operating System that act as an interface manager between computer's hardware and application programs.
35. The machinery that is part of the computer
36. The brain of the computer
37. Devices used to show the results of programs
38. Compilers and other programs that help run the computer
39. The steps to solve a problem
40. The process that converts a C program into machine language
41. A software tool designed to work with data stored in a grid or a table
42. The rules that define the punctuation and words that can be used in a program
43. The interface between the user and the hardware
44. The part of a computer that performs the mathematical computations
45. The process of removing errors from a program
46. Errors discovered during the execution of a program
47. The programs help you print files and copy files
48. A central processing unit contained in a single integrated-circuit chip
49. An important phase of the software development life cycle that involves fixing of errors identified by the users.
50. The representation of a program in binary

SOLUTIONS:

1. F
2. T
3. T
4. T
5. F
6. T
7. F
8. F
9. T
10. T
11. T
12. F
13. F
14. T
15. T
16. F
17. F
18. F
19. F
20. T
21. (d)
22. (b)
23. (c)
24. (a)
25. (a)
26. (d)
27. (c)
28. (c)
29. (d)
30. (c)
31. (a)
32. (c)
33. (b)
34. Kernel
35. hardware
36. CPU
37. output devices
38. system software
39. algorithm
40. compilation
41. spreadsheet
42. syntax or grammar
43. operating system
44. ALU
45. debugging
46. logic errors
47. utilities
48. microprocessor
49. software maintenance
50. machine language

CHAPTER

2 Getting Started with MATLAB

Objectives

After reading this chapter, you should be able to

- understand the MATLAB screen layout, windows, and interactive environments,
- initialize and use scalars, vectors, and matrices in computations,
- write simple programs using MATLAB, and
- create and use script M-files.

ENGINEERING ACHIEVEMENT: WIND TUNNELS

Wind tunnels are test chambers built to generate precise wind speeds. Accurate scale models of new aircraft and missiles can be mounted on force-measuring supports in the test chamber, and then measurements of the forces acting on the models can be made at many different wind speeds and angles of the models relative to the wind direction. Some wind tunnels can operate at hypersonic velocities, generating wind speeds of thousands of miles per hour. The sizes of wind tunnel test sections vary from a few inches across to sizes large enough to accommodate a fighter jet. At the completion of a wind tunnel test series, many sets of data have been collected that can be used to determine the lift, drag, and other aerodynamic performance characteristics of a new aircraft at its various operating speeds and positions. Wind tunnels are also used to test the performance of sports equipment like composite skis, snowboards, bicycles, and racing cars. In this chapter, we give examples of using MATLAB to analyze wind tunnel results.

2.1 INTRODUCTION TO MATLAB AND MATLAB WINDOWS

MATLAB is one of a number of commercially available, sophisticated mathematical computation tools, such as Mathematica and MathCad. Despite what their proponents may claim, none of these tools is "the best." They all have strengths and weaknesses.

Each will allow you to perform basic mathematical computations, but they differ in the ways that they handle symbolic calculations and more complicated mathematical processes. MATLAB excels at computations involving matrices. In fact, its name, **MATLAB**, is short for **Mat**rix **Lab**oratory. At a very basic level, you can think of these programs as sophisticated, computer-based calculators. They can perform the same functions as your scientific calculator, but they can also do much more. In many engineering programs, students are learning to use mathematical computational tools like MATLAB, in addition to also learning a high-level language such as JAVA, C, or C++. This then gives you the option of choosing the right tool or language for the problem that you are solving.

Today's MATLAB has capabilities far beyond the original MATLAB and is an interactive system and programming language for general scientific and technical computation. Because MATLAB commands are similar to the way that we express engineering steps in mathematics, writing computer solutions in MATLAB can be much quicker than writing solutions in a high-level language. It is important to understand when to use a computational program such as MATLAB and when to use a general purpose, high-level programming language. MATLAB excels at numerical calculations, especially matrix calculations, and graphics. Usually, high-level programs do not offer easy access to graphing. The primary area of overlap between MATLAB and high-level programs is in "number crunching"—programs that require repetitive calculations or processing of large quantities of data. Both MATLAB and high-level languages are good at processing numbers. It is usually easier to write a "number crunching" program in MATLAB, but it usually executes faster in C or C++. The one exception to this rule is with matrices. Because MATLAB is optimized for matrices, if a problem can be formulated with a matrix solution, MATLAB executes substantially faster than a similar program in a high-level language.

HINT

A number of examples are presented in this text. We encourage you to type the example problems into MATLAB as you read the book, and observe the results. You may think that some of the examples are too simple to type in yourself—that just reading the material is sufficient. However, you will remember the material much better if you both read it and type it.

To begin MATLAB, use your mouse to click on the MATLAB icon on the desktop or choose it from the list of Applications on your computer. To exit MATLAB, use the close icon (x) from the upper right-hand corner of the screen for a PC or use the red circle at the upper left-hand corner of the screen for an Apple computer. These are essentially the only differences that you will see between PC's and Apple computers. The MATLAB screens and output will be the same. You should see the MATLAB **prompt >>** (or **EDU** ≫ if you are using the Student Edition) in the middle of the screen which tells you that MATLAB is waiting for you to enter a command. To exit MATLAB, type **`quit`** or **`exit`** at the MATLAB prompt, or select the close icon (x) from the top of the screen.

MATLAB uses display windows. The default view shown in Figure 2.1 includes a large command window in the center, the current folder window on the left, and