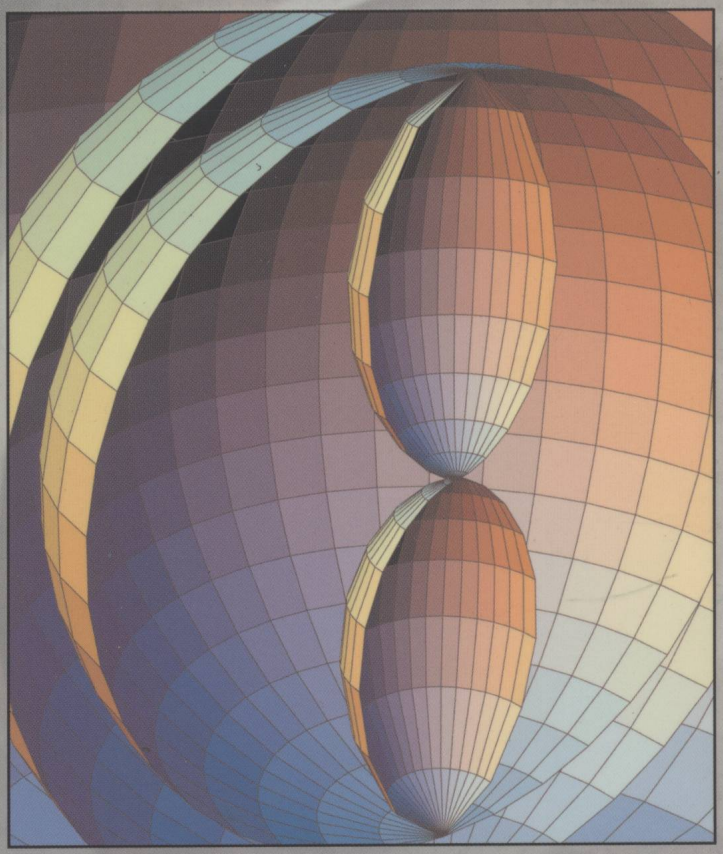
MATHEMATICA®

FOR PHYSICS

SECOND EDITION



Robert L. Zimmerman
Fredrick I. Olness

MATHEMATICA FOR PHYSICS

SECOND EDITION

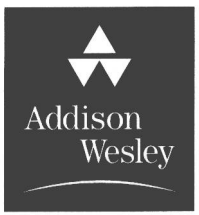
Robert L. Zimmerman
University of Oregon

Fredrick I. Olness
Southern Methodist University

San Francisco Boston New York
Capetown Hong Kong London Madrid Mexico City
Montreal Munich Paris Singapore Sydney Tokyo Toronto

Acquisitions Editor: Adam Black
Project Editor: Nancy Benton
Production Editor: Joan Marsh
Text Designer: Leslie Galen
Cover Designer: Blakeley Kim
Marketing Manager: Christy Lawrence
Manufacturing Coordinator: Vivian McDougal
Project Coordination and Electronic Page Makeup: Integre Technical Publishing Co., Inc.

Library of Congress Cataloging-in-Publication Data
Zimmerman, Robert L.
Mathematica for physics / Robert Zimmerman, Fredrick Olness–2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 0-8053-8700-5
1. Mathematical physics—Data processing 2. *Mathematica* (Computer file) I. Olness, Fredrick Iver. II. Title.
QC20.Z56 2003
530′.0285′53–dc21 2002004623



1 2 3 4 5 6 7 8 9 10 —MA— 9897969594
www.aw.com/aw

Foreword

To be good at physics you no longer have to be good at calculating.

When I started doing physics in the early 1970s, it was generally thought that being able to do long calculations by hand was an essential skill. But as it happens I was never very good at such calculations. And so I looked for an alternative, and before long I found one: I discovered that I did not really need to do the calculations myself—I could get a computer to do them instead.

My physics papers from the late 1970s were full of elaborate calculations. And from seeing this many people came to the conclusion that I must be a great calculator. But in fact that was far from the truth, and what was actually going on was that I was doing all my calculations by computer.

Of course, in those days it required a considerable amount of programming effort to get a computer to do one's calculations—particularly when algebra and graphics were involved. But after a few years I realized that with new generations of computers and software engineering tools it would actually be possible to build a single software system that would be able to handle all the various kinds of calculations that one needed to do. And from this realization I came in the end to develop *Mathematica*.

