Preface

Technical computing environments

Computers are now an essential component of research and education in science, engineering, and almost all technical fields. Experimentalists routinely use computers to collect and analyze data while theoreticians use computers to manipulate equations numerically and symbolically. For both, computer simulation and modeling have become indispensable investigative tools. In response, technical education has changed to incorporate the use of computers into the curriculum both as a topic and as a medium for the presentation of technical material.

Advances in computer hardware have progressed to the point where researchers and students now have spectacularly powerful machinery at their fingertips. It is not uncommon for a first-year engineering student to have their own laptop which contains more raw computing power than most researchers could even dream of five years ago.

Oftentimes, the spectacular gains in processing power have not been met with similar advances in software. It often seems as if the power and utility of software are inversely proportional to their ease of use. Fortunately, for those who do technical computing, *Mathematica* has long provided a fully integrated environment, including a programming language suitable for individuals who are not full-time programmers but who need to create programs to carry out their work. It includes:

- Built-in mathematical and graphical capabilities that are both powerful and flexible;
- A programming language that can be used to extend its capabilities virtually without limit. That language is interactive, has the capability to perform both numeric and symbolic manipulation, makes broad use of pattern matching, and supports a functional style of programming favored by many computer scientists (while incorporating constructs for more conventional programming styles).
- Extensive online help facilities that make it easy to learn about built-in functions and get their syntax correct.

- The ability to connect *Mathematica* to other computing environments and other languages.

- A programmable interface in which all of the above elements are seamlessly integrated.

In this book, we focus on the *Mathematica* programming language. While there are many books, including the reference manual by Stephen Wolfram, *The Mathematica Book* (Wolfram 2003), that discuss various aspects of *Mathematica*, there has been a need for a text explaining how to use the underlying programming language so that *Mathematica*'s capabilities can be fully utilized. This book explains *Mathematica*'s programming language for the beginning programmer, first discussing the syntax and structure of the language and then focusing on how to use different programming styles to solve problems.

The audience for this book

This book was written for two distinct groups of readers:

- Users who are familiar with *Mathematica* but wish to deepen their understanding of the program and go beyond the most basic kinds of computations. This book introduces you to the syntax and structure of the *Mathematica* programming language, thus helping you to see the bigger picture of how *Mathematica* works.

- Those who want to learn to program and would like a friendly and useful language to start with. Most low-level languages, like C and Fortran, require many lines of complicated code in order to do something interesting, while many high-level languages, like Lisp, are a bit esoteric and difficult to master. Each of these traditional languages require an edit-compile-run cycle every time you wish to check your program. *Mathematica*, on the other hand, has a more natural syntax and provides very high-level built-in operations with which you can do a lot of interesting things right away. Since it is an interactive environment, you can prototype and test your code immediately, all in the same environment.

If you already have programming experience in one of the more traditional languages, you will often write *Mathematica* programs in a procedural style, producing code that looks like "Fortran in *Mathematica*" for example. By understanding how *Mathematica* programs can be constructed using the functional and rule-based styles of programming, much simpler and more efficient code can often be written.

Changes in the new edition

Since the last edition of this book was published, significant changes have occurred both in *Mathematica* and in its use throughout the world. Several new major versions of *Mathematica* have been released (as of the publication date of this book, *Mathematica* 5.1 is the current version) and, with them, quite a few new functions and features, and some changes to existing functions. This new edition of *An Introduction to Programming with Mathematica* has kept pace, including information and examples throughout the book on the most important new features and functions.

Some of the new functions which are now discussed in this book include: `Array`, `Depth`, `ArrayPlot`, `ArrayQ`, `ArrayRules`, `BitAnd`, `BitOr`, `BitXor`, `ButtonBox`, `ButtonFunction`, `ColumnAlignments`, `DisplayForm`, `Element`, `Evaluation`, `Monitor`, `Export`, `FindFit`, `GridBox`, `Import`, `MakeBoxes`, `MakeExpression`, `Mean`, `NestWhile`, `NestWhileList`, `Norm`. `NotebookGet`, `NotebookPut`, `NumericQ`, `PadLeft`, `PadRight`, `Pick`, `PieceWise`, `RegularExpression`, `RowAlignments`, `RowBox`, `SelectionEvaluate`, `SelectionMove`, `Sequence`, `SparseArray`, `Split`, `StepMonitor`, `StringExpression`, `StringCases`, `StyleForm`, `Subsets`, `ToFileName`, `TraditionalForm`.

