

Graduate Texts in Physics

Andrey Grozin

Introduction to Mathematica[®] for Physicists

 Springer

Graduate Texts in Physics

For further volumes:
<http://www.springer.com/series/8431>

Graduate Texts in Physics

Graduate Texts in Physics publishes core learning/teaching material for graduate- and advanced-level undergraduate courses on topics of current and emerging fields within physics, both pure and applied. These textbooks serve students at the MS- or PhD-level and their instructors as comprehensive sources of principles, definitions, derivations, experiments and applications (as relevant) for their mastery and teaching, respectively. International in scope and relevance, the textbooks correspond to course syllabi sufficiently to serve as required reading. Their didactic style, comprehensiveness and coverage of fundamental material also make them suitable as introductions or references for scientists entering, or requiring timely knowledge of, a research field.

Series Editors

Professor William T. Rhodes
Department of Computer and Electrical Engineering and Computer Science
Imaging Science and Technology Center
Florida Atlantic University
777 Glades Road SE, Room 456
Boca Raton, FL 33431
USA
wrhodes@fau.edu

Professor H. Eugene Stanley
Center for Polymer Studies Department of Physics
Boston University
590 Commonwealth Avenue, Room 204B
Boston, MA 02215
USA
hes@bu.edu

Professor Richard Needs
Cavendish Laboratory
JJ Thomson Avenue
Cambridge CB3 0HE
UK
m11@cam.ac.uk

Professor Martin Stutzmann
Technische Universität München
Am Coulombwall
85747 Garching, Germany
stutz@wsi.tu-muenchen.de

Professor Susan Scott
Department of Quantum Science
Australian National University
ACT 0200, Australia
susan.scott@anu.edu.au

Andrey Grozin

Introduction to Mathematica[®] for Physicists

 Springer

Andrey Grozin
Theory Division
Budker Institute of Nuclear Physics
Novosibirsk, Russia

ISSN 1868-4513
ISBN 978-3-319-00893-6
DOI 10.1007/978-3-319-00894-3
Springer Cham Heidelberg New York Dordrecht London

ISSN 1868-4521 (electronic)
ISBN 978-3-319-00894-3 (eBook)

Library of Congress Control Number: 2013941732

© Springer International Publishing Switzerland 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Computer algebra systems are widely used in pure and applied mathematics, physics, and other natural sciences, engineering, economics, as well as in higher and secondary education (see, e.g., [1–5]). For example, many important calculations in theoretical physics could never be done by hand, without wide use of computer algebra. Polynomial or trigonometric manipulations using paper and pen are becoming as obsolete as school long division in the era of calculators.

There are several powerful general-purpose computer algebra systems. The system *Mathematica* is most popular. It contains a huge amount of mathematical knowledge in its libraries. The fundamental book on this system [6] has more than 1,200 pages. Fortunately, the same information (more up-to-date than in a printed book) is available in the help system and hence is always at the fingertips of any user. Many books about *Mathematica* and its application in various areas have been published; see, for example, the series [7–10] of four books (each more than 1,000 pages long) or [11]. The present book does not try to replace these manuals. Its first part is a short systematic introduction to computer algebra and *Mathematica*; it can (and should) be read sequentially. The second part is a set of unrelated examples from physics and mathematics which can be studied selectively and in any order. Having understood the statement of a problem, try to solve it yourself. Have a look at the book to get a hint only when you get stuck. Explanations in this part are quite short.

This book¹ is a result of teaching at the physics department of Novosibirsk State University. Starting from 2004, the course “Symbolic and numeric computations in physics applications” is given to students preparing for M.Sc., and an introduction to *Mathematica* is the first part of this course (the second part is mainly devoted to Monte Carlo methods). Practical computer classes form a required (and most important) part of the course. Most students have no problems with mastering the basics of *Mathematica* and applying it to problems in their own areas of interest.

The book describes *Mathematica* 9. Most of the material is applicable to other versions too. The *Mathematica* Book (fifth edition) [6], as well as, e.g., the book

¹ Work partially supported by the Russian Ministry of Education and Science.

series [7–10], describes *Mathematica* 5. The main source of up-to-date information is the *Mathematica* Help system.

The whole book (except Lecture 1 and Problems for students) consists of *Mathematica* notebooks. They can be found at

<http://www.inp.nsk.su/~grozin/mma/mma.zip>

The zip file is password protected. The password is the last sentence of Lecture 7 (case-sensitive, including the trailing period). The reader is encouraged to experiment with these notebook files. In the printed version of the book, plots use different curve styles (dashed, dotted, etc.) instead of colors.

The book will be useful for students, Ph.D. students, and researchers in the area of physics (and other natural sciences) and mathematics.

Novosibirsk, Russia

Andrey Grozin

Contents

Preface	v
Part I Lectures	
1 Computer Algebra Systems	3
2 Overview of <i>Mathematica</i>	9
2.1 Symbols	9
2.2 Numbers	10
2.3 Polynomials and Rational Functions	10
2.4 Elementary Functions	12
2.5 Calculus	13
2.6 Lists	14
2.7 Plots	15
2.8 Substitutions	17
2.9 Equations	18
3 Expressions	21
3.1 Atoms	21
3.2 Composite Expressions	22
3.3 Queries	24
3.4 Forms of an Expression	25
4 Patterns and Substitutions	27
4.1 Simple Patterns	27
4.2 One-Shot and Repeated Substitutions	28
4.3 Products	29
4.4 Sums	31
4.5 Conditions	32
4.6 Variable Number of Arguments	33

5	Functions	35
5.1	Immediate and Delayed Assignment	35
5.2	Functions	36
5.3	Functions Remembering Their Values	36
5.4	Fibonacci Numbers	37
5.5	Functions from Expressions	38
5.6	Antisymmetric Functions	39
5.7	Functions with Options	40
5.8	Attributes	40
5.9	Upvalues	41
6	Mathematica as a Programming Language	43
6.1	Compound Expressions	43
6.2	Conditional Expressions	44
6.3	Loops	46
6.4	Functions	47
6.5	Local Variables	49
6.6	Table	50
6.7	Parallelization	50
6.8	Functions with an Index	51
6.9	Hold and Evaluate	51
7	Gröbner Bases	55
7.1	Statement of the Problem	55
7.2	Monomial Orders	55
7.3	Reduction of Polynomials	56
7.4	S-Polynomials	57
7.5	Buchberger Algorithm	58
7.6	Is the System Compatible?	59
7.7	Gröbner Bases with Respect to Lexicographic Order	60
7.8	Is the Number of Solutions Finite?	61
8	Calculus	63
8.1	Series	63
8.2	Differentiation	67
8.3	Integration	68
8.4	Summation	70
8.5	Differential Equations	70
9	Numerical Calculations	73
9.1	Approximate Numbers in <i>Mathematica</i>	73
9.2	Solving Equations	76
9.3	Numerical Integration and Summation	77
9.4	Differential Equations	78

10	Risch Algorithm	79
	10.1 Rational Functions	79
	10.2 Logarithmic Extension	80
	10.3 Exponential Extension	85
	10.4 Elementary Functions	89
11	Linear Algebra	91
	11.1 Constructing Matrices	91
	11.2 Parts of a Matrix	92
	11.3 Queries	93
	11.4 Operations with Matrices and Vectors	93
	11.5 Eigenvalues and Eigenvectors	95
	11.6 Jordan Form	96
	11.7 Symbolic Vectors, Matrices, and Tensors	97
12	Input–Output and Strings	99
	12.1 Reading and Writing .m Files	99
	12.2 Output	101
	12.3 C, Fortran, and TeX Forms	102
	12.4 Strings	102
13	Packages	105
	13.1 Contexts	105
	13.2 Packages	106
	13.3 Writing Your Own Package	106
Part II Computer Classes		
14	Plots	111
	14.1 2D Plots	111
	14.2 3D Plots	120
15	Trigonometric Functions	125
16	Quantum Oscillator	127
	16.1 Lowering and Raising Operators	127
	16.2 Ground State	129
	16.3 Excited States	129
	16.4 Some Properties	131
17	Spherical Harmonics	133
	17.1 Angular Momentum in Quantum Mechanics	133
	17.2 $Y_{ll}(\theta, \varphi)$	134
	17.3 $Y_{lm}(\theta, \varphi)$	135

18	Adding Angular Momenta in Quantum Mechanics	139
19	Classical Nonlinear Oscillator	145
	19.1 Statement of the Problem	145
	19.2 The First Correction	146
	19.3 The Second Correction	147
	19.4 The n th Correction	149
20	Quantum Nonlinear Oscillator	153
	20.1 Perturbation Theory	153
	20.2 Nonlinear Oscillator	154
	20.3 Energy Levels	154
	20.4 Correspondence Principle	157
	20.5 States	159
21	Riemann Curvature Tensor	161
22	Multi-ζ Functions	165
	22.1 Definition	165
	22.2 Stuffing Relations	165
	22.3 Integral Representation	167
	22.4 Shuffling Relations	168
	22.5 Duality Relations	170
	22.6 Weight 4	170
	22.7 Weight 5	171
23	Rainbow	173
	23.1 Statement of the Problem	173
	23.2 0 Ray Segments Inside the Drop	174
	23.3 1 Ray Segment Inside the Drop	179
	23.4 2 Ray Segments Inside the Drop	181
	23.5 L Ray Segments Inside the Drop	188
24	Cyclohexane	193
	24.1 Statement of the Problem	193
	24.2 First Steps	194
	24.3 Equations	196
	24.4 Projection onto the x, y Plane	198
	24.5 Complete Analysis of the Solutions	205
	24.6 Shape of the Molecule	206
25	Problems for Students	209
	References	213
	Index	215

Part I

Lectures

Catching a lion, the computer-algebra method: catch a cat, put it into the cage and lock it; then substitute a lion for the cat.

Chapter 1

Computer Algebra Systems

First attempts to use computers for calculations not only with numbers but also with mathematical expressions (e.g., symbolic differentiation) were made in the 1950s. In the 1960s research in this direction became rather intensive. This area was known under different names: symbolic calculations, analytic calculations, and computer algebra. Recently this last name is most widely used. Why algebra and not, say, calculus? The reason is that it is most useful to consider operations usually referred to calculus (such as differentiation) as algebraic operations in appropriate algebraic structures (differential fields).

First universal (i.e., not specialized for some particular application area) computer algebra systems appeared at the end of the 1960s. Not many such systems have been constructed; they are shown in the Table 1.1. Creating a universal computer algebra system is a huge amount of work, at the scale of hundreds of man-years. Some projects of this kind were not sufficiently developed and subsequently died; they are not shown in Table 1.1.

Table 1.1 Universal computer algebra systems

System	Year	Implementation language	Current name	Status
REDUCE	1968	Lisp		Free (BSD)
Macsyma	1969		Maxima	Free (GPL)
Scratchpad	1974		Axiom OpenAxiom FriCAS	Free (BSD)
muMATH	1979		Derive	Dead
Maple	1983			Proprietary
<i>Mathematica</i>	1988	C, C++		Proprietary
MuPAD	1992		MATLAB symbolic toolbox	Proprietary

Theoretical physicist A. Hearn (known to specialists for the Drell–Hearn sum rule) has written a Lisp program REDUCE to automatize some actions in

calculating Feynman diagrams. It quickly grew into a universal system. At first, it was distributed free (it was sufficient to ask for Hearn's permission) and became widely used by physicists. Later it became commercial. At the end of 2008 it has become free, with a modified BSD license.

Macsyma was born in the MAC project at MIT (1969), the name means MAC SYmbolic MANipulator. The project has nothing to do with Macintosh computers, which appeared much later. Its name had several official meanings (Multiple-Access Computer, Man And Computer, Machine Aided Cognition) and some unofficial ones (Man Against Computer, Moses And Company, Maniacs And Clowns, etc.). The work was done on a single PDP-6, later PDP-10 computer (about 1 MByte memory; there were no bytes back then, but 36-bit words). One of the first time-sharing operating systems, ITS, was written for this computer, and many users at once worked on it interactively. Later this computer became one of the first nodes of ARPANET, the ancestor if Internet, and users from other universities could use Macsyma.