So now that *Mathematica* exists, what does it mean for physics? In the years since the first version of *Mathematica* became available (version 1.0 was released on June 23, 1988), a vast amount of new physics has been done with it. Indeed, for example, if one looks today at any of the leading physics journals, one can tell that a large fraction of the calculations and pictures in them were done with *Mathematica*.

But it is not just at the level of research that *Mathematica* affects the way physics is done. Anyone who learns physics today can do so in a very different way because of *Mathematica*.

And that is the point of this book. What the authors have done is to take the topics from a mainstream physics course and show how each of them can be handled in a new way with *Mathematica*.

The results are impressive. Over and over again what was once a calculation too lengthy to be reproduced as part of a course now becomes a few lines of *Mathematica* input that can be executed in a matter of seconds. And instead of having to explain in painful detail the mechanics of the calculation, one can concentrate on the conceptual issues that underlie it.

One of the things that one sees in many of the examples in the book is the extent to which *Mathematica* has narrowed the gap between physics learning and physics research. For once a calculation has been set up in *Mathematica*, one can try the calculation in many different cases—not just ones already covered in textbooks, but also ones that may never have been tried before.

This book will no doubt be read by many students of physics and many professors. Students may wonder how many of the calculations it describes could ever have been done before *Mathematica*. Perhaps sometimes their professors will tell them tales from the heroic age of hand calculation. But mostly I hope that students and professors alike will be able to use the technology that we have built and material of the kind that is included in this book to learn and discover physics in a new and exciting way.

Stephen Wolfram
Creator of *Mathematica*

Introduction

Computer algebra software has already had an important impact on the way physics is taught and research is performed. While computers cannot replace thinking, they significantly enhance problem-solving abilities of the scientist and student by eliminating the tedious mathematics. As computers become more powerful and more readily available, the scope of problems solved in both research and teaching tremendously expands.

This book serves as a guide for using the computer algebra program *Mathematica* for physics research and teaching. The flexibility of *Mathematica* to manipulate analytical, numerical, and graphical expressions will further broaden the scope of problems that the student and researcher can solve.

Physics is not a spectator sport. The best way to demonstrate *Mathematica* is by solving a variety of physics problems chosen to illustrate its ability to display the output in many forms. A significant asset of *Mathematica* is the ease with which results can be visualized. This feature brings physics problems "alive" so that the reader can interact and experiment with the solutions. One can change the parameters and immediately observe the consequences, thereby gaining deeper insight into the physics of the solution. Instead of being throughly exhausted by the burdensome mathematics required to obtain the answer, *Mathematica* enables us to focus our attention on understanding the solution.

HOW TO USE THIS BOOK

This book is intended for undergraduate students, graduate students, and practicing physicists who want to learn new *Mathematica* techniques for solving a general class of physics problems. For the student, we expect this text to be a supplement to the standard course texts in mechanics, electrodynamics, relativity, and quantum mechanics; the student should use this book to get ideas on how to use *Mathematica* to solve the problems assigned by the instructor. Since we cover the canonical problems from the core courses, the student can practice with our solutions, and then modify our solutions to solve the particular problems assigned. This should help the student move up the *Mathematica* learning curve quickly. This book is also suitable for a course designed to teach the applications of *Mathematica* to physicists.

As such, the design of this book is more like a reference book than a novel to be read cover to cover. Each problem is self-contained (up to user-defined functions discussed at the beginning of each chapter), so the reader can go immediately to the portion of the book that is relevant to his or her problem.

There are two sections of the book which we recommend all readers examine before trying the problems.

- The section in this preface on troubleshooting, so that the reader knows where to turn in case difficulties arise.
- The first chapter contains useful information about style, notation, and short-cuts that we will make use of throughout the book.

Note that we assume the reader is reasonably familiar with *Mathematica* (at least at the level of the tutorial in the *Mathematica* manual), so we focus on the physics applications and not on rudimentary *Mathematica* techniques.

The book consists of two levels of material. The few sections of each chapter are readily understandable by the undergraduate physics student. The latter portions of the chapters are intended for advanced undergraduate and graduate physics students, and cover a broader range of topics.

Each of the chapters 2 through 10 are divided into three parts:

- an introduction to the physics and *Mathematica* commands;
- solved problems that cover the standard ideas and methods found in the discipline;
- unsolved exercises.

As the reader gains insight into the power of symbolic computations, they can easily extend the techniques demonstrated here to go beyond these examples and explore more difficult problems.

ABOUT THE ELECTRONIC SUPPLEMENT

The *Mathematica* input code for the entire book is available in the electronic supplement so that you can begin working the examples immediately. We will also post tips and suggestions about using this book, extensions to other problems and related fields, and bug fixes (should we encounter any).