We have reorganized the structure of the chapters to make it easier to find information on broad topics of interest to the reader. There is a new chapter (Chapter 2) devoted specifically to the syntax and structure of the *Mathematica* language that has culled material from several chapters of the earlier edition. The old chapters on iteration and conditional definitions have been incorporated into a new chapter on procedural programming. Rule-based programming now gets its own chapter. The chapter on numerics has been extensively rewritten to take into account the many changes to numerics over the past several versions of *Mathematica*. The chapter on applications has also been reorganized and rewritten to better explain application development. And we have added an entirely new chapter on front end programming, something that was not feasible prior to *Mathematica* 3. See the section below, "Book Contents," for details on each chapter.

Throughout this new edition, many new examples have been worked into the text, in particular in Chapters 4, 5, 9, and 11. Exercises have been expanded over the previous edition, as have the solutions to the exercises. In addition to the printed solutions to exercises at the end of this book, they are also available as a *Mathematica* notebook online in several locations (see below).

Readers of the first two editions found a handful of mistakes, which have been corrected in this edition. We welcome further communication from our readers.

How to use this book

There are several ways in which this book should be useful:

- As a primary text in an introductory course on programming. Perhaps the most obvious use would be in an introductory computer science course for students in science, engineering, or mathematics, who would solve technical problems of interest to them, while at the same time learning to program. The *Mathematica* programming language supports many programming styles that are present in other programming languages, so programming skills learned using *Mathematica* can be readily transferred to these languages.

- As a supplemental text in a course in which *Mathematica* is being used as a tool for studying another technical subject. In these kinds of courses, the principles of *Mathematica* programming can be introduced as needed during the course, although we have found that a structured introduction to *Mathematica* programming at the beginning of the semester works best.

- As a self-study book, particularly for *Mathematica* users who need to use its programming capabilities more fully or who are interested in understanding how the *Mathematica* programming language works.

As educators, we strongly believe in the axiom that one learns by doing. Hence we have not only included hundreds of exercises to this book, but have tried hard to make sure that the exercises are interesting and instructive. You will learn how to program in the *Mathematica* language best by trying to solve problems. We strongly encourage you to try the examples in the text on your own, modifying them as you see fit, but also you should try to do as many of the exercises at the end of each section as you can. Quite a few concepts and features are explained in the exercises and their solutions so it is well worthwhile spending some time on those aspects of the book.

When you work with *Mathematica*, you will generally find that you work with several different programming styles, deal with numerical issues as well as symbolics, crunch numbers but also use graphics, etc. Although topics such as functional programming, procedural programming, and rule-based programming are presented in separate chapters, you should not view such a division as ironclad. In fact, it is merely a convenience to enable us to discuss the various topics that make up the body of this book. We have tried to make the materials self-contained and have everything flow logically from beginning to end, but in some instances, this is quite difficult and, in fact, arbitrary. After digesting the first two

or three chapters, you should feel free to jump around from chapter to chapter as you work through the materials in this book.

Conventions

All input and output in this book appear in a different font from the regular text. This is true for examples that appear in the middle of text, such as `Expand[(a + b)^4]`, as well as displayed *Mathematica* code. So, for example, lines of input (what you type at your computer) appear as

```
In[1]:= 3 + 5
```

whereas all output (what *Mathematica* prints on your computer screen) appears in a slightly lighter font than the input.

```
Out[1]= 8
```

You do not have to type the prompts *In[1]:=* or *Out[1]=*; *Mathematica* will do that for you automatically.

All of the programs that are defined in this book can be located in the index under the heading **Programs**. So, for example, the function `runEncode` defined in Section 7.3 is listed in the index under **Programs**, `runEncode`.

Book contents

We should perhaps start with what is *not* in this book, namely, a complete list and explanation of the hundreds of operations built into *Mathematica*. For that list, the indispensable reference is *The Mathematica Book*, (Wolfram, 2003).

This book is also not a complete or advanced treatment of programming in *Mathematica*, nor is it a book about using *Mathematica* to solve problems from one particular discipline. So, for example, the reader will not find a discussion of object-oriented programming in *Mathematica*, nor of databases, nor a formal discussion of data types. Fortunately, there are other excellent sources of information on such topics, including books such as *The Mathematica Programmer I* and *II*, or *Computer Science with Mathematica: Theory and Practice for Science, Mathematics, and Engineering*, all by Roman Maeder.

