

to *The Mathematica GuideBooks* *Mathematica* Concepts—Programming Examples—Scientific Applications

0.1 Overview

■ 0.1.1 Content Summaries

The *Mathematica GuideBooks* are published as four independent books: *The Mathematica GuideBook to Programming*, *The Mathematica GuideBook to Graphics*, *The Mathematica GuideBook to Numerics*, and *The Mathematica GuideBook to Symbolics*.

■ The Programming volume deals with the structure of *Mathematica* expressions and with *Mathematica* as a programming language. This volume includes the discussion of the hierarchical construction of all *Mathematica* objects out of symbolic expressions (all of the form *head[argument]*), the ultimate building blocks of expressions (numbers, symbols, and strings), the definition of functions, the application of rules, the recognition of patterns and their efficient application, the order of evaluation, program flows and program structure, the manipulation of lists (the universal container for *Mathematica* expressions of all kinds), as well as a number of topics specific to the *Mathematica* programming language. Various programming styles, especially *Mathematica*'s powerful functional programming constructs, are covered in detail.

■ The Graphics volume deals with *Mathematica*'s two-dimensional (2D) and three-dimensional (3D) graphics. The chapters of this volume give a detailed treatment on how to create images from graphics primitives, such as points, lines, and polygons. This volume also covers graphically displaying functions given either analytically or in discrete form. A number of images from the *Mathematica* Graphics Gallery are also reconstructed. Also discussed is the generation of pleasing scientific visualizations of functions, formulas, and algorithms. A variety of such examples are given.

■ The Numerics volume deals with *Mathematica*'s numerical mathematics capabilities—the indispensable sledgehammer tools for dealing with virtually any “real life” problem. The arithmetic types (fast machine, exact integer, and rational, verified high-precision, and interval arithmetic) are carefully analyzed. Fundamental numerical operations, such as compilation of programs, numerical Fourier transforms, minimization, numerical solution of equations, ordinary/partial differential equations are analyzed in detail and are applied to a large number of examples in the main text and in the solutions to the exercises.

■ The Symbolics volume deals with *Mathematica*'s symbolic mathematical capabilities—the real heart of *Mathematica* and the ingredient of the *Mathematica* software system that makes it so unique and powerful. Structural and mathematical operations on systems of polynomials are fundamental to many symbolic calculations and are covered in detail. The solution of equations and differential equations, as well as the classical calculus operations are exhaustively treated. In addition, this

volume discusses and employs the classical orthogonal polynomials and special functions of mathematical physics. To demonstrate the symbolic mathematics power, a variety of problems from mathematics and physics are discussed.

The four *GuideBooks* contain about 25,000 *Mathematica* inputs, representing more than 70,000 lines of commented *Mathematica* code. (For the reader already familiar with *Mathematica*, here is a more precise measure: The `LeafCount` of all inputs would be about 800,000 when collected in a list.) The *GuideBooks* also have more than 4,000 graphics, 100 animations, 8,000 references, and 1,000 exercises. More than 10,000 hyperlinked index entries and hundreds of hyperlinks from the overview sections connect all parts in a convenient way. The evaluated notebooks of all four volumes have a cumulative file size of about 10 GB. Although these numbers may sound large, the *Mathematica GuideBooks* actually cover only a portion of *Mathematica*'s functionality and features and give only a glimpse into the possibilities *Mathematica* offers to generate graphics, solve problems, model systems, and discover new identities, relations, and algorithms. The *Mathematica* code is explained in detail throughout all chapters. More than 10,000 comments are scattered throughout all inputs and code fragments.

■ 0.1.2 Relation of the Four Volumes

The four volumes of the *GuideBooks* are basically independent, in the sense that readers familiar with *Mathematica* programming can read any of the other three volumes. But a solid working knowledge of the main topics discussed in *The Mathematica GuideBook to Programming*—symbolic expressions, pure functions, rules and replacements, list manipulations—is required for the Graphics, Numerics, and Symbolics volumes. Compared to these three volumes, the Programming volume might appear to be a bit “dry”. But similar to learning a foreign language, before being rewarded with the beauty of novels or a poem, one has to sweat and study. The whole suite of graphical capabilities and all of the mathematical knowledge in *Mathematica* are accessed and applied through lists, patterns, rules, and pure functions, the material discussed in the Programming volume.

Naturally, graphics are the center of attention of the *The Mathematica GuideBook to Graphics*. While in the Programming volume some plotting and graphics for visualization are used, graphics are not crucial for the Programming volume. The reader can safely skip the corresponding inputs to follow the main programming threads. The Numerics and Symbolics volumes, on the other hand, make heavy use of the graphics knowledge acquired in the Graphics volume. Hence, the prerequisites for the Numerics and Symbolics volumes are a good knowledge of *Mathematica*'s programming language and of its graphics system.

The Programming volume contains only a few percent of all graphics, the Graphics volume contains about two-thirds, and the Numerics and Symbolics volume, about one-third of the overall 4,000+ graphics. The Programming and Graphics volume use some mathematical commands, but they restrict the use to a relatively small number (especially `Expand`, `Factor`, `Integrate`, `Solve`). And the use of the function `N` for numericalization is unavoidable for virtually any “real life” application of *Mathematica*. The last functions allow us to treat some mathematically not uninteresting examples in the Programming and Graphics volumes. In addition to putting these functions to work for nontrivial problems, a detailed discussion of the mathematics functions of *Mathematica* takes place exclusively in the Numerics and Symbolics volumes.

The Programming and Graphics volumes contain a moderate amount of mathematics in the examples and exercises, and focus on programming and graphics issues. The Numerics and Symbolics volumes contain a substantially larger amount of mathematics.

Although printed as four books, the fourteen individual chapters (six in the Programming volume, three in the Graphics volume, two in the Numerics volume, and three in the Symbolics volume) of the *Mathematica GuideBooks* form one organic whole, and the author recommends a strictly sequential reading, starting from Chapter 1 of the Programming volume and ending with Chapter 3 of the Symbolics volume for gaining the maximum benefit. The electronic component of each book contains the text and inputs from all the four *GuideBooks*, together with a comprehensive hyperlinked index. The four volumes refer frequently to one another.

