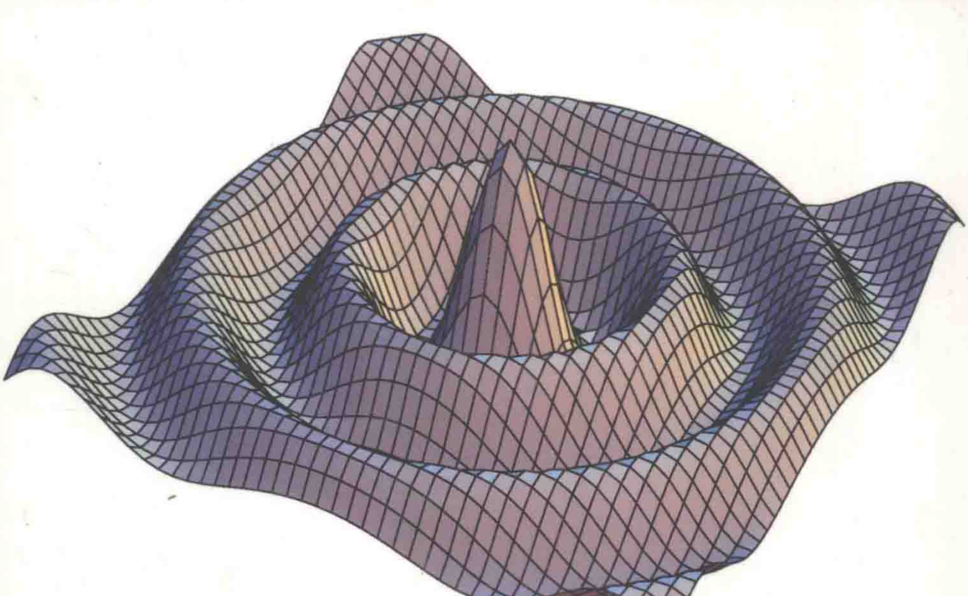
JOHN S. ROBERTSON

ENGINEERING MATHEMATICS WITH MATHEMATICA



ENGINEERING MATHEMATICS WITH MATHEMATICA

John S. Robertson
Professor of Applied Mathematics
United States Military Academy

McGraw-Hill, Inc.

New York St. Louis San Francisco Auckland Bogotá Caracas
Lisbon London Madrid Mexico Milan Montreal New Delhi
San Juan Singapore Sydney Tokyo Toronto

ENGINEERING MATHEMATICS WITH MATHEMATICA

This book is printed on acid-free paper.

2 3 4 5 6 7 8 9 0 DOC DOC 9 0 9 8 7 6 5

ISBN 0-07-053171-4

The editors were Maggie Lanzillo and Jack Shira; the production supervisor was Denise L. Puryear. R.R. Donnelley & Sons was printer and binder.

The illustration shown on the cover was generated by *Mathematica* and depicts the surface created by rotating the graph of the Bessel function of the first kind of order zero about the z-axis. In particular,

$$z = J_0(\sqrt{x^2 + y^2})$$

with $-15 < x < 15$ and $-15 < y < 15$.

This manual was typeset by the author with LaTeX, the document preparation system by Leslie Lamport built atop the TeX engine of Donald Knuth. Special design macros were provided by ETP Services. The illustrations were prepared on a Macintosh IIci with *Mathematica* and MacDraw Pro.

Library of Congress Cataloging-in-Publication Data

ENGINEERING MATHEMATICS WITH MATHEMATICA

International Series in Pure and Applied Mathematics

Ahlfors: *Complex Analysis*
Bender and Orszag: *Advanced Mathematical Methods for Scientists and Engineers*
Boas: *Invitation to Complex Analysis*
Brown and Churchill: *Fourier Series and Boundary Value Problems*
Buchanan and Turner: *Numerical Methods and Analysis*
Buck: *Advanced Calculus*
Chartrand and Oellerman: *Applied and Algorithmic Graph Theory*
Colton: *Partial Differential Equations*
Conte and de Boor: *Elementary Numerical Analysis: An Algorithmic Approach*
Edelstein-Keshet: *Mathematical Models in Biology*
Farlow: *An Introduction to Differential Equations and Their Applications*
Goldberg: *Matrix Theory with Applications*
Gulick: *Encounters with Chaos*
Hill: *Experiments in Computational Matrix Algebra*
Kurtz: *Foundations of Abstract Mathematics*
Lewin and Lewin: *An Introduction to Mathematical Analysis*
Morash: *Bridge to Abstract Mathematics: Mathematical Proof and Structures*
Parzynski and Zipse: *Introduction to Mathematical Analysis*
Pinsky: *Partial Differential Equations and Boundary-Value Problems with Applications*
Pinter: *A Book of Abstract Algebra*
Ralston and Rabinowitz: *A First Course in Numerical Analysis*
Ritger and Rose: *Differential Equations with Applications*
Robertson: *Engineering Mathematics with Mathematica*
Rudin: *Functional Analysis*
Rudin: *Principles of Mathematical Analysis*
Rudin: *Real and Complex Analysis*
Simmons: *Differential Equations with Applications and Historical Notes*
Small and Hosack: *Calculus: An Integrated Approach*
Small and Hosack: *Explorations in Calculus with a Computer Algebra System*
Vanden Eynden: *Elementary Number Theory*
Walker: *Introduction to Abstract Algebra*