The electronic supplement is available as item number 0206-862 from MathSource(TM). MathSource is an on-line archive of *Mathematica* related materials contributed by Wolfram Research and *Mathematica* users around the world. You can reach this site at:

```
http://mathsource.wri.com/
```

The material is also available from the authors web sites at:

```
http://www.physics.smu.edu/~olness
```

and

```
http://darkwing.uoregon.edu/~phys600/
```

HOW TO USE THE ELECTRONIC SUPPLEMENT

To spare the reader tedious typing, we have put all the source code for each chapter into a single notebook file. This enables the reader to start solving problems immediately without re-typing the lengthy commands. Furthermore, if there is any confusion about what input we used to generate these problems, you can simply cross-check with the source code.

For example, if you want to solve Problem 6 in Chapter 7, all you need do is to open the source code for chapter 7 (ch7.nb), execute the initialization cells, and then proceed directly to problem 6 (or whatever problem you choose) and begin working.

COMMUNICATION WITH THE AUTHORS

We welcome any comments and suggestions regarding this book. You may contact us via e-mail or regular mail at:

Robert Zimmerman
Institute of Theoretical Science
University of Oregon
Eugene, OR 97403
bob@zim.uoregon.edu

Fredrick Olness
Department of Physics
Southern Methodist University
Dallas, TX 75275-0175
olness@mail.physics.smu.edu

We welcome bug reports, and will post such information on the MathSource server (should we encounter any). However, we are unable to offer any help debugging problems specific to the compatibility of different *Mathematica* versions or hardware installations; for this type of assistance, we must refer you to the previous section on Troubleshooting, or to your local system manager.

ACKNOWLEDGMENTS

It is our pleasure to acknowledge the people who have contributed to this project. We thank Fumitaka Umewaka whose masters thesis, "Applications of *Mathematica* in Teaching Physics," at the University of Oregon, initiated this project. We thank Pearson Education Japan for publishing the Japanese edition of this book. A special thanks to John G. Cramer of the University of Washington, who updated many of the commands in the first edition. We thank our many colleagues at the University of Oregon and Southern Methodist University, too numerous to name, who have supported the development of this book. We also extend our gratitude to the graduate students at these institutions who have tested solutions to several of the problems and to Lester E. Matson for reading some of

the *Mathematica* files and making constructive suggestions. For the second edition, we would also like to thank the many referees who carefully read the manuscript and provided valuable feedback, in particular, John Cramer and Paul Abbott, who provided many useful comments.

Fred Olness would like to thank Davison E. Soper of the University of Oregon for introducing him to *Mathematica*. He would like to thank the Lightner-Sams Foundation for support, and Southern Methodist University for an Instructional Technology Grant. Fred Olness also acknowledges the U.S. Department of Energy for support of his high energy physics research; many techniques developed in the course of his research were incorporated in this book.

At Addison-Wesley, we would like to acknowledge the invaluable contributions by Adam Black, Nancy Benton, and Joan Marsh.

We also thank those who contributed to the first edition: Stuart Johnson (Physics Editor), Amy Willcutt, Jennifer Albanese, Nev Hanke, Eileen Hoff, and Laurie Petrycki.

At Wolfram Research, we thank Glenn Scholebo for elegantly converting the *Mathematica* notebooks into final typeset form for our first edition, and Stephen Wolfram for contributing the foreword.

At Integre Technical Publishing Company, we thank Don DeLand, Leslie Galen, Karen Couzin, and Lynn Ryan for their work on the second edition.

Troubleshooting

This manuscript was generated directly from a *Mathematica* notebook to minimize the possibility of introducing errors. The draft manuscript was written in *Mathematica*, and then converted to Tex using the `TexSave` feature. Final formatting was performed using the Tex source.

The final version of the *Mathematica* code was run using version 4.1, but is fully compatible with all *Mathematica* versions back to 3.0.

There are subtle differences between different versions and implementations of *Mathematica*. This means that on occasion the reader will find that some of the examples presented in this book need to be slightly modified to adapt to your particular version. We have tested these examples extensively on different platforms and with different versions to ensure that they are robust. However, we list below the most common difficulties that the reader is likely to encounter.

In our examples, we have been careful to present enough intermediate output so that the reader can cross-check their results for consistency. This allows the reader to isolate any differences that may arise, and find the cause quickly.