■ 0.1.3 Chapter Structure

A rough outline of the content of a chapter is the following:

- The main body discusses the *Mathematica* functions belonging to the chapter subject, as well their options and attributes. Generically, the author has attempted to introduce the functions in a “natural order”. But surely one cannot be axiomatic with respect to the order. (Such an order of the functions is not unique, and the author intentionally has “spread out” the introduction of various *Mathematica* functions across the four volumes.) With the introduction of a function, some small examples of how to use the functions and comparisons of this function with related ones are given. These examples typically (with the exception of some visualizations in the Programming volume) incorporate functions already discussed. The last section of a chapter often gives a larger example that makes heavy use of the functions discussed in the chapter.
- A programmatically constructed overview of each chapter functions follows. The functions listed in this section are hyper-linked to their attributes and options, as well as to the corresponding reference guide entries of *The Mathematica Book*.
- A set of exercises and potential solutions follow. Because learning *Mathematica* through examples is very efficient, the proposed solutions are quite detailed and form up to 50% of the material of a chapter.
- References end the chapter.

Note that the first few chapters of the Programming volume deviate slightly from this structure. Chapter 1 of the Programming volume gives a general overview of the kind of problems dealt with in the four *GuideBooks*. The second, third, and fourth chapters of the Programming volume introduce the basics of programming in *Mathematica*. Starting with Chapters 5 of the Programming volume and throughout the Graphics, Numerics, and Symbolics volume, the above-described structure applies.

In the 14 chapters of the *GuideBooks* the author has chosen a “we” style for the discussions of how to proceed in constructing programs and carrying out calculations to include the reader tightly.

■ 0.1.4 Code Presentation Style

The typical style of a unit of the main part of a chapter is: Define a new function, discuss its arguments, options, and attributes, and then give examples of its usage. The examples are virtually always *Mathematica* inputs and outputs. The majority of inputs is in `InputForm` are the notebooks. On occasion `StandardForm` is also used. Although `StandardForm` mimics classical mathematics notation and makes short inputs more readable, for “program-like” inputs, `InputForm` is typically more readable and easier and more natural to align. For the outputs, `StandardForm` is used by default and occasionally the author has resorted to `InputForm` or `FullForm` to expose digits of numbers and to `TraditionalForm` for some formulas. Outputs are mostly not programs, but nearly always “results” (often mathematical expressions, formulas, identities, or lists of numbers rather than program constructs). The world of *Mathematica* users is divided into three groups, and each of them has a nearly religious opinion on how to format *Mathematica* code [1★], [2★]. The author follows the `InputForm` cult(ure) and hopes that the *Mathematica* users who do everything in either `StandardForm` or `TraditionalForm` will bear with him. If the reader really wants to see all code in either `StandardForm` or `TraditionalForm`, this can easily be done with the `Convert To` item from the `Cell` menu. (Note that the relation between `InputForm` and `StandardForm` is not symmetric. The `InputForm` cells of this book have been line-broken and aligned by hand. Transforming them into `StandardForm` or `TraditionalForm` cells works well because one typically does not line-break manually and align *Mathematica* code in these cell types. But converting `StandardForm` or `TraditionalForm` cells into `InputForm` cells results in much less pleasing results.)

In the inputs, special typeset symbols for *Mathematica* functions are typically avoided because they are not monospaced. But the author does occasionally compromise and use Greek, script, Gothic, and doublestruck characters.

In a book about a programming language, two other issues come always up: indentation and placement of the code.

- The code of the *GuideBooks* is largely consistently formatted and indented. There are no strict guidelines or even rules on how to format and indent *Mathematica* code. The author hopes the reader will find the abook's formatting style readable. It is a compromise between readability (mental parsability) and space conservation, so that the printed version of the *Mathematica GuideBook* matches closely the electronic version.

- Because of the large number of examples, a rather imposing amount of *Mathematica* code is presented. Should this code be present only on the disk, or also in the printed book? If it is in the printed book, should it be at the position where the code is used or at the end of the book in an appendix? Many authors of *Mathematica* articles and books have strong opinions on this subject. Because the main emphasis of the *Mathematica GuideBooks* is on *solving problems with Mathematica* and not on the actual problems, the *GuideBooks* give all of the code at the point where it is needed in the printed book, rather than "hiding" it in packages and appendices. In addition to being more straightforward to read and conveniently allowing us to refer to elements of the code pieces, this placement makes the correspondence between the printed book and the notebooks close to 1:1, and so working back and forth between the printed book and the notebooks is as straightforward as possible.

0.2 Requirements

■ 0.2.1 Hardware and Software

Throughout the *GuideBooks*, it is assumed that the reader has access to a computer running a current version of *Mathematica* (version 4.0 or newer). For readers without access to a licensed copy of *Mathematica*, it is possible to view all of the material on the disk using *MathReader*. (*MathReader* is downloadable from www.wolfram.com/mathreader.)

The files of the *GuideBooks* are relatively large, altogether more than 10 GB. This is also the amount of hard disk space needed to store uncompressed versions of the notebooks. To view the notebooks comfortably, the reader's computer needs 64 MB RAM; to evaluate the evaluation units of the notebooks 512 MB RAM or more is recommended.

In the *GuideBooks*, a large number of animations are generated. Although they need more memory than single pictures, they are easy to create, to animate, and to store on typical year-2003 hardware, and they provide a lot of joy.

■ 0.2.2 Reader Prerequisites

Although prior *Mathematica* knowledge is not needed to read *The Mathematica GuideBook to Programming*, it is assumed that the reader is familiar with basic actions in the *Mathematica* front end, including entering Greek characters using the keyboard, copying and pasting cells, and so on. Freely available tutorials on these (and other) subjects can be found at <http://library.wolfram.com>.

For a complete understanding of most of the *GuideBooks* examples, it is desirable to have a background in mathematics, science, or engineering at about the bachelor's level or above. Familiarity with mechanics and electrodynamics is assumed. Some examples and exercises are more specialized, for instance, from quantum mechanics, finite element analysis, statistical mechanics, solid state physics, number theory, and other areas. But the *GuideBooks* avoid very advanced (but tempting) topics such as renormalization groups [6★], parquet approximations [26★], and modular moonshines [14★]. (Although *Mathematica* can deal with such topics, they do not fit the character of the *Mathematica GuideBooks* but rather the one of a *Mathematica Topographical Atlas* [a monumental work to be carried out by the *Mathematica*-Bourbakians of the 21st century]).

Each scientific application discussed has a set of references. The references should easily give the reader both an overview of

0.3 What the GuideBooks Are and What They Are Not

■ 0.3.1 Doing Computer Mathematics

As discussed in the Preface, the main goal of the *GuideBooks* is to demonstrate, showcase, teach, and exemplify scientific problem solving with *Mathematica*. An important step in achieving this goal is the discussion of *Mathematica* functions that allow readers to become fluent in programming when creating complicated graphics or solving scientific problems. This again means that the reader must become familiar with the most important programming, graphics, numerics, and symbolics functions, their arguments, options, attributes, and a few of their time and space complexities. And the reader must know which functions to use in each situation.