The company Symbolics was spun off MIT. It produced Lisp machines—computers with a hardware support of Lisp, as well as software for these computers, including Macsyma—the largest Lisp program at that time. Later production of Lisp machines became unprofitable, because general-purpose workstations (Sun, etc.) became faster and cheaper. Symbolics went bankrupt; Macsyma business was continued by Macsyma Inc., who sold Macsyma for a number of platforms and operating systems. Its market share continued to shrink because of the success of Maple and *Mathematica*, and finally the company was sold in 1999 to Andrew Topping. The new owner stopped Macsyma development and marketing. Then he died, and the rights to the commercial Macsyma now belong to his inheritors. All efforts spent on improving this branch of Macsyma are irreversibly lost.

Fortunately, this was not the only branch. Macsyma development at MIT was largely funded by DOE, and MIT transferred this codebase to DOE who distributed it. This version was ported to several platforms. All these ports died except one. Professor William Schelter ported DOE Macsyma to Common Lisp, the new Lisp standard, and developed this version until he died in 2001. This version was called Maxima, to avoid trademark problems. In 1998 he obtained permission from DOE to release Maxima under GPL. He also developed GCL (GNU Common Lisp). Currently Maxima is an active free software project and works on many Common Lisp implementations.

Macsyma has played a huge role in the development of computer algebra systems. It was the first system in which modern algorithms for polynomials, integration in elementary functions, etc., were implemented (REDUCE and Macsyma influenced each other strongly and are rather similar to each other). Macsyma was designed as an interactive system. For example, if the form of an answer depends on the sign of a parameter, it will ask the user

Is a positive or negative?

Scratchpad was born in IBM research laboratories (1974). At first it did not differ from other systems (Macsyma, REDUCE) very much and borrowed chunks of code from them. It was radically redesigned in the version Scratchpad II (1985).

And this design, perhaps, still remains the most beautiful one from a mathematical point of view. It is a strongly typed system (the only one among universal computer algebra systems). Any object (formula) in it belongs to some domain (e.g., it is a single-variable polynomial with integer coefficients). Each domain belongs to some category (e.g., it is a ring, or a commutative group, or a totally ordered set). New domains can be constructed from existing ones. For example, a matrix of elements belonging to any ring can be constructed. It is sufficient to program a matrix multiplication algorithm once. This algorithm calls the operations of addition and multiplication of the elements. If matrices of rational numbers are being multiplied, then addition and multiplication of rational numbers are called; and if matrices of polynomials—then addition and multiplication of polynomials.

Scratchpad was never distributed to end users by IBM. At last, IBM decided to stop wasting money for nought (or for basic research) and sold Scratchpad II to the English company NAG (famous for its numerical libraries). It marketed this system under the name Axiom. However, the product did not bring enough profit and was withdrawn in 2001. Axiom development took about 300 man-years of work of researchers having highest qualification. All this could easily disappear without a trace. Fortunately, one of old-time Scratchpad II developers at IBM, Tim Daly, has succeeded in convincing NAG to release Axiom under the modified BSD license. Now it is a free software project and still the most beautiful system from mathematical point of view. But unfortunately, due to incompatible visions of the directions of the future development, two forks appeared—OpenAxiom and FriCAS. And it is not clear which one is better.

muMATH (Soft Warehouse, Hawaii, 1979) got to the list of universal computer algebra systems with some stretch. It was written for microprocessor systems with a very limited memory (later called personal computers); mu in its name, of course, means μ , i.e., micro. This system never implemented advanced modern algorithms. It used heuristic methods instead, as taught in university calculus courses: let's try this and that, and if you can't get it, you can't get it. But it was surprisingly powerful at its humble size. The system has been essentially rewritten in 1988 and got a menu interface, graphics, and the new name, Derive. Then Soft Warehouse was bought by Texas Instruments, who presented a calculator with a (Derive-based) computer algebra system in 1995. Derive was withdrawn from market in 2007.

All these systems can be referred to the first generation. They are all written in various dialects of Lisp. They were considered related to the area of artificial intelligence.

The first representative of the second generation is the Canadian system Maple. It has a small kernel written in C, which implements an interpreted procedural language convenient for writing computer algebra algorithms. The major part of its mathematical knowledge is contained in the library written in this language. Maple can work on many platforms. It quickly became popular. In 2009 Maplesoft (Waterloo Maple Inc.) has been acquired by the Japanese company Cybernet Systems Group; development of Maple is not affected. By the way, numerical program MathCAD used a cut-down version of Maple to provide some computer algebraic capabilities.

In the beginning of the 1980s, a young theoretical physicist Steven Wolfram, an active Macsyma user, together with a few colleagues, has written a system SMP (Symbolic Manipulation Program). The project was a failure (I still have a huge SMP manual sent to me by S. Wolfram). After that, he understood what mass users want—they want a program to look pretty. He, together with a few colleagues, has rewritten the system, paying a lot of attention to the GUI and graphics (the symbolic part was largely based on SMP). The result was *Mathematica*, version 1 (1988). And Wolfram got his first million in three months of selling it.

Mathematica heavily relies on substitutions. Even a procedure call is a substitution. Pattern matching and their replacing by right-hand sides of substitutions are highly advanced in *Mathematica*. Often a set of mathematical concepts can be easily and compactly implemented via substitutions. On the other hand, this can lead to inefficiency: pattern matching is expensive.

The latest arrival in the list of universal computer algebra systems is MuPAD (its name initially meant Multi-Processor Algebra Data tool, and indeed early versions contained experimental support of multiprocessor systems, which later disappeared). The system was designed and implemented by a research group at the University of Paderborn in Germany (this is one more meaning of PAD in the name) in 1992 and later was distributed commercially by the company SciFace. Initially, MuPAD was quite similar to Maple. Later it borrowed many ideas from Axiom (domains, categories; however, MuPAD is dynamically typed). During a long period, it was allowed to download and use MuPAD Light for free; it had no advanced GUI, but its symbolic functionality was not cut down. Funding of the University project was stopped in 2005; in 2008, SciFace was bought by Mathworks, the makers of MATLAB. After that, MuPAD is available only as a MATLAB add-on.

It seems that *Mathematica* dominates the market of commercial computer algebra systems, with Maple being number two. *Mathematica* is highly respected for the huge amount of mathematical knowledge accumulated in its libraries. It is not bug-free (this is true for all systems). Often it requires more resources (memory, processor time) for solving a given problem than other systems. But it is very convenient and allows a user to do a lot in a single framework.

In addition to universal systems, there are a lot of specialized computer algebra systems. Here we'll briefly discuss just one example important for theoretical physics.

In the 1960s, a well-known Dutch theoretical physicist M. Veltman, a future Nobel prize winner, has written a system Schoonschip in the assembly language of CDC-6000 computers (in Dutch Schoonschip means “to clean a ship,” in a figurative sense “to put something in order,” “to throw unneeded things overboard”). This system was designed for handling very long sums (millions of terms) whose size can be much larger than the main memory and is limited only by the available disk space. All operations save one are local: they are substitutions which replace a single term by several new ones. The system gets a number of terms from the disk, applies the substitution to them, and puts the results back to the disk. The only unavoidable nonlocal operation is collecting similar terms; it is done with advanced disk sorting algorithms. Built-in mathematical knowledge of the system is very limited; the

user has to program everything from scratch. Many nontrivial algorithms, such as polynomial factorization, are highly nonlocal and impossible to implement. On the other hand, this was the only system which could work with very large expressions, orders of magnitude larger than in other systems. Later Schoonschip was ported to IBM-360 (in PL/I; you can guess that this was not done by Veltman :-). Then Veltman has rewritten it from the CDC assembly language to the 680x0 assembly language. When 680x0-based personal computers (Amiga, Atari) became extinct, it became clear that something similar but more portable is needed.

In 1989 another well-known Dutch theoretical physicist, Vermaseren, has written (in C) a new system, Form. It follows the same ideology, but many details differ. It was distributed free of charge as binaries for a number of platforms; recently it became free software (GPL). Development of Form continues. A parallel version for multiprocessor computers and for clusters with fast connections now exists. Many important Feynman diagram calculations could never have been done without Schoonschip and later Form.

The percentage of theoretical physicists among authors of computer algebra systems is suspiciously high. Some of them remained physicists (and even got a Nobel prize); some completely switched to development of their systems (and even became millionaires).

In conclusion we'll discuss a couple of important computer algebra concepts. For some (sufficiently simple) classes of expressions an algorithm of reduction to a *canonical form* can be constructed. Two equal expressions reduce to the same canonical form. In particular, any expression equal to 0, in whatever form it is written, has the canonical form 0.

For example, it is easy to define a canonical form for polynomials of several variables with integer (or rational) coefficients: one has to expand all brackets and collect similar terms. What's left is to agree upon an unambiguous order of terms, and we have a canonical form (this can be done in more than one way).

It is more difficult, but possible, to define a canonical form for rational expressions (ratios of polynomials). One has to expand all brackets and to bring the whole expression to a common denominator (collecting similar terms, of course). However, this is not sufficient: one can multiply both the numerator and the denominator by the same polynomial and obtain another form of the rational expression. It is necessary to cancel the greatest common divisor (gcd) of the numerator and the denominator. Calculating polynomial gcd's is an algorithmic operation, but it can be computationally expensive. What's left is to fix some minor details—an unambiguous order of terms in both the numerator and the denominator and, say, the requirement that the coefficient of the first term in the denominator is 1, and we obtain a canonical form.

A *normal form* for a class of expressions satisfies a weaker requirement: any expression equal to 0 must reduce to the normal form 0. For example, bringing to common denominator (without canceling gcd) defines a normal form for rational expressions.

For more general classes of expressions containing elementary functions, not only canonical but even normal form does not exist. Richardson has proved that it is algorithmically undecidable if such an expression is equal to 0.

2.2 Numbers

Mathematica can work with arbitrarily long integer numbers.

In[7] := Factorial[100]

Out[7] = 933262154439441526816992388562667004907159682643816214685\
9296389521759999322991560894146397615651828625369792082722375\
8251185210916864000000000000000000000000

When working with a rational number, the greatest common divisors of its numerator and denominator are canceled.

In[8] := a = 1234567890/987654321

Out[8] = $\frac{137174210}{109739369}$

Calculations with rational numbers are exact.

In[9] := a^5

Out[9] = 48569355286282885522765185491603110100000/
15915207065345784618237986236670245907849

How much is this numerically? Say, with 30 significant digits?

In[10] := N[a, 30]

Out[10] = 1.24999998860937500014238281250

Mathematica can work with real (floating-point) numbers having arbitrarily high precision.

In[11] := a = 1234567890987654321.1234567890987654321

Out[11] = 1.234567890987654321123456789098765432 $\times 10^{18}$

In[12] := a^5

Out[12] = 2.86797187177160567275921531725363508 $\times 10^{90}$

Here are π and e with 100 significant digits.

In[13] := N[Pi, 100]

Out[13] = 3.14159265358979323846264338327950288419716939937510582097\
4944592307816406286208998628034825342117068

In[14] := N[E, 100]

Out[14] = 2.71828182845904523536028747135266249775724709369995957496\
6967627724076630353547594571382178525166427

2.3 Polynomials and Rational Functions

Let's take a polynomial.

In[15] := a = (x + y + z)^6

Out[15] = $(x + y + z)^6$

Expand it.