ABOUT THE AUTHOR

John S. Robertson is a Professor of Applied Mathematics at the United States Military Academy. He received his Ph.D. in Mathematics from Rensselaer Polytechnic Institute. Since 1991 he has been a full professor at West Point. He is an active researcher in computational acoustics and computer applications to mathematics education. His work has been supported by the Office of Naval Research, the National Aeronautics and Space Administration, and a number of Army research agencies. He is a member of the Acoustical Society of America and the Society for Industrial and Applied Mathematics.

To the memory of Don Alvaro del Portillo
1914–1994
Deo omnia gloria!

PREFACE

This book is intended for use as a supplemental tool for courses in engineering mathematics, applied ordinary and partial differential equations, vector analysis, applied complex analysis, and other advanced courses in which *Mathematica*[1] is used. My goal in writing this text was to prepare a supplementary book that could be used to guide students through a series of laboratory exercises that would do the following:

- present cogent applications of the mathematics
- demonstrate the effective uses of the computational tool (in this case, *Mathematica*) to do the mathematics
- provide discussion of the results obtained by using *Mathematica*
- stimulate thought about and analysis of additional applications

Each chapter has been written so that the material it contains may be covered in a typical laboratory session of about 1-1/2 to 2 hours. The goals for every laboratory are stated at the beginning of each chapter. Mathematical concepts are then discussed within a framework of abundant engineering applications and problem-solving techniques using *Mathematica*. I have tried to keep the *Mathematica* instruction *per se* to a minimum, but have included enough material to get students up and running quickly.

Each chapter is followed by a set of exercises. Many of these are exploratory in nature and are intended to serve as a starting point for a student's mathematical experimentation. In addition, since most of the exercises can be solved

[1] *Mathematica* is a registered trademark of Wolfram Research, Inc.

in more than one way, I have not provided an answer key for the student or instructor. Students should be encouraged to develop their own problem-solving skills with *Mathematica* and not just look for the "correct" answer.

Colleges and universities across the nation have been carefully re-examining the ways in which undergraduate mathematics is taught and done. The advent of computer algebra systems such as *Mathematica*, which can perform elaborate symbolic calculations, in conjunction with the rapidly expanding power of computers to function as graphic visualization devices have forced a critical rethinking of how much of what is called higher mathematics ought to be taught and done.

The calculus reform movement has already borne much fruit. As it matures, the same style of innovative thinking must subsequently be brought to bear on virtually all the mathematics courses taken by upperclassmen. While core mathematics has been the object of substantial national attention, less emphasis has been placed on adapting advanced courses to the technology and the student expectations it brings with it.

For these reasons, one of the most promising areas in which to exploit computation is in "engineering mathematics," a rubric which covers applied ordinary and partial differential equations, vector analysis, and applied complex analysis in courses normally taken junior and senior year. One approach to this problem is to write new textbooks with clearly-woven computational threads. I have taken a different approach by presenting discussion that is compatible with a broad range of engineering mathematics texts, as well as smaller, more specialized texts in differential equations and complex variables.

Although it might be desirable to make such a laboratory text independent of any particular software package, this goal is not yet in sight. There are simply too many differences in package front-ends and capabilities. In my view, a laboratory text must deal concretely with the details of a specific package. For a number of reasons, I have selected *Mathematica* as the computer algebra package for this text. *Mathematica* possesses a wealth of features which make it an excellent laboratory tool for engineering mathematics. In addition, *Mathematica* is available on a broad array of platforms—386 PCs, Macintoshes, Sun Sparcstations, IBM RS/6000s, etc.—and is found at many universities.