The *GuideBooks* treat only aspects of *Mathematica* that are ultimately related to “doing mathematics”. This means that the *GuideBooks* focus on the functionalities of the kernel rather than on those of the front end. The knowledge required to use the front end to work with the notebooks can easily be gained by reading the corresponding chapters of the online documentation of *Mathematica*. Some of the subjects that are treated either lightly or not at all in the *GuideBooks* include the basic use of *Mathematica* (starting the program, features, and special properties of the notebook front end [16★]), typesetting, the preparation of packages, external file operations, the communication of *Mathematica* with other programs via *MathLink*, special formatting and string manipulations, computer- and operating system-specific operations, audio generation, and commands available in various packages. “Packages” includes both, those distributed with *Mathematica* as well as those available from *MathSource* (<http://library.wolfram.com/database/MathSource>) and commercial sources, such as *MathTensor* for doing general relativity calculations (<http://smc.vnet.net/MathTensor.html>) or *FeynCalc* for doing high-energy physics calculations (<http://www.feyncalc.com>). This means, in particular, that probability and statistical calculations are barely touched on because most of the relevant commands are contained in the packages. The *GuideBooks* make little or no mention of the machine-dependent possibilities offered by the various *Mathematica* implementations. For this information, see the *Mathematica* documentation.

Mathematical and physical remarks introduce certain subjects and formulas to make the associated *Mathematica* implementations easier to understand. These remarks are not meant to provide a deep understanding of the (sometimes complicated) physical model or underlying mathematics; some of these remarks intentionally oversimplify matters.

The reader should examine all *Mathematica* inputs and outputs carefully. Sometimes, the inputs and outputs illustrate little-known or seldom-used aspects of *Mathematica* commands. Moreover, for the efficient use of *Mathematica*, it is very important to understand the possibilities and limits of the built-in commands. Many commands in *Mathematica* allow different numbers of arguments. When a given command is called with fewer than the maximum number of arguments, an internal (or user-defined) default value is used for the missing arguments. For most of the commands, the maximum number of arguments and default values are discussed.

When solving problems, the *GuideBooks* generically use a “straightforward” approach. This means they are not using particularly clever tricks to solve problems, but rather direct, possibly computationally more expensive, approaches. (From time to time, the *GuideBooks* even make use of a “brute force” approach.) The motivation is that when solving new “real life” problems a reader encounters in daily work, the “right mathematical trick” is seldom at hand. Nevertheless, the reader can more often than not rely on *Mathematica* being powerful enough to often succeed in using a straightforward approach. But attention is paid to *Mathematica*-specific issues to find time- and memory-efficient implementations—something that should be taken into account for any larger program.

As already mentioned, all larger pieces of code in this book have comments explaining the individual steps carried out in the calculations. Many smaller pieces of code have comments when needed to expedite the understanding of how they work. This enables the reader to easily change and adapt the code pieces. Sometimes, when the translation from traditional mathematics

into *Mathematica* is trivial, or when the author wants to emphasize certain aspects of the code, we let the code “speak for itself”. While paying attention to efficiency, the *GuideBooks* only occasionally go into the computational complexity ([8★], [39★], and [7★]) of the given implementations. The implementation of very large, complicated suites of algorithms is not the purpose of the *GuideBooks*. The *Mathematica* packages included with *Mathematica* and the ones at *MathSource* (<http://library.wolfram.com/database/MathSource>) offer a rich variety of self-study material on building large programs. Most general guidelines for writing code for scientific calculations (like descriptive variable names and modularity of code; see, e.g., [19★] for a review) apply also to *Mathematica* programs.

The programs given in a chapter typically make use of *Mathematica* functions discussed in earlier chapters. Using commands from later chapters would sometimes allow for more efficient techniques. Also, these programs emphasize the use of commands from the current chapter. So, for example, instead of list operation, from a complexity point of view, hashing techniques or tailored data structures might be preferable. All subsections and sections are “self-contained” (meaning that no other code than the one presented is needed to evaluate the subsections and sections). The price for this “self-containedness” is that from time to time some code has to be repeated (such as manipulating polygons or forming random permutations of lists) instead of delegating such programming constructs to a package. Because this repetition could be construed as boring, the author typically uses a slightly different implementation to achieve the same goal.

■ 0.3.2 Programming Paradigms

In the *GuideBooks*, the author wants to show the reader that *Mathematica* supports various programming paradigms and also show that, depending on the problem under consideration and the goal (e.g., solution of a problem, test of an algorithm, development of a program), each style has its advantages and disadvantages. (For a general discussion concerning programming styles, see [3★], [40★], [22★], [31★], [15★], and [9★].) *Mathematica* supports a functional programming style. Thus, in addition to classical procedural programs (which are often less efficient and less elegant), programs using the functional style are also presented. In the first volume of the *Mathematica GuideBooks*, the programming style is usually dictated by the types of commands that have been discussed up to that point. A certain portion of the programs involve recursive, rule-based programming. The choice of programming style is, of course, partially (ultimately) a matter of personal preference. The *GuideBooks*’ main aim is to explain the operation, limits, and efficient application of the various *Mathematica* commands. For certain commands, this dictates a certain style of programming. However, the various programming styles, with their advantages and disadvantages, are not the main concern of the *GuideBooks*. In working with *Mathematica*, the reader is likely to use different programming styles depending if one wants a quick one-time calculation or a routine that will be used repeatedly. So, for a given implementation, the program structure may not always be the most elegant, fastest, or “prettiest”.

The *GuideBooks* are not a substitute for the study of *The Mathematica Book* [44★] (<http://documents.wolfram.com/v4>). It is impossible to acquire a deeper (full) understanding of *Mathematica* without a *thorough* study of this book (reading it twice from the first to the last page is highly recommended). It *defines* the language and the spirit of *Mathematica*. The reader will probably from time to time need to refer to parts of it, because not all commands are discussed in the *GuideBooks*. However, the story of what can be done with *Mathematica* does not end with the examples shown in *The Mathematica Book*. The *Mathematica GuideBooks* go beyond *The Mathematica Book*. They present larger programs for solving various problems and creating complicated graphics. In addition, the *GuideBooks* discuss a number of commands that are not or are only fleetingly mentioned in the manual (e.g., some specialized methods of mathematical functions and functions from the `Developer`` and `Experimental`` contexts), but which the author deems important. In the notebooks, the author gives special emphasis to discussions, remarks, and applications relating to several commands that are typical for *Mathematica* but not for most other programming languages, e.g., `Map`, `MapAt`, `MapIndexed`, `Distribute`, `Apply`, `Replace`, `ReplaceAll`, `Inner`, `Outer`, `Fold`, `Nest`, `NestList`, `FixedPoint`, `FixedPointList`, and `Function`. These commands allow to write exceptionally elegant, fast, and powerful programs. All of these commands are discussed in *The Mathematica Book* and others that deal with programming in *Mathematica* (e.g., [32★], [33★], and [41★]). However, the author’s experience suggests that a deeper understanding of these commands and their optimal applications comes only after working with *Mathematica* in the solution of more complicated problems.