In[16] := a = Expand[a]

$$\begin{aligned} \text{Out[16]} = & x^6 + 6x^5y + 15x^4y^2 + 20x^3y^3 + 15x^2y^4 + 6xy^5 + y^6 + 6x^5z + 30x^4yz + \\ & 60x^3y^2z + 60x^2y^3z + 30xy^4z + 6y^5z + 15x^4z^2 + 60x^3yz^2 + 90x^2y^2z^2 + \\ & 60xy^3z^2 + 15y^4z^2 + 20x^3z^3 + 60x^2yz^3 + 60xy^2z^3 + 20y^3z^3 + 15x^2z^4 + \\ & 30xyz^4 + 15y^2z^4 + 6xz^5 + 6yz^5 + z^6 \end{aligned}$$

The degree in x .

In[17] := Exponent[a,x]

$$\text{Out[17]} = 6$$

The coefficient of x^2 .

In[18] := Coefficient[a,x,2]

$$\text{Out[18]} = 15y^4 + 60y^3z + 90y^2z^2 + 60yz^3 + 15z^4$$

Collect terms with the same power of x together.

In[19] := Collect[a,x]

$$\begin{aligned} \text{Out[19]} = & x^6 + y^6 + 6y^5z + 15y^4z^2 + 20y^3z^3 + 15y^2z^4 + 6yz^5 + z^6 + x^5(6y + 6z) + \\ & x^4(15y^2 + 30yz + 15z^2) + x^3(20y^3 + 60y^2z + 60yz^2 + 20z^3) + \\ & x^2(15y^4 + 60y^3z + 90y^2z^2 + 60yz^3 + 15z^4) + \\ & x(6y^5 + 30y^4z + 60y^3z^2 + 60y^2z^3 + 30yz^4 + 6z^5) \end{aligned}$$

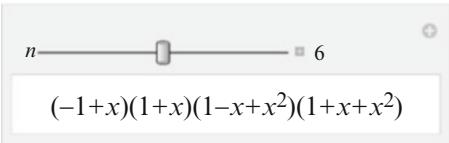
Factorize it.

In[20] := a = Factor[a]

$$\text{Out[20]} = (x + y + z)^6$$

Suppose we want to factorize polynomials $x^n - 1$ with various n . The parameter n can be varied from 2 to 10 by dragging the marker with the mouse.

In[21] := Manipulate[Factor[x^n - 1], {n, 2, 10, 1, Appearance -> "Labeled"}]

Out[21] = 

There exists an algorithm which completely factorizes any polynomial with integer coefficients into factors which also have integer coefficients.

In[22] := Factor[x^4 - 1]

$$\text{Out[22]} = (-1 + x)(1 + x)(1 + x^2)$$

If we want to get factors whose coefficients come from an extension of the ring of integers, say, by the imaginary unit i , we should say so explicitly.

In[23] := Factor[x^4 - 1, Extension -> I]

$$\text{Out[23]} = (-1 + x)(-i + x)(i + x)(1 + x)$$

This polynomial factorizes into two factors with integer coefficients.

In[24] := a = x^4 - 4; Factor[a]

$$\text{Out[24]} = (-2 + x^2)(2 + x^2)$$

If coefficients from the extension of the ring of integers by $\sqrt{2}$ are allowed—into three factors.

In[25] := Factor[a, Extension -> Sqrt[2]]

$$\text{Out[25]} = -(\sqrt{2} - x)(\sqrt{2} + x)(2 + x^2)$$

And if the ring of coefficients is extended by both $\sqrt{2}$ and i —into four factors.

In[26] := Factor[a, Extension->{Sqrt[2], I}]

$$\text{Out[26]} = -(\sqrt{2} - x)(\sqrt{2} - ix)(\sqrt{2} + ix)(\sqrt{2} + x)$$

And this is a rational function.

In[27] := (x^3 - y^3)/(x^2 - y^2)

$$\text{Out[27]} = \frac{x^3 - y^3}{x^2 - y^2}$$

It is not canceled by the greatest common divisor of its numerator and denominator; this should be done explicitly.

In[28] := Cancel[%]

$$\text{Out[28]} = \frac{x^2 + xy + y^2}{x + y}$$

(% means the result of the previous calculation). A sum of rational functions.

In[29] := a = x/(x + y) + y/(x - y)

$$\text{Out[29]} = \frac{y}{x - y} + \frac{x}{x + y}$$

Let's put it over the common denominator.

In[30] := a = Together[a]

$$\text{Out[30]} = \frac{x^2 + y^2}{(x - y)(x + y)}$$

Partial fraction decomposition with respect to x .

In[31] := Apart[a, x]

$$\text{Out[31]} = 1 + \frac{y}{x - y} - \frac{y}{x + y}$$

In[32] := Clear[a]

2.4 Elementary Functions

Mathematica knows some simple properties of elementary functions.

In[33] := Sin[-x]

$$\text{Out[33]} = -\text{Sin}[x]$$

In[34] := Cos[Pi/4]

$$\text{Out[34]} = \frac{1}{\sqrt{2}}$$

In[35] := Sin[5 * Pi/6]

$$\text{Out[35]} = \frac{1}{2}$$

In[36] := Log[1]

$$\text{Out[36]} = 0$$

In[37] := Log[E]

$$\text{Out[37]} = 1$$

In[38] := Exp[Log[x]]

$$\text{Out[38]} = x$$

In[39] := Log[Exp[x]]

Out[39] = $\text{Log}[e^x]$

And why not x ? Because this simplification is not always correct. Try to substitute $2\pi i$.

In[40] := Sqrt[0]

Out[40] = 0

In[41] := Sqrt[x]^2

Out[41] = x

In[42] := Sqrt[x^2]

Out[42] = $\sqrt{x^2}$

And why not x ? Try to substitute -1 .

In[43] := a = Sqrt[12 * x^2 * y]

Out[43] = $2\sqrt{3}\sqrt{x^2 y}$

This result can be improved, if we know that $x > 0$.

In[44] := Simplify[a, x > 0]

Out[44] = $2\sqrt{3}x\sqrt{y}$

And this is the case $x < 0$.

In[45] := Simplify[a, x < 0]

Out[45] = $-2\sqrt{3}x\sqrt{y}$

Expansion of trigonometric functions of multiple angles, sums, and differences:

In[46] := TrigExpand[Cos[2 * x]]

Out[46] = $\text{Cos}[x]^2 - \text{Sin}[x]^2$

In[47] := TrigExpand[Sin[x - y]]

Out[47] = $\text{Cos}[y]\text{Sin}[x] - \text{Cos}[x]\text{Sin}[y]$

The inverse operation—transformation of products and powers of trigonometric functions into linear combinations of such functions—is used more often. Let's take a truncated Fourier series.

In[48] := a = a1 * Cos[x] + a2 * Cos[2 * x] + b1 * Sin[x] + b2 * Sin[2 * x]

Out[48] = $a_1 \text{Cos}[x] + a_2 \text{Cos}[2x] + b_1 \text{Sin}[x] + b_2 \text{Sin}[2x]$

Its square is again a truncated Fourier series.

In[49] := TrigReduce[a^2]

Out[49] = $\frac{1}{2} (a_1^2 + a_2^2 + b_1^2 + b_2^2 + 2 a_1 a_2 \text{Cos}[x] + 2 b_1 b_2 \text{Cos}[x] + a_1^2 \text{Cos}[2x] - b_1^2 \text{Cos}[2x] + 2 a_1 a_2 \text{Cos}[3x] - 2 b_1 b_2 \text{Cos}[3x] + a_2^2 \text{Cos}[4x] - b_2^2 \text{Cos}[4x] - 2 a_2 b_1 \text{Sin}[x] + 2 a_1 b_2 \text{Sin}[x] + 2 a_1 b_1 \text{Sin}[2x] + 2 a_2 b_1 \text{Sin}[3x] + 2 a_1 b_2 \text{Sin}[3x] + 2 a_2 b_2 \text{Sin}[4x])$

2.5 Calculus

Let's take a function.

In[50] := f = Log[x^5 + x + 1] + 1/(x^5 + x + 1)

Out[50] = $\frac{1}{1 + x + x^5} + \text{Log}[1 + x + x^5]$

Calculate its derivative.

In[51] := g = D[f, x]

$$\text{Out[51]} = -\frac{1 + 5x^4}{(1 + x + x^5)^2} + \frac{1 + 5x^4}{1 + x + x^5}$$

Put over the common denominator.

In[52] := g = Together[g]

$$\text{Out[52]} = \frac{(1 + 5x^4)(x + x^5)}{(1 + x + x^5)^2}$$

A stupid integration algorithm would try to solve the fifth degree equation in the denominator, in order to decompose the integrand into partial fractions. *Mathematica* is more clever than that.

In[53] := Integrate[g, x]

$$\text{Out[53]} = \frac{1}{1 + x + x^5} + \text{Log}[1 + x + x^5]$$

Let's expand our function in x at 0 up to x^{10} .

In[54] := Series[f, {x, 0, 10}]

$$\text{Out[54]} = 1 + \frac{x^2}{2} - \frac{2x^3}{3} + \frac{3x^4}{4} - \frac{4x^5}{5} + \frac{11x^6}{6} - \frac{20x^7}{7} + \frac{31x^8}{8} - \frac{44x^9}{9} + \frac{32x^{10}}{5} + O[x]^{11}$$

Mathematica can calculate many definite integrals even when the corresponding indefinite integral cannot be taken. Here is an integral from 0 to 1.

In[55] := Integrate[Log[x]^2/(x + 1), {x, 0, 1}]

$$\text{Out[55]} = \frac{3 \text{Zeta}[3]}{2}$$

Mathematica knows how to sum many series.

In[56] := Sum[1/n^4, {n, 1, Infinity}]

$$\text{Out[56]} = \frac{\pi^4}{90}$$

Let's clear all the garbage we have generated—a very good habit.

In[57] := Clear[f, g]

2.6 Lists

We have already encountered this construct several times:

In[58] := a = {x, y, z}

$$\text{Out[58]} = \{x, y, z\}$$

This is a list. And here are its elements.

In[59] := a[[1]]

$$\text{Out[59]} = x$$

In[60] := a[[2]]

$$\text{Out[60]} = y$$

In[61] := a[[3]]

$$\text{Out[61]} = z$$

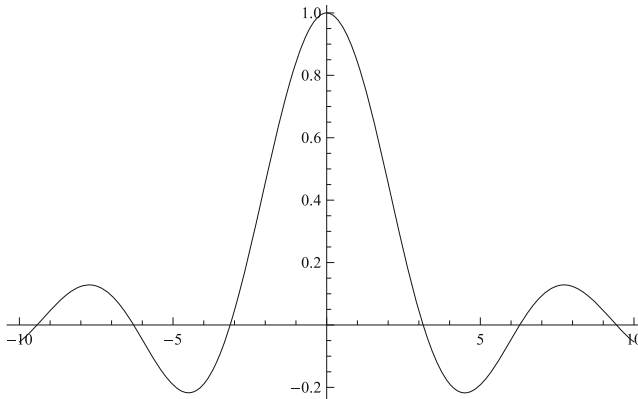
In[62] := Clear[a]

2.7 Plots

A simple plot of a function.

In[63] := Plot[Sin[x]/x, {x, -10, 10}]

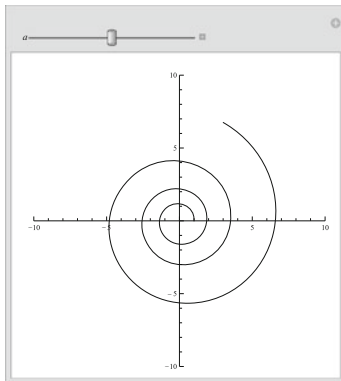
Out[63] =



A curve given parametrically— x and y are functions of t . This particular curve contains a parameter a , which can be adjusted by the mouse. If you click the small plus sign near the marker, a control panel will open. There you can start (and stop) animation.