Possible TEX Conversion Errors

The typesetting features of *Mathematica* significantly enhance the user's ability to read and understand the *Mathematica* output. Unfortunately, this also means that the process of converting the *Mathematica* code to TEX is significantly more complex. While we have been very careful to avoid errors, there is the possibility that some characters are corrupted in the conversion process. Unfortunately, one of the more common errors is to have missing braces "{" and "}". For this reason, in part, we have posted the complete set of input files on the web. Should you encounter a "mysterious" problem that you suspect may be linked to a typographic error, you can use the input source code (which has not had any conversion) to ensure you have the correct expression. In fact, we recommend you use the input source code from the web in general to save yourself the effort of re-typing our expressions.

Order of Roots in `Solve`

A significant difference between different *Mathematica* versions is the order of the solutions returned by `Solve` and `DSolve`. Throughout the book, you will note we are careful to select the desired root using the `[[i]]` notation. If you have difficulty with the problems, this is one of the first places to look. Compare the intermediate output displayed in the book with your results. Use this information to isolate the problem, and determine if it is due to the ordering of the roots of the `Solve` command.

Debugging Techniques

To save paper (and trees), we have not displayed intermediate output for all expressions. Additionally, we have not displayed intermediate graphics output using the `$DisplayFunction->Identity` command. While this is a good practice when writing a book, when we were developing the problems we did display this intermediate output to help guide us through the problem; only after we obtained our solution did we go back and "clean up" the output.

How to Debug `Modules` and User-Defined Functions

In Chapter 3, Section 2, Problem 5, (Understanding the User-defined procedure `smallOsc`) we discuss how to redefine a `Module` so that the variables are `Global` and can be examined. This allows the reader to debug a `Module` program by stepping through each statement of the `Module` individually to try and isolate the error. This is a very useful debugging technique.

Commands That Never Return an Answer

In cases where the *Mathematica* command takes a long time (more than a minute on a ~1 GHz Pentium 4) to return an answer, we have indicated this in the text. If you wait a long time, get no output, and suspect a problem, there are many ways to approach this problem. The most obvious is to break the procedure up into smaller steps. We often group commands together to shorten our solution. We must confess that when we initially solved the problem, we actually performed each step individually, examined the output, and afterwards decided the most efficient set of steps to use and display.

Avoid the `FullSimplify` Command

In particular, if you are having trouble with lengthy expressions, try to avoid the `FullSimplify` command. Until you know the scope of your problem, you will find that the `Expand`, `Together` and ordinary `Simplify` command are much more efficient at reducing the answer. Once you have massaged the result into something *Mathematica* can swallow in a single bite, then you may want to try and `FullSimplify` the result.

Animation

The details of animating a series of graphics varies widely with different implementations of *Mathematica*. In this text we simply show the reader how to generate the sequence of graphics. The reader must refer to the computer specific user guide for the details of displaying the animation.

Clearing Variables

If *Mathematica* is yielding unusual results, a common cause is that there are variable definitions (possibly from a previous problem) that are conflicting with the assumed definitions for the present problem. The command `Clear["Global`*"]` will solve most of these problems. In fact, each chapter of this book was initially a single *Mathematica* notebook.

Using the `Clear["Global`*"]` command, we were able to run each chapter as a single *Mathematica* session from start to finish. (Again, this helped us eliminate errors, and verify that our examples were correct.) While this command should solve most all problems of this type, it may be worth restarting the kernel if mysterious problems still persist.

WHAT'S NEW WITH *MATHEMATICA* IN THE SECOND EDITION

The first edition of our text was produced with *Mathematica* version 2.2. The current version of *Mathematica* has dramatically changed since that time.

- Typesetting. Beginning with version 3.0, *Mathematica* introduced typeset input and output. This significantly shortens the length of the output (compare with our first edition); more importantly, it makes the input and output much more readable. Expressions that previously would span multiple pages (or fill the computer screen) are now succinctly displayed.
- *Mathematica* has greatly expanded its knowledge base. In the first edition, we often had to help *Mathematica* with complex integrals or differential equations. In this edition, *Mathematica* is fully capable of evaluating such expressions.
- For our first edition in 1995, it took about an hour on a "modern" computer to execute a chapter. With improved *Mathematica* algorithms and improved CPU's, an entire chapter runs in a few minutes.

WHAT'S NEW WITH THIS BOOK IN THE SECOND EDITION

In this second edition we have significantly expanded the number and variety of problems. In particular, we have expanded the book from seven to ten chapters; the three additional chapters are on Nonlinear Systems and Chaos, Discrete Systems, and Chaos and Orbiting Bodies. New problems and exercises have also been added to all chapters.