Both the printed book and the electronic component contain material that is meant to teach in detail how to use *Mathematica* to solve problems, rather than to present the underlying details of the various scientific examples. It cannot be overemphasized that to master the use of *Mathematica*, its programming paradigms and individual functions, the reader must experiment; this is especially important, insightful, easily verifiable, and satisfying with graphics, which involve manipulating expressions, making small changes, and finding different approaches. Because the results can easily be visually checked, generating and modifying graphics is an ideal method to learn programming in *Mathematica*.

0.4 Exercises and Solutions

■ 0.4.1 Exercises

Each chapter includes a set of exercises and a detailed solution proposal for each exercise. When possible, all of the purely *Mathematica*-programming related exercises (these are most of the exercises of the Programming volume) should be solved by every reader. The exercises coming from mathematics, physics, and engineering should be solved according to the reader's interest. The most important *Mathematica* functions needed to solve a given problem are generally those of the associated chapter.

For a rough orientation about the content of an exercise, the subject is included in its title. The relative degree of difficulty is indicated by level superscript of the exercise number (^{L1} indicates easy, ^{L2} indicates medium, and ^{L3} indicates difficult). The author's aim was to present understandable interesting examples that illustrate the *Mathematica* material discussed in the corresponding chapter. Some exercises were inspired by recent research problems; the references given allow the interested reader to dig deeper into the subject.

The exercises are intentionally not hyperlinked to the corresponding solution. The independent solving of the exercises is an important part of learning *Mathematica*.

■ 0.4.2 Solutions

The *GuideBooks* contain solutions to each of the more than 1,000 exercises. Many of the techniques used in the solutions are not just one-line calls to built-in functions. It might well be that with further enhancements, a future version of *Mathematica* might be able to solve the problem more directly. (But due to different forms of some results returned by *Mathematica*, some problems might also become more challenging.) The author encourages the reader to try to find shorter, more clever, faster (in terms of runtime as well complexity), more general, and more elegant solutions. *Doing* various calculations is the most effective way to learn *Mathematica*. A proper *Mathematica* implementation of a function that solves a given problem often contains many different elements. The function(s) should have sensibly named and sensibly behaving options; for various (machine numeric, high-precision numeric, symbolic) inputs different steps might be required; shielding against inappropriate input might be needed; different parameter values might require different solution strategies and algorithms, helpful error and warning messages should be available. The returned data structure should be intuitive and easy to reuse; to achieve a good computational complexity, nontrivial data structures might be needed, etc. Most of the solutions do not deal with all of these issues, but only with selected ones and thereby leave plenty of room for more detailed treatments; as far as limit, boundary, and degenerate cases are concerned, they represent an outline of how to tackle the problem. Although the solutions do their job in general, they often allow considerable refinement and extension by the reader.

The reader should consider the given solution to a given exercise as a proposal; quite different approaches are often possible and sometimes even more efficient. The routines presented in the solutions are not the most general possible, because to make them foolproof for every possible input (sensible and nonsensical, evaluated and unevaluated, numerical and symbolical), the books would have had to go considerably beyond the mathematical and physical framework of the *GuideBooks*. In addition, few warnings are implemented for improper or improperly used arguments. The graphics provided in the solutions are mostly subject to a long list of refinements. Although the solutions do work, they are often sketchy and can be considerably refined and extended by the reader. This also means that the provided solutions to the exercises programs are not always very suitable for solving larger classes of problems. To increase their applicability would require considerably more code. Thus, it is not guaranteed that given routines will work correctly on related problems. To guarantee this generality and scalability, one would have to protect the variables better, implement formulas for more general or specialized cases, write functions to accept different numbers of variables, add type-checking and error-checking functions, and include corresponding error messages and warnings.

To simplify working through the solutions, the various steps of the solution are commented and are not always not packed in a `Module` or `Block`. In general, only functions that are used later are packed. For longer calculations, such as those in some of the exercises, this was not feasible and intended. The arguments of the functions are not always checked for their appropriateness as is desirable for robust code. But, this makes it easier for the user to test and modify the code.

0.5 The Books Versus the Electronic Components

■ 0.5.1 Working with the Notebooks

Each volume of the *GuideBooks* comes with a multiplatform DVD, containing fourteen main notebooks tailored for *Mathematica* 4 and compatible with *Mathematica* 5. Each notebook corresponds to a chapter from the printed books. (To avoid large file sizes of the notebooks, all animations are located in the Animations directory and not directly in the chapter notebooks.) The chapters (and so the corresponding notebooks) contain a detailed description and explanation of the *Mathematica* commands needed and used in applications of *Mathematica* to the sciences. Discussions on *Mathematica* functions are supplemented by a variety of mathematics, physics, and graphics examples. The notebooks also contain complete solutions to all exercises. Forming an electronic book, the notebooks also contain all text, as well as fully typeset formulas, and reader-editable and reader-changeable input. (Readers can copy, paste, and use the inputs in their notebooks.) In addition to the chapter notebooks, the DVD also includes a navigation palette and fully hyperlinked table of contents and index notebooks. The *Mathematica* notebooks corresponding to the printed book are fully evaluated. The evaluated chapter notebooks also come with hyperlinked overviews; these overviews are not in the printed book.

When reading the printed books, it might seem that some parts are longer than needed. The reader should keep in mind that the primary tool for working with the *Mathematica* kernel are *Mathematica* notebooks and that on a computer screen and there “length does not matter much”. The *GuideBooks* are basically a printout of the notebooks, which makes going back and forth between the printed books and the notebooks very easy. The *GuideBooks* give large examples to encourage the reader to investigate various *Mathematica* functions and to become familiar with *Mathematica* as a system for doing mathematics, as well as a programming language. Investigating *Mathematica* in the accompanying notebooks is the best way to learn its details.

To start viewing the notebooks, open the table of contents notebook `TableOfContents.nb`. *Mathematica* notebooks can contain hyperlinks, and all entries of the table of contents are hyperlinked. Navigating through one of the chapters is convenient when done using the navigator palette `GuideBooksNavigator.nb`.