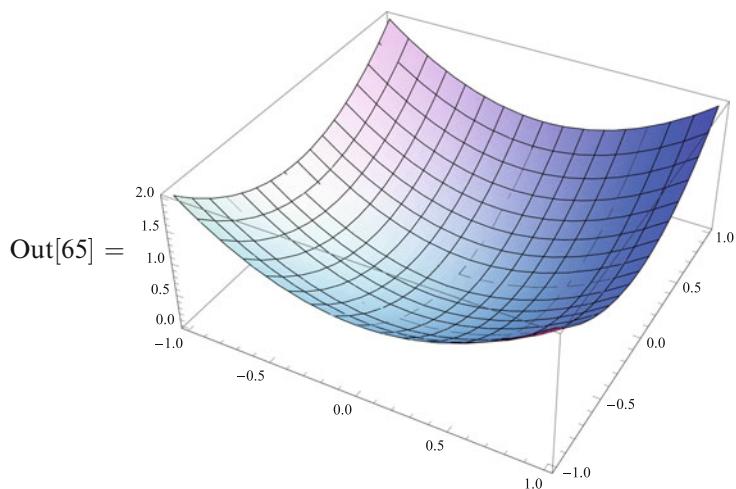
In[64] := Manipulate[ParametricPlot[{Exp[a * t] * Cos[t], Exp[a * t] * Sin[t]}, {t, 0, 20}, PlotRange -> {{-10, 10}, {-10, 10}}, {a, 0.1}, 0, 0.2]]

Out[64] =



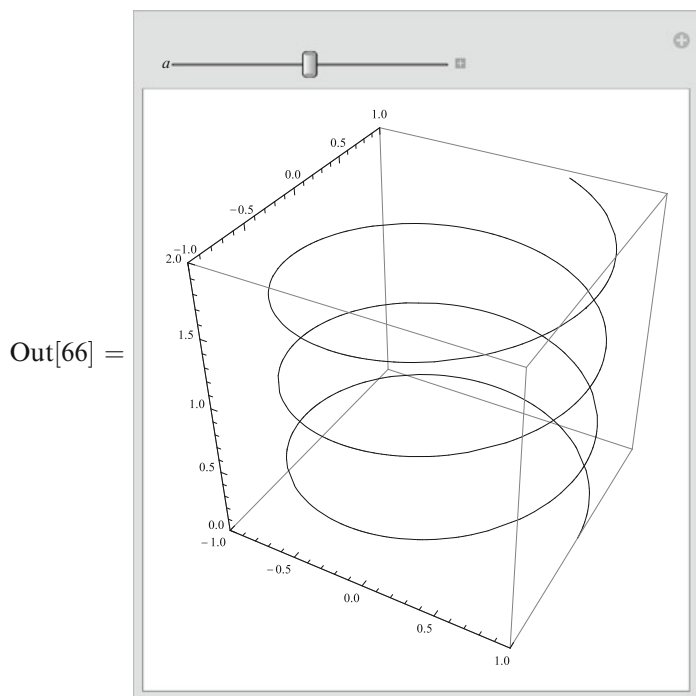
A three-dimensional plot of a function of two variables. It can be rotated by the mouse.

In[65] := Plot3D[x^2 + y^2, {x, -1, 1}, {y, -1, 1}]



A three-dimensional curve given parametrically. The parameter a can be adjusted by the mouse.

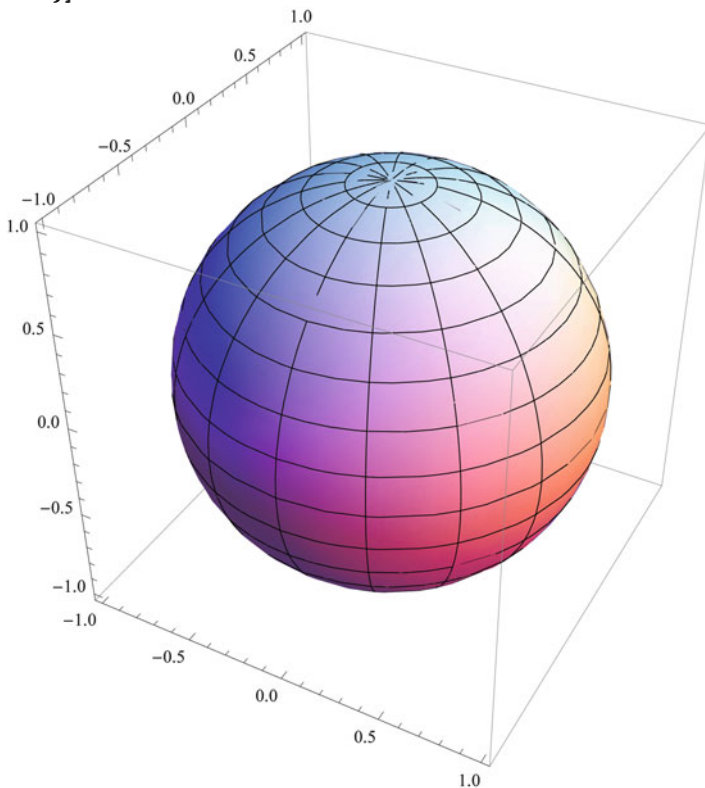
In[66] := Manipulate[ParametricPlot3D[{Cos[t], Sin[t], a*t}, {t, 0, 20}, PlotRange -> {{-1, 1}, {-1, 1}, {0, 2}}], {a, 0.1}, 0, 0.2]



A surface given parametrically.

In[67] := ParametricPlot3D[{Sin[t] * Cos[u], Sin[t] * Sin[u], Cos[t]}, {t, 0, Pi}, {u, 0, 2 * Pi}]

Out[67] =



2.8 Substitutions

Substitutions are a fundamental concept in *Mathematica*, its main working instrument. This substitution replaces $f[x]$ by x^2 .

In[68] := S = f[x] -> x^2

Out[68] = $f[x] \rightarrow x^2$

Let's apply it to the expression $f[x]$.

In[69] := f[x]/.S

Out[69] = x^2

We've got x^2 , as expected. And what if we apply it to $f[y]$?

In[70] := f[y]/.S

Out[70] = $f[y]$

It hasn't triggered. The following substitution replaces the function f with an arbitrary argument by the square of this argument.

In[71] := S = f[x.] -> x^2

Out[71] = f[x.] -> x^2

Let's check.

In[72] := {f[x], f[y], f[2]}/.S

Out[72] = {x^2, y^2, 4}

In[73] := Clear[S]

2.9 Equations

Here is an equation.

In[74] := Eq = a*x + b == 0

Out[74] = b + ax == 0

Let's solve it for x .

In[75] := S = Solve[Eq, x]

Out[75] = $\left\{ \left\{ x \rightarrow -\frac{b}{a} \right\} \right\}$

We've got a list of solutions, in this particular case having a single element. Each solution is a list of substitutions, which replaces our unknowns by the corresponding expressions. And how can we extract the value of x from this result? Let's take the first (and the only) element of the list S .

In[76] := S1 = First[S]

Out[76] = $\left\{ x \rightarrow -\frac{b}{a} \right\}$

And now we apply this list of substitutions (in this particular case, it's single element) to the unknown x .

In[77] := x/.S1

Out[77] = $-\frac{b}{a}$

Here is a more advanced example—a quadratic equation. It has two solutions.

In[78] := S = Solve[a*x^2 + b*x + c == 0, x]

Out[78] = $\left\{ \left\{ x \rightarrow \frac{-b - \sqrt{b^2 - 4ac}}{2a} \right\}, \left\{ x \rightarrow \frac{-b + \sqrt{b^2 - 4ac}}{2a} \right\} \right\}$

How can we extract the value of x in the second solution? Let's apply the second element of the solutions list S (which is a single-element list of substitutions) to the unknown x .

In[79] := x/.S[[2]]

Out[79] = $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$

And here is a system of 2 linear equations.

In[80] := Eq = {a*x + b*y == e, c*x + d*y == f}

Out[80] = {ax + by == e, cx + dy == f}

It has a single solution.

In[81] := S = Solve[Eq, {x,y}]

$$\text{Out[81]} = \left\{ \left\{ x \rightarrow -\frac{de - bf}{bc - ad}, y \rightarrow -\frac{-ce + af}{bc - ad} \right\} \right\}$$

This (first and the only) solution is a list of two substitutions.

In[82] := S1 = S[[1]]

$$\text{Out[82]} = \left\{ x \rightarrow -\frac{de - bf}{bc - ad}, y \rightarrow -\frac{-ce + af}{bc - ad} \right\}$$

How to find the values of x and y in this solution? Apply this list of substitutions to the unknowns x and y .

In[83] := {x/.S1,y/.S1}

$$\text{Out[83]} = \left\{ -\frac{de - bf}{bc - ad}, -\frac{-ce + af}{bc - ad} \right\}$$

In[84] := Clear[Eq,S,S1]

Chapter 3

Expressions

All objects with which *Mathematica* works are expressions. There are two classes of them—atoms and composite expressions.

3.1 Atoms

There are three kinds of atoms—numbers, symbols, and strings.

Numbers

Integer numbers (of unlimited size).

In[1] := 1234567890

Out[1] = 1234567890

A rational number consists of the numerator and the denominator.

In[2] := 1234567890/987654321

Out[2] = $\frac{137174210}{109739369}$

A complex number consists of the real and imaginary parts.

In[3] := 1 + 2 * I

Out[3] = 1 + 2i

Real numbers can have arbitrarily high precision.

In[4] := 1234567890.987654321

Out[4] = $1.23456789098765432 \times 10^9$

Symbols

A variable can be in one of two states. Initially it is free—it means itself (a symbol).

In[5] := x

Out[5] = x

Assigning a value to it, we make it bound.

In[6] := x = 123

Out[6] = 123

Now, when we use it (e.g., just by asking *Mathematica* to print it), its value is substituted.

In[7] := x

Out[7] = 123

How to make it free again?

In[8] := Clear[x]

Let's check.

In[9] := x

Out[9] = x

Strings

In[10] := "This is a string"

Out[10] = This is a string

3.2 Composite Expressions

A composite expression is a function of a number of arguments, each of which is an expression (i.e., an atom or a composite expression).

In[11] := a = f[g[x, 1], h[y, z, 2]]

Out[11] = f[g[x, 1], h[y, z, 2]]

Each composite expression has a head—the function which is applied to arguments.

In[12] := Head[a]

Out[12] = f

The number of arguments is given by the function Length.

In[13] := Length[a]

Out[13] = 2

Arguments are extracted by the function Part.

In[14] := Part[a, 1]

Out[14] = g[x, 1]

In[15] := Part[a, 2]

Out[15] = h[y, z, 2]

And this is the first part of the second part of the expression *a*.

In[16] := Part[a, 2, 1]

Out[16] = y

An alternative syntax.

In[17] := a[[2, 1]]

Out[17] = y

Zeroth part of an expression is its head.

In[18] := Part[a, 0]

Out[18] = f

By the way, a head can be any expression, not just a symbol.

In[19] := b = f[x][y, 1]

Out[19] = f[x][y, 1]

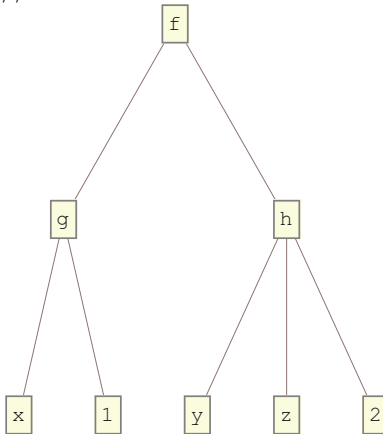
In[20] := Head[b]

Out[20] = f[x]

Expressions are trees whose leaves are atoms.

In[21] := TreeForm[a]

Out[21]//TreeForm =



Parts of an expressions can be changed.