Contents

1 ■ Getting Started 1

1.1 Introduction 1
1.2 Arithmetic and Algebra 4
1.3 Functions and Procedures 14
1.4 Packages 23
1.5 Calculus 27
1.6 Graphics 32
1.7 Exercises 39

2 ■ General Physics 44

2.1 Introduction 44
2.2 Newtonian Mechanics in Inertial Frames 44
2.3 Newtonian Mechanics in Rotating Frames 77
2.4 Electricity and Magnetism 93
2.5 Modern Physics 116
2.6 Exercises 127

3 ■ Oscillating Systems 130

3.1 Introduction 130
3.2 Linear Oscillations 131
3.3 Small Oscillations 157
3.4 Oscillating Circuits 182
3.5 Exercises 191

4 ■ Nonlinear Oscillating Systems 193

4.1 Introduction 193
4.2 Nonlinear Pendulum 195
4.3 Duffing Equation 232
4.4 Exercises 255

5 ■ Discrete Dynamical Systems 258
5.1 Introduction 258
5.2 Logistic Map 260
5.3 Other Maps 276
5.4 Fractals 291
5.5 Exercises 296

6 ■ Lagrangians and Hamiltonians 298
6.1 Introduction 298
6.2 Lagrangian Problems without Lagrange Multipliers 299
6.3 Lagrangian Problems with Lagrange Multipliers 335
6.4 Hamiltonian Problems 344
6.5 Hamilton-Jacobi Problems 365
6.6 Exercises 375

7 ■ Orbiting Bodies 377
7.1 Introduction 377
7.2 The Two-Body Problem 378
7.3 Restricted Three-Body Problem 396
7.4 Exercises 433

8 ■ Electrostatics 435
8.1 Introduction 435
8.2 Point Charges, Multipoles, and Image Charges 437
8.3 Laplace's Equation in Cartesian and Cylindrical Coordinates 467
8.4 Laplace's Equation in Spherical Coordinates 488
8.5 Exercises 511

9 ■ Quantum Mechanics 513
9.1 Introduction 513
9.2 One-Dimensional Schrödinger's Equation 515
9.3 Three-Dimensional Schrödinger's Equation 549
9.4 Exercises 576

10 ■ Relativity and Cosmology 578
10.1 Introduction 578
10.2 Special Relativity 579
10.3 General Relativity 597
10.4 Cosmology 621
10.5 Exercises 639

Index 643

CHAPTER

1 Getting Started

1.1 ■ INTRODUCTION

1.1.1 ■ Computers as a Tool

The solution of realistic physics problems is often hampered because the algebra is too complex for anyone but the dedicated researcher. Just as the calculator eliminated laborious numerical computations, symbolic software programs eliminate arduous algebraic computations. While computer power is no substitute for thinking, it spares the scientist from performing mundane mathematical steps, and thereby frees time for creative thinking. The scientist is able to explore complex relationships among quantities, ask "What if ... ?", and obtain an immediate answer. *Mathematica* is only one of the popular systems for doing such calculations; other systems include Maple, Derive, Axiom, Macsyma, and Reduce.

Because there are many ways to solve physics problems, we present a variety of styles to illustrate different ways of solving similar problems using *Mathematica*. We emphasize that the solutions presented here are not necessarily the most efficient for dealing with all possible instances. While we do discuss writing efficient *Mathematica* code, we sometimes sacrifice the most elegant or efficient solution in favor of one that is most easily understood pedagogically. For example, we introduce some user-defined procedures to automate repetitive tasks in an intuitive way. Had our goal been to write an efficient "black-box" program that could handle all possible inputs in an error-free manner, the routines would be less pedagogical, more complex to read, and encumbered with additional code to trap error conditions. Where appropriate, we prompt the reader to expand upon our solutions.

Each chapter has a short overview of the major physics and mathematics topics emphasized in the chapter. The problems are chosen to cover a broad range of physics problems and to illustrate a variety of *Mathematica* procedures. Exercises are included at the end of the chapter to reinforce the techniques developed in the examples, and to suggest additional applications not covered.

Our goal is to focus on the *Mathematica* techniques that are most appropriate for solving physics problems; thus, we assume the reader is familiar with the most basic features of *Mathematica* as discussed in *The Mathematica Book*, Fourth Edition, by Stephen Wolfram. At a minimum, the reader should be familiar with the *Tour of Mathematica*, which is contained in *The Mathematica Book*.