I would like to thank many, many people for their help while I prepared this book. My wife Julia alternately encouraged and cajoled me during the many months of its writing. Colonel Frank Giordano provided numerous useful insights and kept more than one wolf at bay. Dr. Norbert Carballo provided timely advice regarding the true purpose of the effort. Prof. Heidi A. Pattee of Oregon State University carefully read the manuscript and offered a myriad of suggestions which substantially improved the book. My editor, Maggie Lanzillo, was simply wonderful.

John S. Robertson

CONTENTS

CHAPTER

1

INTRODUCTION

1.1 GOALS

a. To become familiar with the basic syntax of *Mathematica*.

b. To plot two-dimensional graphs.

c. To plot three-dimensional surfaces.

1.2 ABOUT *Mathematica*

Mathematica is a powerful symbolic algebra tool that provides an extraordinarily rich variety of symbolic, graphical and numerical capabilities to anyone working with engineering mathematics.

Mathematica

FIGURE 1.1
Clicking on this (or a similar) icon starts *Mathematica* on the Macintosh.

Mathematica sessions are started differently on different machines. For example, on the Macintosh, you can click on a *Mathematica* icon, as shown in Fig. 1.1. This will open a *Mathematica* notebook into which you can start typing.

On a UNIX-based workstation, such as an IBM RISC System 6000, you would type `math` at the command prompt. In this case you will see the following text in your window:[1]

```
Mathematica 2.0 for IBM RISC System 6000
Copyright 1988-91 Wolfram Research, Inc.
 -- X11 windows graphics initialized --

In[1]:=
```

Refer to the *Local Guide* that came with your *Mathematica* implementation on starting *Mathematica* on other computer systems.[2]

Mathematica syntax, while initially a bit strange, is fairly easy to learn and remarkably consistent. As you work with it, keep the following rules in mind:

- *Mathematica* is case-sensitive.
- All *Mathematica* functions are capitalized.
- Function arguments are always delineated with square brackets, i.e. `[...]`.
- Lists are always delineated with curly brackets, i.e. {...}.
- Variable ranges (for integration, plotting, and counting) are always built with lists.
- A double question mark ?? followed by the *Mathematica* function name will elicit a short help message. This is useful only if you already understand the command. There is no substitute for referring to the *Mathematica* book[3] for definitive guidance on a particular function or operation.

1.3 BASIC ALGEBRA AND CALCULUS OPERATIONS

There are a few fundamental operations which must be mastered early on if the power of *Mathematica* is to be put to good use in the laboratory exercises. Consider the following expression:

[1] As of this writing, notebooks are available for major UNIX platforms. With the release of *Mathematica* Version 2.2, you can also obtain a window directly to the *Mathematica* kernel on the Macintosh. Notebooks, though, are easier to manage, and their use should be *de rigeur* when available.

[2] In this and all subsequent chapters, input and output will be confined between the upward- and downward-facing horizontal brackets as shown in the above example. In addition, within a section, statements will always be consecutively numbered beginning with `1`, e.g. `In[1]:=` and `Out[1]=`.

[3] Stephen Wolfram, *Mathematica: A system for doing mathematics by computer*, Addison-Wesley, Redwood City, CA, 1991.

```
In[1]:= (x + y)^2

                2
Out[1]= (x + y)
```

In this example, `x` and `y` are variables and the ^ operator denotes exponentiation. *Mathematica* returns the result in so-called display form, not unlike the way we would write the result down on paper. Input lines are always numbered like `In[n]` and output lines like `Out[n]` where `n` is the sequential number of the line beginning with the `n = 1` for the current *Mathematica* session.

Mathematica can be directed to expand the result with the `Expand[]` function:

```
In[2]:= Expand[%]

         2            2
Out[2]= x  + 2 x y + y
```

The `%` symbol stands for the output of the immediately previous calculation. In this example, `%` is equivalent to `Out[1]`. This result of the last command can be further manipulated, say, by subtracting `4 x y` from it:

```
In[3]:= % - 4 x y

         2            2
Out[3]= x  - 2 x y + y
```

Note that spaces between symbols designates *implied* multiplication. The use of the asterisk `*` makes the intention to multiply explicit, i.e., `4*x*y` is equivalent to `4 x y`.

In any case, to factor the result shown in `Out[3]` we use the `Factor[]` function:

```
In[4]:= Factor[%]

                 2
Out[4]= (-x + y)
```