In[22] := a[[1, 2]] = 0; a

Out[22] = f[g[x, 0], h[y, z, 2]]

In[23] := a[[0]] = j; a

Out[23] = j[g[x, 0], h[y, z, 2]]

A group of arguments can be selected, not just a single argument.

In[24] := b = f[x1, x2, x3, x4, x5, x6]

Out[24] = f[x1, x2, x3, x4, x5, x6]

In[25] := Part[b, Span[2, 4]]

Out[25] = f[x2, x3, x4]

An alternative syntax.

In[26] := b[[2;;4]]

Out[26] = f[x2, x3, x4]

From the beginning to 3:

In[27] := b[;,3]

Out[27] = $f[x_1, x_2, x_3]$

From 4 to the end:

In[28] := b[4;]

Out[28] = $f[x_4, x_5, x_6]$

From 1 to 5 by 2:

In[29] := b[1;;5;;2]

Out[29] = $f[x_1, x_3, x_5]$

If such a form is used in the left-hand side of an assignment, each of the selected arguments will be replaced:

In[30] := b[1;;5;;2] = x; b

Out[30] = $f[x, x_2, x, x_4, x, x_6]$

In[31] := Clear[a, b]

3.3 Queries

Let's define an integer number, a rational number, a real (floating point) number, and a complex number.

In[32] := i = -1234567890; r = -1234567890/987654321;

f = -1234567890987654321.1234567890987654321; c = 1 - 2 * I;

The query AtomQ (Q from Query) returns the symbol True if its argument is an atom and False if it is a composite expression.

In[33] := {AtomQ[i], AtomQ[r], AtomQ[c], AtomQ[f[x]]}

Out[33] = {True, True, True, False}

The function Head can be applied even to atoms.

In[34] := {Head[i], Head[r], Head[c], Head[f]}

Out[34] = {Integer, Rational, Complex, Real}

The function FullForm shows the internal form of an expression with which *Mathematica* operates (to some approximation). For example, a rational number has the head Rational and two arguments—its numerator and denominator.

In[35] := FullForm[r]

Out[35] // FullForm =

Rational[-137174210, 109739369]

A complex number has the head Complex and two arguments—its real and imaginary parts.

In[36] := FullForm[c]

Out[36] // FullForm =

Complex[1, -2]

The internal representation of a floating point number is rather complicated. It contains the mantissa and the exponent and also the number of significant (decimal) digits. In this particular case, there are 37 significant digits.

In[37] := FullForm[f]

Out[37]//FullForm =

$-1.234567890987654321123456789098765432137.09151497751671 * ^18$

The query IntegerQ checks if its argument is an integer number.

In[38] := {IntegerQ[i], IntegerQ[r], IntegerQ[c]}

Out[38] = {True, False, False}

The functions Numerator and Denominator extract the parts of a rational number.

In[39] := {Numerator[r], Denominator[r]}

Out[39] = {-137174210, 109739369}

The functions Re and Im extract the real and imaginary parts of a complex number.

In[40] := {Re[c], Im[c]}

Out[40] = {1, -2}

In[41] := Clear[i, r, f, c]

3.4 Forms of an Expression

FullForm is a very useful function. It shows what *Mathematica* really thinks about an expression. Use it often, and you will learn a lot. For example, the following expression is a sum of 4 terms, one of which is the number -1 multiplied by the symbol z .

In[42] := FullForm[x + y - z - 1]

Out[42]//FullForm =

Plus[-1, x, y, Times[-1, z]]

And this one is a product of 4 factors, among which are the rational number $2/3$ and the negative power z^{-1} .

In[43] := a = 2 * x * y / (3 * z)

Out[43] = $\frac{2xy}{3z}$

In[44] := FullForm[a]

Out[44]//FullForm =

Times[Rational[2, 3], x, y, Power[z, -1]]

Nevertheless, the functions Numerator and Denominator work as expected.

In[45] := {Numerator[a], Denominator[a]}

Out[45] = {2xy, 3z}

In[46] := Clear[a]

We have already handled lists many times. A list appears to be just the function List with arguments—elements of the list.

In[47] := FullForm[{x, y, z}]

Out[47]//FullForm =

List[x, y, z]

Any *Mathematica* command can be written as a function with arguments (sometimes, it can also be written in some other way). For example, assignment is the function Set. In order to see this, we'll have to put an assignment inside

the function `Hold`. Otherwise it would be executed immediately, and the function `FullForm` would receive only the result returned by the assignment—the symbol x .

In[48] := FullForm[Hold[a = x]]

Out[48]//FullForm =

Hold[Set[a,x]]

Here is a rational expression.

In[49] := a = Together[x/(x + y) + y/(x - y)]

Out[49] = $\frac{x^2 + y^2}{(x - y)(x + y)}$

Its full form:

In[50] := FullForm[a]

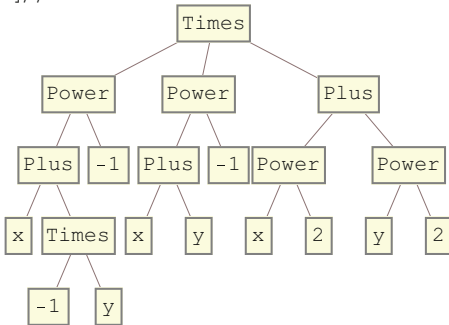
Out[50]//FullForm =

Times[Power[Plus[x, Times[-1, y]], -1], Power[Plus[x, y], -1],
Plus[Power[x, 2], Power[y, 2]]]

And this is the same expression as a tree.

In[51] := TreeForm[a]

Out[51]//TreeForm =



In[52] := Clear[a]

Chapter 4

Patterns and Substitutions

Substitution is the most fundamental operation in *Mathematica*. Its left-hand side is a pattern. In a given expression, all subexpressions matching the pattern are found and replaced by the right-hand side of the substitution.

4.1 Simple Patterns

$f[x]$ with a specific argument x .

In[1] := {f[x], f[y]}/. f[x]->x^2

Out[1] = {x^2, f[y]}

f with an arbitrary argument.

In[2] := {f[x], f[y]}/. f[x_-]->x^2

Out[2] = {x^2, y^2}

f with two identical (arbitrary) arguments.

In[3] := {f[x,x], f[x,y]}/. f[x_.,x_-]->g[x]

Out[3] = {g[x], f[x,y]}

An example of a more complicated pattern.

In[4] := f[g[f[x],y],h[f[x]]]/. f[g[x_.,y],h[x_-]->F[x,y]

Out[4] = F[f[x],y]

f with an argument being an arbitrary integer number.

In[5] := {f[x], f[2]}/. f[x_Integer]->x^2

Out[5] = {f[x], 4}

In fact, such a form of an arbitrary argument checks its head. This substitution applies when the argument's head is g .

In[6] := {f[g[x,y]], f[h[x,y]]}/. f[x_g]->x^2

Out[6] = {g[x,y]^2, f[h[x,y]]}

And this one—when the argument is a sum.

In[7] := {f[{x,y}], f[x+y]}/. f[x_Plus]->x^2

Out[7] = {f[{x,y}], (x+y)^2}

And this one—when the argument is a list. By the way, note what happens when a list is being squared.

In[8] := {f[{x,y}],f[x+y]}/.f[x_List]->x^2

Out[8] = {{x²,y²},f[x+y]}

One more example.

In[9] := a = Sqrt[x]/Sqrt[y]

Out[9] = $\frac{\sqrt{x}}{\sqrt{y}}$

In[10] := a/.{Sqrt[x]->u,Sqrt[y]->v}

Out[10] = $\frac{u}{\sqrt{v}}$

Why hasn't the second substitution triggered?

In[11] := FullForm[a]

Out[11]//FullForm =

Times[Power[x,Rational[1,2]],Power[y,Rational[-1,2]]]

a does not contain $y^{1/2}$, only $y^{-1/2}$; therefore, the substitution $y^{1/2} \rightarrow v$ does not work.

Out[11] = **Clear[a]**

4.2 One-Shot and Repeated Substitutions

Here is an expression.

In[12] := a = x^2 + y^2

Out[12] = $x^2 + y^2$

Let's increase x by 1 in it. This example demonstrates that a substitution is not applied repeatedly. *Mathematica* searches for subexpressions matching the pattern (in this case x) in the expression a . After finding such a subexpression, *Mathematica* replaces it by the right-hand side. The result of such a replacement is not searched again for subexpressions matching the pattern.

In[13] := a = a/.x->x+1

Out[13] = $(1+x)^2 + y^2$

A list of substitutions can be applied to an expression. They are all applied in parallel—*Mathematica* searches for subexpressions matching some pattern from the list and replaces these subexpressions by the corresponding right-hand side. Therefore two symbols can be interchanged in an expression in this simple way.

In[14] := a = a/.{x->y,y->x}

Out[14] = $x^2 + (1+y)^2$

In[15] := Clear[a]

The operator `//.` (in contrast to `/.`) applies a substitution repeatedly, while it is applicable. If several substitutions are applicable to some subexpression, *Mathematica* first applies the most specific one (it is not always easy to determine which substitution is more specific and which is more general; in simple cases, this is clear).

In[16] := fac[10]//. {fac[0]->1, fac[n_-]->n * fac[n - 1]}

Out[16] = 3628800

By the way, what are the real names of /. and // . ?

In[17] := FullForm[Hold[a/.x->y]]

Out[17]//FullForm =

Hold[ReplaceAll[a, Rule[x, y]]]

In[18] := FullForm[Hold[a//.x->y]]

Out[18]//FullForm =

Hold[ReplaceRepeated[a, Rule[x, y]]]

4.3 Products

Let's take a product.

In[19] := FullForm[a = 2 * x * y * z]

Out[19]//FullForm =

Times[2, x, y, z]

The pattern xy is considered contained in this product, though the internal representation of a does not contain $\text{Times}[x, y]$ explicitly.

In[20] := a/.x*y->z

Out[20] = $2z^2$

And this product does not contain xy .

In[21] := FullForm[a = 2 * x^2 * y * z]

Out[21]//FullForm =

Times[2, Power[x, 2], y, z]

In[22] := a/.x*y->z

Out[22] = $2x^2yz$

This product contains powers of x and y .

In[23] := FullForm[a = 2 * x^2 * y^3 * z]

Out[23]//FullForm =

Times[2, Power[x, 2], Power[y, 3], z]

We want to replace each product of powers of x and y by the function f of these powers. Such a problem occurs very often. For example, we want to integrate some class of expressions, and we know the result of integration as a function of powers of some variables (or subexpressions).

In[24] := a/.x^n_*y^m->f[n, m]

Out[24] = $2zf[2, 3]$

This works OK. And here?

In[25] := FullForm[a = 2 * x^2 * y * z]

Out[25]//FullForm =

Times[2, Power[x, 2], y, z]

In[26] := a/.x^n_*y^m->f[n, m]

Out[26] = $2x^2yz$

This doesn't work. The product a does not contain a product of *powers* of x and y : the symbol y is not in the argument of the function `Power`. In the next example the substitution works again—as we have seen, *Mathematica* considers dividing by y as multiplying by y^{-1} .

In[27] := FullForm[a = 2 * x^2 * z/y]

Out[27]//FullForm =

Times[2, Power[x, 2], Power[y, -1], z]

In[28] := a/.x^n_.*y^m_-->f[n,m]

Out[28] = 2zf[2, -1]

Let's return to the previous expression. How can we instruct *Mathematica* to consider y as a particular case of the pattern “ y to an arbitrary power”? This is what an optional arbitrary argument $m_.$ is for. When it is used in an exponent, its default value (which is used when there is no power at all) is 1.