This book is about teaching you how to program with *Mathematica*. We assume that you have little, if any, programming experience (though you may already be a *Mathematica* user), and we take you step by step through the various programming styles that are available in *Mathematica* and explain when to use what style.

The chapter structure is as follows:

- Chapter 1 An Introduction to *Mathematica*. For the reader who has little or no knowledge of *Mathematica*, we have included a brief overview of some of its built-in capabilities, including graphics and numerical and symbolic computations. In addition, this chapter contains some of the fundamentals of *Mathematica*, such as how to use the *Mathematica* interface and how to enter expressions. Experienced *Mathematica* users should feel free to skim this material.

- Chapter 2 The *Mathematica* Language. This chapter gives an introduction to the syntax and structure of the *Mathematica* programming language. It introduces the basic construct in the *Mathematica* language, an expression, and then discusses some of the building blocks that you will use over and over again in constructing your programs.

- Chapter 3 Lists. The heart of the book begins here, with a discussion of this most important of data types. Aside from numbers, no other data type is more useful in programming. We describe the use of built-in functions to create and manipulate lists. This chapter also discusses strings and characters as their manipulation is so similar to that for lists.

- Chapter 4 Functional Programming. This chapter introduces functional programming, a style quite distinct from what is available in traditional languages. We discuss nested function calls, iterating functions, and introduce some of the more important higher-order functions, such as `Map`, `Apply`, and `Thread`. A section on pure functions introduces this important construct borrowed from the lambda calculus. The chapter concludes with a section demonstrating how to construct "one-liners," short programs that solve a particular problem in one line.

- Chapter 5 Procedural Programming. Procedures can be thought of as recipes. They spell out a sequence of steps to follow. They often include control structures that determine what action to take depending upon some property of an input. This programming style is more characteristic of conventional languages like C than it is of *Mathematica*, but is still important. This chapter introduces the concepts of loops, iteration, flow control, and conditional definitions, using some traditional problems such as Newton's method and the Sieve of Eratosthenes to demonstrate these concepts.

- Chapter 6 Rule-Based Programming. This chapter, new to the third edition of this book, explains how rules can be used to transform expressions. Blanks, pattern matching, string patterns, and alternatives are discussed extensively and some classical problems from computer science are examined using rule-based approaches: encoding and sorting.

- Chapter 7 Recursion. This method of programming, in which a function is defined in terms of itself, is heavily used in mathematics and computer science and has a natural implementation in *Mathematica*. For those with little experience programming recursively, we give a gentle introduction to thinking recursively and then use several well-known problems such as merge sort, Gaussian elimination, tree representation, and Huffman encoding to implement the ideas in this chapter. The chapter also includes discussions of dynamic programming (or caching).

- Chapter 8 Numerics. *Mathematica* contains a variety of types of numbers, both exact and approximate. This chapter discusses working with exact vs. inexact numbers and machine-precision vs. high-precision numbers. Efficiency issues are also presented including a discussion of packed arrays and sparse arrays, two data formats that are optimized for both speed and memory. The chapter concludes with a discussion of how to write programs that best take advantage of the numerical capabilities in *Mathematica*.

- Chapter 9 Graphics Programming. This chapter introduces the basic concepts of *Mathematica* graphics. It uses some of the techniques of the previous chapters to create graphics-based programs and to solve problems that are inherently graphical in nature. It also demonstrates how to include options in your programs.

- Chapter 10 Front End Programming. The front end is the graphical user interface to *Mathematica*. Beginning in Version 3, many aspects of the front end became programmable and this functionality was further extended in Versions 4 and 5. This new chapter introduces the structure of the objects in the front end that can be manipulated using the techniques discussed in earlier chapters. It includes sections on notebooks and cells, cell data types, box structures, and buttons (a special kind of box).

- Chapter 11 Examples and Applications. *Mathematica* is used to solve larger-scale programming problems. Included are an application to manipulate and visualize data, as well as *Mathematica* implementations of the Game of Life using functional

and rule-based programming, random walks, and finally the creation of a mini picture-description language. Options and default values and documentation implementations are also discussed.

- Chapter 12 Writing Packages. The last chapter introduces the method of organizing libraries of *Mathematica* functions into convenient units called packages. Discussion of options, error-trapping, and messages are included.