1.1.2 ■ A Note about Notation and Style

Here we make a few notes about what notation we do and do not use throughout the text, as well as some stylistic issues.

1. We do not use subscript notation for variables. Although this often makes the output more readable, we encounter some very subtle features when using subscript notation. Therefore, our preference is to create variables in the form {x1, x2, x3, ...} rather than $\{x_1, x_2, x_3, \ldots\}$. We explain this in more detail in the next section.
2. Initially we will type certain symbols in their expanded form. For example, the symbol "->" is equivalent to "→"; however, we will initially enter this expression in the format "->" so the beginning *Mathematica* user knows exactly what to type.
3. We often use special characters, including letters from the Greek alphabet. These can be entered using the palettes, or using keyboard short-cuts. For example, the character θ can be entered by typing ⟨esc⟩q⟨esc⟩, or by typing ⟨esc⟩\theta⟨esc⟩, where ⟨esc⟩ represents the escape key. The latter form may be more familiar to those who use the TEX typesetting program.
4. In general, we avoid introducing short-hand definitions for commands with a lengthy name. While we often use such short-cuts in our personal work, doing so here makes it difficult for the reader who must first look up our definition for our short-cut, and then look up the *Mathematica* command. This philosophy is good practice when you are sharing your notebooks with others. In place of using short-cut commands to save typing, *Mathematica* has a very useful menu feature called **Complete Selection**. Refer to the description of this command in the Help Browser.
5. We will use the semicolon ";" to suppress unwanted intermediate text and graphics output. While this makes the book more compact, when you are initially solving a problem you probably will want to view the intermediate output to check for errors.
6. Our preference is to turn off the automatic spell checking within *Mathematica*. Again, this is helpful for debugging, but for a finished product (such as this book), it is not necessary.

```
In[1]:= Off[General :: spell1];
        Off[General :: spell];
```

1.1.3 ■ Notation and Symbols (For Experts Only)

Here we explain some of the subtle features of *Mathematica* notation. Beware: you should read this section only if you are an experienced *Mathematica* user, *and* need to use subscript notation for variables, e.g., $\{x_1, x_2, x_3, \ldots\}$. Otherwise, you should immediately skip to the next section.

We include this section here so that if you encounter strange behavior when trying to use subscript notation for variables, you will know where to look for explanations. This section also uses some commands that are not introduced until later in this chapter; therefore, you may find it best to come back to this section after reading the rest of the chapter.

Subscript notation for variables is tricky because you can have two expressions that have identical output display formats, but they can in fact have very different internal forms.

Debugging *Mathematica* code with such expressions becomes complicated, and this is why, in general, we do not use subscript notation in this book.

For example, let us try to use the symbol x_1 as an independent variable. If we enter this symbol using the basic input palette or with a keyboard short-cut, we obtain

```
In[2]:= x₁//FullForm
Out[2]= Subscript[x, 1]
```

When we try to use such notation in a simple expression, we are in for some surprises. The following expression looks simple enough.

```
In[3]:= term = a x₁ + b x₂ + c x
Out[3]= c x + a x₁ + b x₂
```

But, if we try to make a substitution for the variable x (without any subscript), which we intend to be a variable independent of x_1, we obtain

```
In[4]:= term /. {x → 1}
Out[4]= c + a 1₁ + b 1₂
```

The result is obviously nonsense.

Next, we make use of the **Symbolize** function contained in the package **Utilities`Notation`**.

```
In[5]:= << Utilities`Notation`

In[6]:= ?Symbolize
"Symbolize[boxrs] forces any box structure matching boxrs to be
treated internally as a single symbol anywhere it appears in
an input expression."
```

However, when we try this command in a simple manner, the result is an error.

```
In[7]:= Symbolize[x₁]
Symbolize :: badSymbolizeBoxes : The Symbolize boxes x do not have an em-
bedded NotationBoxTag TagBox. The Symbolize statement Symbolize[x ] may not
have been entered using the palette, or the embedded TagBox may have been
deleted. The embedded TagBox ensures correct parsing and retention of proper
styling and grouping information. 1 1
Out[7]= $Failed
```

If, instead, we use the **Symbolize** command from the Notation Palette, we get a variable that appears to be identical to x_1 defined above, but has a very different internal form.

```
In[8]:= x₁ //FullForm
        (* Entered from the Notation Palette *)
Out[8]= NotationBoxTag[SubscriptBox["x", "1"]]
```