When opening a notebook, the front end minimizes the amount of memory needed to display the notebook by loading it incrementally. Depending on the reader’s hardware, this might result in a slow scrolling speed. Clicking the “Load notebook cache” button of the `GuideBooksNavigator` palette speeds this up by loading the complete notebook into the front end.

For the vast majority of sections, subsections, and solutions of the exercises, the reader can just select such a structural unit and evaluate it (at once) on a year-2003 computer (≥ 512 MB RAM) typically in a matter of minutes. (On a pre-OSX Macintosh system, it might be necessary to increase the default memory sizes given for the *Mathematica* kernel and the front end.) Some sections and solutions containing many graphics may need hours of computation time. Also, more than 100 pieces of code run hours, even days. The inputs that are very memory intensive or produce large outputs and graphics are in inactive cells which can be activated by clicking the adjacent button. Because of potentially overlapping variable names between various sections and subsections, the author advises the reader not to evaluate an entire chapter at once.

The Overview Section of the chapters is set up for a front end and kernel running on the same computer and having access to the same file system. When using a remote kernel, the directory specification for the package `Overview.m` must be changed accordingly.

References can be conveniently extracted from the main text by selecting the cell(s) that refer to them (or parts of a cell) and then clicking the “Extract References” button. A new notebook with the extracted references will then appear.

The notebooks contain color graphics. (To rerender the pictures with a greater color depth or at a larger size, choose `Rerender Graphics` from the `Cell` menu.) With some of the used colors, black-and-white printouts would occasionally give

low-contrast results. For better black-and-white printouts of these graphics, the author recommends setting the `ColorOutput` option of the relevant graphics function to `GrayLevel`. The notebooks with animations (in the printed book, animations are typically printed as an array of about 10 to 20 individual graphics) typically contain between 60 and 120 frames. Rerunning the corresponding code with a large number of frames will allow the reader to generate smoother and longer-running animations.

Because many cell styles used in the notebooks are unique to the *GuideBooks*, when copying expressions and cells from the *GuideBooks* notebooks to other notebooks, one should first attach the style sheet notebook `GuideBooksStylesheet.nb` to the destination notebook, or define the needed styles in the style sheet of the destination notebook.

■ 0.5.2 Reproducibility of the Results

The 14 chapter notebooks contained in the electronic version of the *GuideBooks* were run under *Mathematica* 4 on a 2 GHz Intel Linux computer with 2 GB RAM. They need more than 100 hours of evaluation time. (This does not include the evaluation of the currently unevaluatable parts of code after the `Make Input` buttons.) For most subsections and sections, 512 MB RAM are recommended for a fast and smooth evaluation “at once” (meaning the reader can select the section or subsection, and evaluate all inputs without running out of memory or clearing variables) and the rendering of the generated graphic in the front end. Some subsections and sections need more memory when run. To reduce these memory requirements, the author recommends restarting the *Mathematica* kernel inside these subsections and sections, evaluating the necessary definitions, and then continuing. This will allow the reader to evaluate all inputs.

In general, regardless of the computer, with the same version of *Mathematica*, the reader should get the same results as shown in the notebooks. (The author has tested the code on Sun and Intel-based Linux computers, but this does not mean that some code might not run as displayed (because of different configurations, stack size settings, etc., but the disclaimer from the Preface applies everywhere). If an input does not work on a particular machine, please inform the author. Some deviations from the results given may appear because of the following:

- Inputs involving the function `Random[...]` in some form. (Often `SeedRandom` to allow for some kind of reproducibility and randomness at the same time is employed.)
- *Mathematica* commands operating on the file system of the computer, or make use of the type of computer (such inputs need to be edited using the appropriate directory specifications).
- Calculations showing some of the differences of floating-point numbers and the machine-dependent representation of these on various computers.
- Pictures using various fonts and sizes because of their availability (or lack thereof) and shape on different computers.
- Calculations involving `Timing` because of different clock speeds, architectures, operating systems, and libraries.
- Formats of results depending on the actual window width and default font size. (Often, the corresponding inputs will contain `Short`.)

Using anything other than *Mathematica* Version 4.0 might also result in different outputs. Examples of results that change form, but are all mathematically correct and equivalent, are the parameter variables used in underdetermined systems of linear equations, the form of the results of an integral, and the internal form of functions like `InterpolatingFunction` and `CompiledFunction`. Some inputs might no longer evaluate the same way because functions from a package were used and these functions are potentially built-in functions in a later *Mathematica* version. *Mathematica* is a very large and complicated program that is constantly updated and improved. Some of these changes might be design changes, superseded functionality, or potentially regressions, and as a result, some of the inputs might not work at all or give unexpected results in future versions of *Mathematica*.

0.6 Style and Design Elements

■ 0.6.1 Text and Code Formatting

The *GuideBooks* are divided into chapters. Each chapter consists of several sections, which frequently are further subdivided into subsections. General remarks about a chapter or a section are presented in the sections and subsections numbered 0. (These remarks usually discuss the structure of the following section and give teasers about the usefulness of the functions to be discussed.) Also, sometimes these sections serve to refresh the discussion of some functions already introduced earlier.

Following the style of *The Mathematica Book* [44★], the *GuideBooks* use the following fonts: For the main text, Times; for *Mathematica* inputs and built-in *Mathematica* commands, Courier plain (like `Plot`); and for user-supplied arguments, Times italic (like *userArgument₁*). Built-in *Mathematica* functions are introduced in the following style:

```
MathematicaFunctionToBeIntroduced[typeIndicatingUserSuppliedArgument(s)]
```

is a description of the built-in command `MathematicaFunctionToBeIntroduced` upon its first appearance. A definition of the command, along with its parameters is given. Here, *typeIndicatingUserSuppliedArgument(s)* is one (or more) user-supplied expression(s) and may be written in an abbreviated form or in a different way for emphasis.

The actual *Mathematica* inputs and outputs appear in the following manner (as mentioned above, virtually all inputs are given in `InputForm`).

```
(* A comment. It will be/is ignored as Mathematica input:
   Return only one of the solutions *)
Last[Solve[{x2 - y == 1, x - y2 == 1}, {x, y}]
```

When referring in text to variables of *Mathematica* inputs and outputs, the following convention is used: Fixed, nonpattern variables (including local variables) are printed in Courier plain (the equations solved above contained the variables `x` and `y`). User supplied arguments to built-in or defined functions with pattern variables are printed in Times italic. The next input defines a function generating a pair of polynomial equations in x and y .

```
equationPair[x_, y_] := {x2 - y == 1, x - y2 == 1}
```

x and y are pattern variables (same letters, but different font from the actual code fragments `x_` and `y_`) that can stand for any argument. Here we call the function `equationPair` with the two arguments $u + v$ and $w - z$.

```
equationPair[u + v, w - z]
```

Occasionally, explanation about a mathematics or physics topic is given before the corresponding *Mathematica* implementation is discussed. These sections are marked as follows:

Mathematical Remark: Special Topic in Mathematics or Physics

A *short* summary or review of mathematical or physical ideas necessary for the following example(s).