In[29] := FullForm[a = 2 * x^2 * y * z]

Out[29]//FullForm =

Times[2, Power[x, 2], y, z]

In[30] := a/.x^n_.*y^m_-->f[n,m]

Out[30] = 2zf[2, 1]

In[31] := FullForm[a = 2 * x * y * z]

Out[31]//FullForm =

Times[2, x, y, z]

In[32] := a/.x^n_.*y^m_-->f[n,m]

Out[32] = 2zf[1, 1]

So far so good. But what if the symbol y is absent? Will *Mathematica* consider this as a particular case of the pattern “ y to an arbitrary power” with the power equal 0? It will not.

In[33] := FullForm[a = 2 * x^2 * z]

Out[33]//FullForm =

Times[2, Power[x, 2], z]

In[34] := a/.x^n_.*y^m_-->f[n,m]

Out[34] = $2x^2z$

The following method will work always. Let's collect several test expressions to a list.

In[35] := a = {2 * x * y * z, 2 * x^2 * y * z, 2 * x^2 * y^3 * z, 2 * x^2 * z/y, 2 * x^2 * z, 2 * z}

Out[35] = $\left\{ 2xyz, 2x^2yz, 2x^2y^3z, \frac{2x^2z}{y}, 2x^2z, 2z \right\}$

The method is as follows. Multiply our expression by $f[0,0]$, and apply a list of substitutions. If f with some arguments is multiplied by an arbitrary power of x , then the first argument of f is increased by this power. Of course, if x is not raised to any power, we want to use the default power equal to 1. Powers of y are treated in the same way. We need to use the repeated substitution `//.`—after one substitution from the list has been applied (e.g., the one about x), we want the other one to be applied to the result (to take y into account).

```

In[36] := s = {x^l_.*f[n_,m_]->f[n+l,m],y^l_.*f[n_,m_]->f[n,m+l]}
Out[36] = {x^l_.*f[n_,m_] -> f[l+n,m],y^l_.*f[n_,m_] -> f[n,l+m]}
In[37] := a*f[0,0]//.s
Out[37] = {2zf[1,1],2zf[2,1],2zf[2,3],2zf[2,-1],2zf[2,0],2zf[0,0]}
In[38] := Clear[a,s]

```

4.4 Sums

Substitutions for sums are similar to those for products. They are used much more rarely. Don't use them if you can avoid this.

```
In[39] := FullForm[a = x + y + z + 2]
```

```
Out[39]//FullForm =
Plus[2,x,y,z]
```

```
In[40] := a/.x+y->z
```

```
Out[40] = 2 + 2z
```

```
In[41] := FullForm[a = 2 * x + y + z + 2]
```

```
Out[41]//FullForm =
Plus[2, Times[2,x],y,z]
```

```
In[42] := a/.x+y->z
```

```
Out[42] = 2 + 2x + y + z
```

This substitution replaces a sum of x and y with arbitrary coefficients by the function f of these coefficients.

```
In[43] := FullForm[a = 2 * x + 3 * y + z + 2]
```

```
Out[43]//FullForm =
Plus[2, Times[2,x], Times[3,y],z]
```

```
In[44] := a/.n_*x+m_*y->f[n,m]
```

```
Out[44] = 2 + z + f[2,3]
```

```
In[45] := FullForm[a = 2 * x + y + z + 2]
```

```
Out[45]//FullForm =
Plus[2, Times[2,x],y,z]
```

```
In[46] := a/.n_*x+m_*y->f[n,m]
```

```
Out[46] = 2 + 2x + y + z
```

```
In[47] := FullForm[a = 2 * x - y + z + 2]
```

```
Out[47]//FullForm =
Plus[2, Times[2,x], Times[-1,y],z]
```

```
In[48] := a/.n_*x+m_*y->f[n,m]
```

```
Out[48] = 2 + z + f[2,-1]
```

Here again an optional arbitrary argument can be used. When it is used as a factor, a subexpression is considered matching this pattern even if there is no such a factor, and its value in this case is taken to be 1.

In[49] := FullForm[a = 2 * x + y + z + 2]

Out[49]//FullForm =
Plus[2, Times[2, x], y, z]

In[50] := a/.n..*x+m..*y->f[n,m]

Out[50] = 2 + z + f[2, 1]

In[51] := FullForm[a = x + y + z + 2]

Out[51]//FullForm =
Plus[2, x, y, z]

In[52] := a/.n..*x+m..*y->f[n,m]

Out[52] = 2 + z + f[1, 1]

In[53] := FullForm[a = x + z + 2]

Out[53]//FullForm =
Plus[2, x, z]

In[54] := a/.n..*x+m..*y->f[n,m]

Out[54] = 2 + x + z

And here is our method which always works.

In[55] := a = {x + y + z + 2, 2 * x + y + z + 2, 2 * x + 3 * y + z + 2, 2 * x - y + z + 2, x + z + 2, z + 2}

Out[55] = {2 + x + y + z, 2 + 2x + y + z, 2 + 2x + 3y + z, 2 + 2x - y + z, 2 + x + z, 2 + z}

In[56] := s = {l..*x + f[n..,m..]->f[n+l,m], l..*y + f[n..,m..]->f[n,m+l]}

Out[56] = {f[n..,m..] + xl.. -> f[l + n, m], f[n..,m..] + yl.. -> f[n, l + m]}

In[57] := a + f[0,0]//.s

Out[57] = {2 + z + f[1, 1], 2 + z + f[2, 1], 2 + z + f[2, 3], 2 + z + f[2, -1], 2 + z + f[1, 0], 2 + z + f[0, 0]}

In[58] := Clear[a, s]

4.5 Conditions

Substitutions which apply only when an arbitrary variable satisfies some condition are often needed.

In[59] := {f[1.5], f[3/2], f[x/2]}/.f[x_?NumberQ]->x^2

Out[59] = $\left\{ 2.25, \frac{9}{4}, f\left[\frac{x}{2}\right] \right\}$

But this method is not very general. It checks a condition depending on a single variable. The operator /; can be applied to a pattern (or its part). It can be read as “such that.” The condition in it can depend on several arbitrary variables.

In[60] := s = {fac[0]->1, fac[n_Integer;/;n > 0]->n * fac[n - 1]}

Out[60] = {fac[0] -> 1, fac[n_Integer;/;n > 0] -> n fac[-1 + n]}

In[61] := {fac[10], fac[-10]}//.s

Out[61] = {3628800, fac[-10]}

Internally, this operator is the function Condition.

In[62] := FullForm[s]

Out[62]//FullForm =
List[Rule[fac[0], 1],


```
Rule[fac[Condition[Pattern[n, Blank[Integer]], Greater[n, 0]],
      Times[n, fac[Plus[-1, n]]]]]
```

In[63] := Clear[s]

One common case is when you want to replace $f[x]$ by $g[x]$ only for some values of x .

In[64] := f[a] + f[b] + f[c]/.f[x_/:x == a|x == b]->g[x]

Out[64] = $f[c] + g[a] + g[b]$

4.6 Variable Number of Arguments

A pattern can involve a construct which matches not a single subexpression but an arbitrary-length subsequence of arguments of a function. This is very convenient for working with functions having an arbitrary number of arguments. Let's consider an example. The function f has any number of arguments. We want to shuffle them in the opposite order. First, let's put a fence at the end of the argument list.

In[65] := a = f[x, y, z]

Out[65] = $f[x, y, z]$

In[66] := a = a/.f[x_---]->f[x, Fence]

Out[66] = $f[x, y, z, Fence]$

Now we take the arguments from the left one by one and throw them over the fence (placing them immediately after the fence).

In[67] := a = a/./f[x_-, y_---, Fence, z_---]->f[y, Fence, x, z]

Out[67] = $f[Fence, z, y, x]$

Now the fence is at the left, and the arguments are after it in the opposite order. What's left is to remove the fence.

In[68] := a = a/.f[Fence, x_---]->f[x]

Out[68] = $f[z, y, x]$

Of course, this method only works when the symbol Fence is not present among the arguments. Here is the method which always works. Let the part of the arguments which has not yet been processed be in the first list and the processed part—in the second one. We take the arguments one by one from the beginning of the first list and move them to the beginning of the second one.

In[69] := a = a/.f[x_---]->f[{x}, {}]

Out[69] = $f[\{z, y, x\}, \{\}]$

In[70] := a = a/./f[{x_-, y_---}, {z_---}]->f[{y}, {x, z}]

Out[70] = $f[\{\}, \{x, y, z\}]$

In[71] := a = a/./f[\{\}, {x_---}]->f[x]

Out[71] = $f[x, y, z]$

In addition to x_{---} (with three underscores), which means an arbitrary subsequence of arguments of a function (maybe an empty one), there is also x_{--} (with two underscores)—an arbitrary nonempty subsequence of arguments. I find the first construct more useful.

Chapter 5

Functions

5.1 Immediate and Delayed Assignment

This is an ordinary (immediate) assignment. The result of calculation of the right-hand side (in this case, a quadratic polynomial) is assigned to the variable a .

In[1] := a = Expand[(x + 1)^2]

Out[1] = $1 + 2x + x^2$

And this is a delayed assignment. The unevaluated right-hand side (in this case, an expression with the function Expand) is assigned to the variable b .

In[2] := b := Expand[(x + 1)^2]

Note that a delayed assignment returns no value (an ordinary assignment returns the result of calculation of its right-hand side). The difference between a and b can be seen if we assign something to the variable x . In the first case, the value of x is substituted into the quadratic polynomial.

In[3] := x = z + 1; a

Out[3] = $1 + 2(1 + z) + (1 + z)^2$

In the second case, the value of x is substituted into the expression with the function Expand, and then this expression is calculated.

In[4] := b

Out[4] = $4 + 4z + z^2$

In[5] := Clear[a, b, x]

The real name of the operator := is SetDelayed.

In[6] := FullForm[Hold[a := x]]

Out[6] // FullForm =
Hold[SetDelayed[a, x]]

Similarly, in addition to ordinary substitutions $a \rightarrow b$ (where the right-hand side is calculated at the moment the substitution is defined), there are delayed substitutions $a :> b$ (where the substitution keeps the right-hand side unevaluated, and it is calculated each time the substitution is applied).

```
In[7] := f[z + 1]/.f[x_]->Expand[(x + 1)^2]
Out[7] = 1 + 2(1 + z) + (1 + z)^2
In[8] := f[z + 1]/.f[x_] :>Expand[(x + 1)^2]
Out[8] = 4 + 4z + z^2
```

5.2 Functions

Left-hand side of an assignment can be a pattern, not just a variable. In this case, in all subsequent calculations, any subexpression matching the pattern will be replaced by the right-hand side of the assignment. This can be canceled by the command `Clear`. A pattern can contain arbitrary variables. This is how functions are defined. Here is an example—a function f . In all subsequent calculations, all subexpressions of the form $f[x]$ with arbitrary arguments x will be replaced by the right-hand side (in this case, a quadratic polynomial), in which the value of the actual argument is substituted for x .

```
In[9] := f[x_] = Expand[(x + 1)^2]
Out[9] = 1 + 2x + x^2
```

And this is another function. Its body is an unevaluated expression with `Expand`. Expanding brackets will take place each time the function g with some argument is calculated.

```
In[10] := g[x_] := Expand[(x + 1)^2]
```

Note the difference between them.

```
In[11] := {f[z + 1], g[z + 1]}
Out[11] = {1 + 2(1 + z) + (1 + z)^2, 4 + 4z + z^2}
In[12] := Clear[f, g]
```

5.3 Functions Remembering Their Values

Let's consider a useful trick—a function remembering its calculated values. If it is called again with the same argument, it will not perform calculations, but just return the remembered result. For example, take the factorial. We know `fac[0]`.

```
In[13] := fac[0] = 1
```

```
Out[13] = 1
```

And now attention—the main trick. A delayed assignment to the pattern `fac[n_]` (with an arbitrary n). And what do we have in the right-hand side? An immediate assignment to the pattern `fac[n]` (for a specific n , namely, the value of the actual argument with which the function `fac` was called). What happens when we call `fac[10]`? If the function was never calculated with this argument, then this definition for an arbitrary argument will be used. The right-hand side with 10 substituted for n will be calculated, namely, an immediate assignment `fac[10] = ...`. Its right-hand side is calculated (it is the factorial of 10), and it is remembered as the value of `fac[10]`.