Supplemental packages and solutions

Materials for this book have been made available in electronic form. *Mathematica* notebooks containing most of the examples, exercises, and solutions are included in an archive appropriate for several platforms: IPM3.zip for Windows, IPM3.tgz for Unix, and IPM3.hqx for Macintosh OS X. The file names correspond directly to the chapter structure. So, for example, the notebook for the first chapter is 01Introduction.nb.

The archive also contains files with *Mathematica* programs that can be loaded into your *Mathematica* sessions as needed. These files (normally called packages), have a filename extension ending with `.m`.

Archives for this book have been placed in two locations:

- Cambridge University Press website: www.cambridge.org/0521846781
- Wolfram Research website: library.wolfram.com/infocenter/Books/5169/

The recommended way to use the notebooks and packages is to first unpack the archive on your computer. This will leave you with two directories/folders, one called IPM3 and the other RandomWalks. Move each of these folders and their contents inside one of the *Mathematica* Applications directories on your computer. The recommended location is inside either `$BaseDirectory` or `$UserBaseDirectory` as files in these two directories will continue to be found even after you upgrade *Mathematica*.

Here is the recommended location for where to install applications. The output will look different on different operating systems; the following was run in Windows XP.

```
ToFileName[{$BaseDirectory, "Applications"}]
```

```
C:\Documents and Settings\All Users\
  Application Data\Mathematica\Applications\
```

Once you have installed the directories in the above location, start up *Mathematica* and, from the Help menu, select Rebuild Help Index. Once this is done, you will find the notebooks in the Help Browser by choosing the Add-ons & Links category.

To load packages from the book, you need only to specify the directory and package name. For example, this loads the package `RandomWalks.m`:

```
<< IPM3`RandomWalks`
```

Further information

While the basic aspects of *Mathematica* programming are discussed in this book, there are a great many more things that can be said about *Mathematica* and the *Mathematica* programming language. The most comprehensive reference source is the manual that comes with each copy of the software: *The Mathematica Book* (Wolfram, 2003).

The *Mathematica* Information Center (library.wolfram.com/infocenter) contains thousands of programming examples and notebooks from all areas of science and engineering. It is organized by collection (articles, courseware, demos, etc.), by subject, or you can simply search on any phrase that you wish.

An Internet newsgroup exists that is devoted to all aspects of *Mathematica*: `comp.soft-sys.math.mathematica`. This moderated newsgroup has been in existence for many years and users have come to rely on the collective wisdom of the hundreds of contributors from around the world who respond to any question that is posed. An archive going back to 1989 exists and can easily be browsed or searched in a web browser by going to forums.wolfram.com/mathgroup/.

The Mathematica Journal is a quarterly journal that publishes articles about all aspects of *Mathematica*. For subscription information, or to view issues online, go to the website www.mathematica-journal.com.

Finally, there are now over 300 books about *Mathematica* or that use *Mathematica* to teach a subject. Some of these can be found in the bibliography at the end of this text. See store.wolfram.com/catalog/books/ for a complete up-to-date publications list.

Colophon

The third edition of this book was produced from original *Mathematica* notebooks. *Mathematica* 5 and 5.1 were used throughout. A *Mathematica* style sheet was customized containing page layout information consistent with the page layout as specified by a book designer. The *Mathematica* notebooks were output to PostScript files and then distilled to PDF files which were sent to the printer electronically.

Acknowledgments

Many people have contributed to our understanding of the *Mathematica* programming language over the years. At Wolfram Research these include Lou D'Andria, Harry Calkins, Andy Hunt, Rob Knapp, Dan Lichtblau, John Novak, Robby Villegas, and Dave Withoff. Also Roman Maeder has answered several esoteric questions about the patterns and the ordering of rules in *Mathematica*.

Thanks to Dana Scott for permission to use his material on sorting (Chapter 6) and Klaus Sutner for permission to use his material on encryption (also in Chapter 6) as well as interesting discussions on the pattern matcher in *Mathematica*.

Paul Wellin would like to thank his wife Sheri for her support, understanding, and sense of humor throughout this project.

Richard J. Gaylord would like to thank Shawn Sheridan who was his guru for both *Mathematica* and the Macintosh.

Samuel Kamin would like to acknowledge the Computer Science Department of the University of Illinois for its excellent working environment and the support of his colleagues there. Above all, he would like to thank Judy and Rebecca for their love and patience.

Paul R. Wellin
wellin@wolfram.com

Richard J. Gaylord
gaylord@uiuc.edu

Samuel N. Kamin
kamin@cs.uiuc.edu

Ocotober 2004