From time to time, *Mathematica* is used to analyze expressions, algorithms, etc. In some cases, results in the form of English sentences are produced programmatically. To differentiate such automatically generated text from the main text, in most instances such text is prefaced by “o” (structurally the corresponding cells are of type "PrintText" versus "Text" for author-written cells).

Code pieces that either run for quite long, or need a lot of memory, or are tangent to the current discussion are displayed in the following manner.

```

mathematicaCodeWhichEitherRunsVeryLongOrThatIsVeryMemoryIntensive:
OrThatProducesAVeryLargeGraphicOrThatIsASideTrackToTheSubjectUnder:
Discussion
(* with some comments on how the code works *)

```

To run a code piece like this, click the **Make Input** button above it. This will generate the corresponding input cell that can be evaluated if the reader’s computer has the necessary resources.

The reader is encouraged to add new inputs and annotations to the electronic notebooks. There are two styles for reader-added material: "ReaderInput" (a *Mathematica* input style and simultaneously the default style for a new cell) and "ReaderAnnotation" (a text-style cell type). They are primarily intended to be used in the Reading environment. These two styles are indented more than the default input and text cells, have a green left bar and a dingbat. To access the "ReaderInput" and "ReaderAnnotation" styles, press the system-dependent modifier key (such as Control or Command) and 9 and 7, respectively.

■ 0.6.2 References

Because the *GuideBooks* are concerned with the solution of mathematical and physical problems using *Mathematica* and are not mathematics or physics monographs, the author did not attempt to give complete references for each of the applications discussed [37★]. The references cited in the text pertain mainly to the applications under discussion. Most of the citations are from the more recent literature; references to older publications can be found in the cited ones. Frequently URLs for downloading relevant or interesting information are given. (The URL addresses worked at the time of printing and, hopefully, will be still active when the reader tries them.) References for *Mathematica*, for algorithms used in computer algebra, and for applications of computer algebra are collected in the Appendix.

The references are listed at the end of each chapter in alphabetical order. In the notebooks, the references are hyperlinked to all their occurrences in the main text. Multiple references for a subject are not cited in numerical order, but rather in the order of their importance, relevance, and suggested reading order for the implementation given.

In a few cases (e.g., pure functions in Chapter 3, some matrix operations in Chapter 6), references to the mathematical background for some built-in commands are given—mainly for commands in which the mathematics required extends beyond the familiarity commonly exhibited by non-mathematicians. The *GuideBooks* do not discuss the algorithms underlying such complicated functions, but sometimes use *Mathematica* to “monitor” the algorithms.

References of the form *abbreviationOfAScientificField/yearMonthPreprintNumber* (such as quant-ph/0012147) refer to the arXiv preprint server [42★], [21★], [29★] at <http://arXiv.org>. When a paper appeared as a preprint and (later) in a journal, typically only the more accessible preprint reference is given. For the convenience of the reader, at the end of these references, there is a **Get Preprint** button. Click the button to display a palette notebook with hyperlinks to the corresponding preprint at the main preprint server and its mirror sites. (Some of the older journal articles can be downloaded free of charge from some of the digital mathematics library servers, such as <http://gdz.sub.uni-goettingen.de>, <http://www.emis.de>, <http://www.numdam.org>, and <http://dieper.aib.uni-linz.ac.at>.)

■ 0.6.3 Variables Scoping, Input Numbering and Warning Messages

Some of the *Mathematica* inputs intentionally cause error messages, infinite loops, and so on, to illustrate the operation of a *Mathematica* command. These messages also arise in the user's practical use of *Mathematica*. So, instead of presenting polished and perfected code, the author prefers to illustrate the potential problems and limitations associated with the use of *Mathematica* applied to "real life" problems. The one exception are the spelling warning messages `General::spell` and `General::spell1` that would appear relatively frequently because "similar" names are used eventually. For easier and less defocused reading, these messages are turned off in the initialization cells. (When working with the notebooks, this means that the pop-up window asking the user "Do you want to automatically evaluate all the initialization cells in the notebook?" should always be answered with a "yes".) For the vast majority of graphics presented, the picture is the focus, not the returned *Mathematica* expression representing the picture. That is why the `Graphics` and `Graphics3D` output is suppressed in most situations.

To improve the code's readability, no attempt has been made to protect all variables that are used in the various examples. This protection could be done with `Clear`, `Remove`, `Block`, `Module`, `With`, and others. Not protecting the variables allows the reader to modify, in a somewhat easier manner, the values and definitions of variables, and to see the effects of these changes. On the other hand, there may be some interference between variable names and values used in the notebooks and those that might be introduced when experimenting with the code. When readers examine some of the code on a computer, reevaluate sections, and sometimes perform subsidiary calculations, they may introduce variables that might interfere with ones from the *GuideBooks*. To partially avoid this problem, and for the reader's convenience, sometimes `Clear[sequenceOfVariables]` and `Remove[sequenceOfVariables]` are sprinkled throughout the notebooks. This makes experimenting with these functions easier.

The numbering of the *Mathematica* inputs and outputs typically does not contain all consecutive integers. Some pieces of *Mathematica* code consist of multiple inputs per cell; so, therefore, the line numbering is incremented by more than just 1. As mentioned, *Mathematica* should be restarted at every section, or subsection or solution of an exercise, to make sure that no variables with values get reused. The author also explicitly asks the reader to restart *Mathematica* at some special positions inside sections. This removes previously introduced variables, eliminates all existing contexts, and returns *Mathematica* to the typical initial configuration to ensure reproduction of the results and to avoid using too much memory inside one session.

■ 0.6.4 Notations and Symbols

The symbols used in typeset mathematical formulas are not uniform and unique throughout the *GuideBooks*. Various mathematical and physical quantities (like normals, rotation matrices, and field strengths) are used repeatedly in this book. Frequently the same notation is used for them, but depending on the context, also different ones are used, e.g. sometimes bold is used for a vector (such as \mathbf{r}) and sometimes an arrow (such as \vec{r}). Matrices appear in bold or as doublestruck letters. Depending on the context and emphasis placed, different notations are used in display equations and in the *Mathematica* input form. For instance, for a time-dependent scalar quantity of one variable $\psi(t; x)$, we might use one of many patterns, such as $\psi[t][x]$ (for emphasizing a parametric t -dependence) or $\psi[t, x]$ (to treat t and x on an equal footing) or $\psi[t, \{x\}]$ (to emphasize the one-dimensionality of the space variable x).