The immediate assignment returns the calculated value of its right-hand side, and this value becomes the result of the function call. If we ask *Mathematica* to calculate `fac[10]` again, then this specific definition for `fac[10]` (generated during the first calculation) will be used, and not the general definition for `fac[n_]`.

In[14] := fac[n_] := fac[n] = n * fac[n - 1]

What does *Mathematica* know about the symbol `fac`?

In[15] := ?fac

Global fac

`fac[0] = 1`

`fac[n_] := fac[n] = n fac[n - 1]`

Only two definitions—for the argument 0 and for an arbitrary argument. Now let's calculate the factorial of 10.

In[16] := fac[10]

Out[16] = 3628800

And what does *Mathematica* know about this symbol now?

In[17] := ?fac

Global fac

`fac[0] = 1`

`fac[1] = 1`

`fac[2] = 2`

`fac[3] = 6`

`fac[4] = 24`

`fac[5] = 120`

`fac[6] = 720`

`fac[7] = 5040`

`fac[8] = 40320`

`fac[9] = 362880`

`fac[10] = 3628800`

`fac[n_] := fac[n] = n fac[n - 1]`

In addition to the general definition, we see also specific ones for all integer values of the argument from 0 to 10. If we ask for the value of `fac` for one of these arguments, then the corresponding specific definition will be used, and the calculation will not be performed again.

In[18] := Clear[fac]

5.4 Fibonacci Numbers

This method is useful but not vital for the factorial, because the time of calculation of `fac[n]` grows linearly with n . For Fibonacci numbers the difference is crucial. For a naive definition, the calculation time grows exponentially. This means you will never get a Fibonacci number with a large n . When results are remembered, the calculation time grows linearly—the result for each value of the argument from 2 to n is calculated once.

```

In[19] := fib[0] = fib[1] = 1
Out[19] = 1
In[20] := fib[n_] := fib[n] = fib[n - 1] + fib[n - 2]
In[21] := ?fib
Global`fib
fib[0] = 1
fib[1] = 1
fib[n_] := fib[n] = fib[n - 1] + fib[n - 2]
In[22] := fib[10]
Out[22] = 89
In[23] := ?fib
Global`fib
fib[0] = 1
fib[1] = 1
fib[2] = 2
fib[3] = 3
fib[4] = 5
fib[5] = 8
fib[6] = 13
fib[7] = 21
fib[8] = 34
fib[9] = 55
fib[10] = 89
fib[n_] := fib[n] = fib[n - 1] + fib[n - 2]
In[24] := Clear[fib]

```

5.5 Functions from Expressions

In most cases, a delayed assignment is used when defining a function. But there are situations when an immediate assignment is needed. Here is one of them. Suppose you have derived an expression a containing a symbol x as a result of some calculation.

```

In[25] := a = D[Expand[(x + 1)^3], x]
Out[25] = 3 + 6x + 3x2

```

Now you want to calculate it many times with different values of x . This can be done by substitutions.

```

In[26] := a /. x -> z + 1
Out[26] = 3 + 6(1 + z) + 3(1 + z)2

```

But this is not very convenient. It would be nice to have a function f with the argument x which is given by the calculated expression a . Such a function can be defined by an immediate assignment. The calculated value is substituted for a in the right-hand side.

```

In[27] := f[x_] = a
Out[27] = 3 + 6x + 3x2
In[28] := f[z + 1]
Out[28] = 3 + 6(1 + z) + 3(1 + z)2
In[29] := Clear[a, f]

```

5.6 Antisymmetric Functions

It is often useful to give a partial definition of a function. To this end we write not just a function with all arguments being arbitrary but a more restrictive pattern in the left-hand side of an assignment. Then, if the values of the actual arguments match the pattern, the function is calculated (i.e., replaced by the right-hand side of the assignment). Otherwise the function remains unevaluated. Here is a simple example. Let's define an antisymmetric function of two arguments. If the arguments are equal, it vanishes.

```
In[30] := f[x_, x_] := 0
```

If they are not equal, we have to decide if they need to be interchanged. The expressions $f[x, y]$ and $f[y, x]$ should reduce to either the first form or the second one, for any x and y . It does not matter to which form, as long as the result is always the same. To this end the function `OrderedQ` is useful. Its argument is a list. It returns `True` if the list is ordered, i.e., each element is “greater than or equal to” the previous one in the sense of some internal ordering *Mathematica* uses for expressions. Details of this ordering are not important.

```

In[31] := {OrderedQ[{x, y}], OrderedQ[{y, x}], OrderedQ[{x, x}]}
Out[31] = {True, False, True}

```

Now it is easy to write a substitution which interchanges the arguments if they are not properly ordered.

```
In[32] := f[x_, y_] /; Not[OrderedQ[{x, y}]] := -f[y, x]
```

```
In[33] := {f[a, a], f[a, b], f[b, a]}
```

```
Out[33] = {0, f[a, b], -f[a, b]}
```

```
In[34] := {f[a + b, a - b], f[a - b, a + b]}
```

```
Out[34] = {-f[a - b, a + b], f[a - b, a + b]}
```

```
In[35] := Clear[f]
```

Of course, a symmetric function can be defined similarly. An odd function of a single argument can be defined in the same way.

```
In[36] := f[0] = 0
```

```
Out[36] = 0
```

```
In[37] := f[x_] /; Not[OrderedQ[{ -x, x}]] := -f[-x]
```

```
In[38] := {f[0], f[a], f[-a]}
```

```
Out[38] = {0, f[a], -f[a]}
```

```
In[39] := {f[a - b], f[b - a]}
```

```
Out[39] = {-f[-a + b], f[-a + b]}
```

```
In[40] := Clear[f]
```

Of course, an even function can be defined similarly.

5.7 Functions with Options

You have undoubtedly noted that many *Mathematica* functions (e.g., Plot) have options. They can be specified in any order; each option is given by a substitution with its name in the left-hand side and its value in the right-hand side. If they are not given, their default values are used. Suppose you want your own function f to have options. This can be done in the following way. Let's assign a list of substitutions giving default values of all options to `Options[f]`. Define the function f with some mandatory arguments and an arbitrary sequence of arguments `opts...` (it may be empty). At the point in the function body where you need the value of the option `opt1` use `opt1/.{opts}/.Options[f]`. The operations `/.` are executed left to right. Therefore, if the user has included a substitution `opt1 → ...` among the arguments, the left `/.` will trigger, and the result will be some value which contains no option names; the right `/.` will not change it. If the user has not given such a substitution, the left `/.` will do nothing, and the right one will replace `opt1` by the default value of this option.

```
In[41] := Options[f] = {opt1->1,opt2->2}
```

```
Out[41] = {opt1 → 1,opt2 → 2}
```

```
In[42] := f[x_.,opts_...] := g[x,opt1/.{opts}/.Options[f],  
opt2/.{opts}/.Options[f]]
```

```
In[43] := {f[a],f[a,opt2->0],f[a,opt2->b,opt1->c]}
```

```
Out[43] = {g[a,1,2],g[a,1,0],g[a,c,b]}
```

```
In[44] := Clear[f]
```

In recent versions of *Mathematica* this can also be written as follows:

```
In[45] := f[x_.,OptionsPattern[f]] := g[x,OptionValue[opt1],OptionValue[opt2]]
```

```
In[46] := Options[f] = {opt1->1,opt2->2};
```

```
In[47] := {f[a],f[a,opt2->0],f[a,opt2->b,opt1->c]}
```

```
Out[47] = {g[a,1,2],g[a,1,0],g[a,c,b]}
```

```
In[48] := Clear[f]
```

5.8 Attributes

A function can have attributes which affect simplification of expressions with this function. The attribute `Flat` removes nested function calls (e.g., `Plus` and `Times` have this attribute).

```
In[49] := Attributes[f] = {Flat}
```

```
Out[49] = {Flat}
```

```
In[50] := f[x,f[y,z],u]
```

```
Out[50] = f[x,y,z,u]
```

The attribute `Orderless` means that the function is symmetric in all arguments, and *Mathematica* may interchange them at will (`Plus` and `Times` have also this attribute).

In[51] := Attributes[f] = {Orderless}

Out[51] = {Orderless}

In[52] := {f[x,y,z],f[z,x,y],f[y,z,x]}

Out[52] = {f[x,y,z],f[x,y,z],f[x,y,z]}

The attribute Listable means that if the first argument is a list, then the function is applied to each element, and the list of results is returned (Plus and Times have this attribute, too).

In[53] := Attributes[f] = {Listable}

Out[53] = {Listable}

In[54] := f[{x,y,z}]

Out[54] = {f[x],f[y],f[z]}

In[55] := f[{x,y,z},a]

Out[55] = {f[x,a],f[y,a],f[z,a]}

There exist a few attributes more. Of course, a function can have several attributes at once. The command Clear[f] removes only substitutions for f (with any arguments), but not its attributes. In order to remove attributes too, use ClearAll.

In[56] := ClearAll[f]; Attributes[f]

Out[56] = {}

5.9 Upvalues

Suppose we want to define a function f such that $f[x] * f[y]$ is replaced by $f[x + y]$ for arbitrary x and y . This can be done by the assignment $f[x_] * f[y_] := f[x + y]$. This definition will be associated with the function Times; *Mathematica* will have to check it each time it multiplies something, i.e., very often, and performance will degrade. It is possible to associate this definition with the function f instead. Then it will be used only when processing a product containing at least one function f .

In[57] := f[x_] * f[y_] ^ := f[Expand[x + y]]

In[58] := f[(x + y)^2] * f[(x - y)^2] * f[x^2] * g[y^2]

Out[58] = f[3x² + 2y²] g[y²]

Here the left-hand side is Times[f[x-], f[y-]], and the definition is associated with f .

If we want a definition for Times[f[x-], g[y-]], we can associate it with either f or g .

In[59] := f /: f[x_] * g[y_] := f[Expand[x - y]]

In[60] := f[(x + y)^2] * f[(x - y)^2] * f[x^2] * g[y^2]

Out[60] = f[3x² + y²]

In[61] := ?f

Global f

f[x_]f[y_] ^ := f[Expand[x + y]]

f /: f[x_]g[y_] := f[Expand[x - y]]

When processing an expression $f[g1[...],g2[...],g3[...]]$, *Mathematica* uses definitions associated with f and also definitions associated with $g1$, $g2$, $g3$, and

having the form $f[\dots] := \dots$ (*upvalues* of $g1$, $g2$, $g3$). It does not look deeper, into arguments of $g1$, $g2$, and $g3$ —this would be too inefficient. In addition to the delayed assignments $\wedge :=$ and $f / : lhs := rhs$ there are also immediate assignments $\wedge =$ and $f / : lhs = rhs$.

In[62] := Clear[f]

Chapter 6

Mathematica as a Programming Language

6.1 Compound Expressions

A compound expression consists of several expressions separated by the operator `;`. They are calculated left to right. The value of a compound expression is the value of the last (rightmost) expression. The values of all the other expressions are thrown away; they are calculated only for side effects. The operator `;` has a low priority, so that it is often necessary to put a compound expression inside brackets. The last expression may be empty. Its value (and hence the value of the compound expression) is the symbol `Null` which is not printed. Therefore, if you want to suppress printing of the result of some calculation (e.g., because it is lengthy), put `;` after it.

```
In[1] := fac[0] = 1;
In[2] := fac[n_] := (Print["n=", n]; n * fac[n - 1])
In[3] := fac[4]
n=4
n=3
n=2
n=1
Out[3] = 24
In[4] := Clear[fac]
In[5] := Null
In[6] := FullForm[x;]
Out[6] // FullForm =
Null
In[7] := FullForm[Hold[a; b]]
Out[7] // FullForm =
Hold[CompoundExpression[a, b]]
```

6.2 Conditional Expressions

If

In[8] := del[x_,y_] := If[x == y, 1, 0]

In[9] := del[a, a]

Out[9] = 1

In[10] := del[1, 2]

Out[10] = 0

When *Mathematica* cannot determine if the condition is true, a conditional expression is returned unevaluated. If such a possibility will appear later, an unevaluated If will be simplified.