Mathematical formulas use standard notation. To avoid confusion with *Mathematica* notations, the use of square brackets is minimized throughout. Following the conventions of mathematics notation, square brackets are used for three cases: a) Functionals, such as $\mathcal{F}_t[f(t)](\omega)$ for the Fourier transform of a function $f(t)$. b) Power series coefficients, $[x^k](f(x))$ denotes the coefficient of x^k of the power series expansion of $f(x)$ around $x = 0$. c) Closed intervals, like $[a, b]$ (open intervals are denoted by (a, b)). Grouping is exclusively done using parentheses. Upper-case double-struck letters denote domains of numbers, \mathbb{Z} for integers, \mathbb{N} for nonnegative integers, \mathbb{Q} for rational numbers, \mathbb{R} for reals, and \mathbb{C} for complex numbers. Points in \mathbb{R}^n (or \mathbb{C}^n) with explicitly given coordinates are indicated using curly braces $\{c_1, \dots, c_n\}$. The symbols \wedge and \vee for And and Or are used in logical formulas.

For variable names in formula- and identity-like *Mathematica* code, the symbol (or small variations of it) traditionally used in mathematics or physics is used. In program-like *Mathematica* code, the author uses very descriptive, sometimes abbreviated, but sometimes also slightly longish, variable names, such as `buildBrillouinZone` and `FibonacciChainMap`.

■ 0.6.5 Units

In the examples involving concepts drawn from physics, the author tried to enhance the readability of the code (and execution speed) by not choosing systems of units involving numerical or unit-dependent quantities. (For more on the choice and treatment of units, see [38★], [4★], [5★], [10★], [13★], [11★], [12★], [35★], [34★], [30★], [36★], [43★], [20★], [24★], [18★], [25★], [23★].) Although *Mathematica* can carry units along with the symbols representing the physical quantities in a calculation, this requires more programming and frequently diverts from the essence of the problem. Choosing a system of units that allows the equations to be written without (unnecessary in computations) units often gives considerable insight into the importance of the various parts of the equations because the magnitudes of the explicitly appearing coefficients are more easily

■ 0.6.6 Cover Graphics

The cover graphics of the *GuideBooks* stem from the *Mathematica GuideBooks* themselves. The construction ideas and their implementation are discussed in detail in the corresponding *GuideBook*.

- The cover graphic of the Programming volume shows 42 tori, 12 of which are in the dodecahedron's face planes and 30 which are in the planes perpendicular to the dodecahedron's edges. Subsection 1.2.5 of Chapter 1 discusses the implementation.
- The cover graphic of the Graphics volume first subdivides the faces of a dodecahedron into small triangles and then rotates randomly selected triangles around the dodecahedron's edges. The proposed solution of Exercise 1b of Chapter 2 discusses the implementation.
- The cover graphic of the Numerics volume visualizes the electric field lines of a symmetric arrangement of positive and negative charges. Subsection 1.11.1 discusses the implementation.
- The cover graphic of the Symbolics volume visualizes the derivative of the Weierstrass \wp function over the Riemann sphere. The "threefold blossoms" arise from the poles at the centers of the periodic array of period parallelograms. Exercise 3j of Chapter 2 discusses the implementation.
- The four spine graphics show the inverse elliptic nome function q^{-1} , a function defined in the unit disk with a boundary of analyticity mapped to a triangle, a square, a pentagon, and a hexagon. Exercise 16 of Chapter 2 of the Graphics volume discusses the implementation.

0.7 Production History

The original set of notebooks was developed in the 1991–1992 academic year on an Apple Macintosh IIfx with 20 MB RAM using *Mathematica* Version 2.1. Over the years, the notebooks were updated to *Mathematica* Version 2.2, then to Version 3, and finally for Version 4 for the first printed edition of the *Mathematica GuideBooks*. The electronic component has been updated to be compatible with *Mathematica* 5. The first step in creating the book was the translation of a set of Macintosh notebooks used for lecturing and written in German into English by Larry Shumaker. This was done primarily by a translation program and afterward by manually polishing the English version. Then the notebooks were transformed into *TEX* files using the program `nb2tex` on a NeXT computer. The resulting files were manually edited, equations prepared in the original German notebooks were formatted with *TEX*, and macros were added corresponding to the design of the book. (The translation to *TEX* was necessary because *Mathematica* Version 2.2 did not allow for book-quality printouts.) They were updated and refined for nearly three years, and then *Mathematica* 3 notebooks were generated from the *TEX* files using a preliminary version of the program `tex2nb`. Historically and technically, this was an important step because it transformed all of the material of the *GuideBooks* into *Mathematica* expressions and allowed for automated changes and updates in the various editing stages. (Using the *Mathematica* kernel allowed one to process and modify the notebook files of these books in a uniform and time-efficient manner.) Then, the notebooks were expanded in size and scope and updated to *Mathematica* 4. In the second half of the year 2003, the *Mathematica* programs of the notebooks were revised to work with *Mathematica* 5. A special set of styles was created to generate the actual PostScript as printouts from the notebooks. All inputs were evaluated with this style sheet, and the generated Postscript was directly used for the book production. Using a little *Mathematica* program, the index was generated from the notebooks (which are *Mathematica* expressions), containing all index entries as cell tags.

0.8 Four General Suggestions

A reader new to *Mathematica* should take into account these four suggestions.

- There is usually more than one way to solve a given problem using *Mathematica*. If one approach does not work or returns the wrong answer or gives an error message, make every effort to understand what is happening. Even if the reader has succeeded with an alternative approach, it is important to try to understand why other attempts failed.
- Mathematical formulas, algorithms, and so on, should be implemented as directly as possible, even if the resulting construction is somewhat “unusual” compared to that in other programming languages. In particular, the reader should not simply translate C, Pascal, Fortran, or other programs line-by-line into *Mathematica*, although this is indeed possible. Instead, the reader should instead reformulate the problem in a clear mathematical way. For example, `Do`, `While`, and `For` loops are frequently unnecessary, convergence (for instance, of sums) can be checked by *Mathematica*, and `If` tests can often be replaced by a corresponding pattern. The reader should start with an exact mathematical description of the problem [27★], [28★]. For example, it does not suffice to know which transformation formulas have to be used on certain functions; one also needs to know how to apply them. “The power of mathematics is in its precision. The precision of mathematics must be used precisely.” [17★]
- If the exercises, examples, and calculation of the *GuideBooks* or the listing of calculation proposals from Exercise 1 of Chapter 1 of the Programming volume are not challenging enough or do not cover the reader’s interests, consider the following idea, which provides a source for all kinds of interesting and difficult problems: The reader should select a built-in command and try to reconstruct it using other built-in commands and make it behave as close to the original as possible in its operation, speed, and domain of applicability, or even to surpass it. (Replicating the following functions is a serious chal-

leng: N, Factor, FactorInteger, Integrate, NIntegrate, Solve, DSolve, NDSolve, Series, Sum, Limit, Root, Prime, or PrimeQ.)

■ If the reader tries to solve a smaller or larger problem in *Mathematica* and does not succeed, keep this problem on a “to do” list and periodically review this list and try again. Whenever the reader has a clear strategy to solve a problem, this strategy can be implemented in *Mathematica*. The implementation of the algorithm might require some programming skills, and by reading through this book, the reader will become able to code more sophisticated procedures and more efficient implementations. After the reader has acquired a certain amount of *Mathematica* programming familiarity, implementing virtually all “procedures” which the reader can (algorithmically) carry out with paper and pencil will become straightforward.

References

- ★1 P. Abbott. *The Mathematica Journal* 4, 415 (2000).
- ★2 P. Abbott. *The Mathematica Journal* 9, 31 (2003).
- ★3 H. Abelson, G. Sussman. *Structure and Interpretation of Computer Programs*, MIT Press, Cambridge, MA, 1985.
- ★4 G. I. Barenblatt. *Similarity, Self-Similarity, and Intermediate Asymptotics*, Consultants Bureau, New York, 1979.
- ★5 F. A. Bender. *An Introduction to Mathematical Modeling*, Wiley, New York, 1978.
- ★6 G. Benfatto, G. Gallavotti. *Renormalization Group*, Princeton University Press, Princeton, 1995.
- ★7 L. Blum, F. Cucker, M. Shub, S. Smale. *Complexity and Real Computation*, Springer, New York, 1998.
- ★8 P. Bürgisser, M. Clausen, M. A. Shokrollahi. *Algebraic Complexity Theory*, Springer, Berlin, 1997.
- ★9 L. Cardelli, P. Wegner. *Comput. Surv.* 17, 471 (1985).
- ★10 J. F. Carinena, M. Santander in P. W. Hawkes (ed.). *Advances in Electronics and Electron Physics* 72, Academic Press, New York, 1988.
- ★11 E. A. Desloge. *Am. J. Phys.* 52, 312 (1984).
- ★12 C. L. Dym, E. S. Ivey. *Principles of Mathematical Modelling*, Academic Press, New York, 1980.
- ★13 A. C. Fowler. *Mathematical Models in the Applied Sciences*, Cambridge University Press, Cambridge, 1997.
- ★14 T. Gannon. *arXiv:math.QA/9906167* (1999). *Get Preprint*
- ★15 R. J. Gaylord, S. N. Kamin, P. R. Wellin. *An Introduction to Programming with Mathematica*, TELOS/Springer-Verlag, Santa Clara, 1993.
- ★16 J. Glynn, T. Gray. *The Beginner’s Guide to Mathematica Version 3*, Cambridge University Press, Cambridge, 1997.
- ★17 D. Greenspan in R. E. Mickens (ed.). *Mathematics and Science*, World Scientific, Singapore, 1990.
- ★18 G. W. Hart. *Multidimensional Analysis*, Springer-Verlag, New York, 1995.
- ★19 A. K. Hartman, H. Rieger. *arXiv:cond-mat/0111531* (2001). *Get Preprint*
- ★20 E. Isaacson, M. Isaacson. *Dimensional Methods in Engineering and Physics*, Edward Arnold, London, 1975.
- ★21 A. Jackson. *Notices Am. Math. Soc.* 49, 23 (2002).
- ★22 R. D. Jenks, B. M. Trager in J. von zur Gathen, M. Giesbracht (eds.). *Symbolic and Algebraic Computation*, ACM Press, New York, 1994.

- ★23 C. G. Jesudason. *arXiv:physics/0403033* (2004). *Get Preprint*
- ★24 C. Kauffmann in A. van der Burgh (ed.). *Topics in Engineering Mathematics*, Kluwer, Dordrecht, 1993.
- ★25 R. Khanin in B. Mourrain (ed.). *ISSAC 2001*, ACM, Baltimore, 2001.
- ★26 P. Kleinert, H. Schlegel. *Physica A* 218, 507 (1995).
- ★27 D. E. Knuth. *Am. Math. Monthly* 81, 323 (1974).
- ★28 D. E. Knuth. *Am. Math. Monthly* 92, 170 (1985).
- ★29 G. Kuperberg. *arXiv:math.HO/0210144* (2002). *Get Preprint*
- ★30 J. D. Logan. *Applied Mathematics*, Wiley, New York, 1987.
- ★31 K. C. Louden. *Programming Languages: Principles and Practice*, PWS-Kent, Boston, 1993.
- ★32 R. Maeder. *Programming in Mathematica*, Addison-Wesley, Reading, 1997.
- ★33 R. Maeder. *The Mathematica Programmer*, Academic Press, New York, 1993.
- ★34 B. S. Massey. *Measures in Science and Engineering*, Wiley, New York, 1986.
- ★35 G. Messina, S. Santangelo, A. Paoletti, A. Tucciarone. *Nuov. Cim. D* 17, 523 (1995).
- ★36 J. Molenaar in A. van der Burgh, J. Simonis (eds.). *Topics in Engineering Mathematics*, Kluwer, Dordrecht, 1992.
- ★37 E. Pascal. *Repertorium der höheren Mathematik* Theil 1/1 (page V, paragraph 3), Teubner, Leipzig, 1900.
- ★38 S. H. Romer. *Am. J. Phys.* 67, 13 (1999).
- ★39 R. Sedgewick, P. Flajolet. *Analysis of Algorithms*, Addison-Wesley, Reading, 1996.
- ★40 R. Sethi. *Programming Languages: Concepts and Constructions*, Addison-Wesley, New York, 1989.
- ★41 D. B. Wagner. *Power Programming with Mathematica: The Kernel*, McGraw-Hill, New York, 1996.
- ★42 S. Warner. *arXiv:cs.DL/0101027* (2001). *Get Preprint*
- ★43 H. Whitney. *Am. Math. Monthly* 75, 115, 227 (1968).
- ★44 S. Wolfram. *The Mathematica Book*, Cambridge University Press and Wolfram Media, 1999.