In[11] := u = del[a, b]

Out[11] = If[a == b, 1, 0]

In[12] := a = b = x; u

Out[12] = 1

And what to do if several actions should be performed in the branches of If? Use compound expressions, of course! The priority of the operator; is higher than that of, (which separates function arguments, in particular, those of If).

In[13] := f[x_] := If[x > 0, Print["x>0"]; 1, Print["x<=0"]; 0]

In[14] := f[1]

x>0

Out[14] = 1

In[15] := Clear[a, b, u, del, f]

Which

This is a choice with many branches. Arguments of the function Which form pairs: a condition and a result. The conditions are evaluated left to right. As soon as a true one is found, the corresponding result is evaluated and returned. Often (but not always) the last condition is True; the corresponding result is returned when none of the previous conditions is satisfied. When *Mathematica* cannot decide if the conditions are true, the function Which returns unevaluated.

In[16] := sign[x_] := Which[x > 0, 1, x < 0, -1]

In[17] := sign[0.1]

Out[17] = 1

In[18] := sign[0]

In[19] := sign[a]

Out[19] = Which[a > 0, 1, a < 0, -1]

In[20] := Clear[sign]

Conditions

What can be used as conditions in If and Which? The operator `==` returns True if its left-hand side and right-hand side are the same expression. Let's stress: mathematically equivalent expressions written in different forms don't qualify. If the left-hand side and the right-hand one are not identical, this operator returns unevaluated. It is used for writing equations, for example, for the function Solve.

In[21] := {a == a, a == b}

Out[21] = {True, a == b}

In contrast to this, the operator `===` returns False if its arguments are not identical (even if they are mathematically equivalent).

In[22] := {a === a, a === b}

Out[22] = {True, False}

Not[a == b] is written as $a != b$, and Not[a === b] as $a != b$.

The function NumberQ checks if its argument is a number (integer, rational, real, complex).

In[23] := {NumberQ[3.14], NumberQ[Pi]}

Out[23] = {True, False}

The function NumericQ returns True also for symbolic mathematical constants.

In[24] := {NumericQ[3.14], NumericQ[Pi]}

Out[24] = {True, True}

The function FreeQ returns True if its first argument contains no subexpressions given by the second argument. It is often used to check if an expression contains a given symbol.

In[25] := {FreeQ[Sin[a + b], a], FreeQ[Sin[b + c], a]}

Out[25] = {False, True}

The function MatchQ checks if the expression—its first argument— matches the pattern, its second argument. When designing a system of substitutions, use this function often in order to check if your ideas about the structure of expressions agree with those of *Mathematica*.

In[26] := MatchQ[x^2, a_^b_]

Out[26] = True

In[27] := MatchQ[1/x, a_^b_]

Out[27] = True

In[28] := MatchQ[x, a_^b_]

Out[28] = False

Switch

The function Switch starts from evaluating its first argument. All the remaining arguments form couples: a pattern and a result. The first argument is matched against the patterns from left to right. As soon as a match is found, the corresponding result is evaluated and returned. Often (but not always) the last pattern is $x_.$ (or just $.$ because we don't need the value of x).

```

In[29] := f[x_] := Switch[x, _Plus, "A sum", _Times, "A product", _,
  "Neither a sum nor a product"]
In[30] := f[a + b]
Out[30] = A sum
In[31] := f[a * b]
Out[31] = A product
In[32] := f[a^b]
Out[32] = Neither a sum nor a product
In[33] := Clear[f]

```

6.3 Loops

Do

This loop is very convenient to those pupils who were ordered by a teacher to write “I shall behave well” 100 times.

```

In[34] := Do[Print["OK"], {4}]
OK
OK
OK
OK

```

In this loop the parameter varies from 1 to an upper limit.

```

In[35] := Do[Print[x^i], {i, 4}]
x
x2
x3
x4

```

And here—from a lower limit to an upper one.

```

In[36] := Do[Print[x^i], {i, 0, 4}]
1
x
x2
x3
x4

```

And now with a given step.

```

In[37] := Do[Print[x^i], {i, 0, 4, 2}]
1
x2
x4

```

This loop takes the elements of a list.

```

In[38] := Do[Print[x^i], {i, {0, 1, 4}}]
1
x
x4

```

While

While the list is not empty, we print and remove its first element.

```
In[39] := l = {a,b,c};
In[40] := While[l!={},Print[First[l]]; l = Rest[l]]
a
b
c
In[41] := l
Out[41] = {}
In[42] := Clear[l]
```

For

This is a C style loop. First the initialization (the first argument) is executed. If the condition (the second argument) is satisfied, then the loop body (the fourth argument) is executed. Then the increment (the third argument) is performed. The condition is checked again, and so on.

```
In[43] := For[i = 0, i < 5, i ++, Print[xi]]
1
x
x2
x3
x4
```

```
In[44] := i
Out[44] = 5
```

A loop running through several parameters (or data structures) in parallel can be easily written.

```
In[45] := For[i = 0; j = 1, i + j < 20, i ++; j* = 2, Print[xi + yj]]
1 + y
x + y2
x2 + y4
x3 + y8
In[46] := Clear[i, j]
```

6.4 Functions

The function `Function` returns an anonymous function. Its first argument is a formal parameter (or a list of formal parameters), and the second one is an expression containing these formal parameters. When the function is called, the actual parameters are substituted for the formal ones in this expression, and the result of its evaluation is returned as the value of the function. A note for experts: the function `Function`

is a λ -expression. Of course, an anonymous function can be assigned to a variable. This is similar to a function defined by $f[x_] := \dots$, but more efficient. The usual method of assigning a function body to a pattern is more general, because it is possible to construct a function which is defined only for arguments which satisfy some condition (such a function returns unevaluated if the conditions are not satisfied).

This is not possible in the case of `Function`.

In[47] := `f = Function[x, x^2]`

Out[47] = `Function[x, x^2]`

An anonymous function can be just applied to some arguments.

In[48] := `Function[{x, y}, x^2 + y^3][a, b]`

Out[48] = `a^2 + b^3`

The function `Map` applies the function given by its first argument to each element of the list given by the second argument and returns the list of results.

In[49] := `Map[f, {a, b, c}]`

Out[49] = `{a^2, b^2, c^2}`

In[50] := `Clear[f]`

An anonymous function can be the first argument of `Map`. Any function with arguments (e.g., `Plus`) can be the second argument, not just a list. The function given by the first argument is applied to each argument of the expression—the second argument. An expression having the same `Head` (as the second argument) and the calculated results is constructed and returned.

In[51] := `Map[Function[x, x^2], a + b + c]`

Out[51] = `a^2 + b^2 + c^2`

The function `Apply[f, l]` applies f to the list of arguments l ; this simply means that the `Head` of l is replaced by f .

In[52] := `Apply[f, {a, b, c}]`

Out[52] = `f[a, b, c]`

In[53] := `Apply[Times, a + b + c]`

Out[53] = `abc`

The first argument of the function `Select` is a list. It returns the list of those elements which satisfy the condition given by the second argument. In order to avoid inventing names for such disposable things, anonymous functions are often used as the second argument.

In[54] := `Select[{1, 5, 3, 6}, Function[x, x > 4]]`

Out[54] = `{5, 6}`

Function Generator

Here's an interesting example. The function `Adder` has a formal parameter n and returns a function which adds n to its argument, that is, `Adder` is a function generator.

In[55] := `Adder = Function[n, Function[x, x + n]]`

Out[55] = `Function[n, Function[x, x + n]]`

Specific functions can be obtained from it. This one, for example, adds 2 to its argument.

```

In[56] := Add2 = Adder[2]
Out[56] = Function[x$, x$ + 2]
In[57] := Map[Add2, {3, x}]
Out[57] = {5, 2 + x}
In[58] := Clear[Add2, Adder]

```

6.5 Local Variables

When writing a function which can be used as a black box by a user, it is crucial to use local variables. Assigning a value to a local variable does not change the global one with the same name (which can store some value precious for the user). To this end the function `Module` is used. Its first argument is a list of local variables.

```

In[59] := x = 1
Out[59] = 1
In[60] := Module[{x}, x = 2; x]
Out[60] = 2
In[61] := x
Out[61] = 1
In[62] := Clear[x]

```

Coding functions like this means inviting big troubles.

```

In[63] := f = Function[{a, b}, x = a; x * b]
Out[63] = Function[{a, b}, x = a; xb]
In[64] := f[c, x]
Out[64] =  $c^2$ 
In[65] := x
Out[65] =  $c$ 
In[66] := Clear[f, x]

```

This is much better.

```

In[67] := f = Function[{a, b}, Module[{x = a}, x * b]]
Out[67] = Function[{a, b}, Module[{x = a}, xb]]
In[68] := f[c, x]
Out[68] =  $cx$ 
In[69] := x
Out[69] =  $x$ 
In[70] := Clear[f]

```

What happens if a local variable escapes from its scope? We can see that *Mathematica* implements local variables in the most trivial way—by renaming.

```

In[71] := Module[{x}, x]
Out[71] = x$103

```

The function `Block` introduces another kind of local variables. As you value your life or your reason keep away from the function `Block`. Especially in those dark hours when the powers of evil are exalted.

Local Constants

Local variables which cannot be changed after initialization can be introduced. This is done by the function `With`. Such local constants can be considered temporary notations introduced to make writing a single expression easier.

In[72] := x = 1

Out[72] = 1

In[73] := With[{x = a + 1}, Print[x^2]]

Out[73] = $(1 + a)^2$

In[74] := x

Out[74] = 1

In[75] := Clear[x]

6.6 Table

The function `Table` constructs a list of values of an expression where a parameter varies in a given way (like in the `Do` loop).

In[76] := Table[0, {4}]

Out[76] = {0, 0, 0, 0}

In[77] := Table[x^i, {i, 4}]

Out[77] = {x, x², x³, x⁴}

In[78] := Table[x^i, {i, 0, 4}]

Out[78] = {1, x, x², x³, x⁴}

In[79] := Table[x^i, {i, 0, 4, 2}]

Out[79] = {1, x², x⁴}

In[80] := Table[x^i, {i, {0, 1, 4}}]

Out[80] = {1, x, x⁴}

Let's turn the list into a product.

In[81] := Table[x + i, {i, 0, 4}]/.List->Times

Out[81] = $x(1 + x)(2 + x)(3 + x)(4 + x)$

6.7 Parallelization

Now most computers have multi-core processors. The function `Parallelize` tries to calculate its argument faster by starting several *Mathematica* kernels and ordering them to calculate parts of the expression and then collecting these parts together.

In[82] := Parallelize[Table[\$KernelID, {n, 0, 7}]]

Out[82] = {4, 4, 3, 3, 2, 2, 1, 1}

(\$KernelID is the number of the kernel in which a particular list element has been evaluated). In addition to `Table`, it can handle `Map[f, {...}]` (or `f[...]` where `f` is a Listable function) and some other cases. Note that the slave *Mathematica*

kernels started by `Parallelize` don't know definitions made in the master process; only built-in functions can be used. If you need a user-defined function f to be available in slave processes, you should explicitly distribute it.

```
In[83] := f[n_] := Integrate[x^n * Sin[x], {x, 0, Pi}]
```

```
In[84] := DistributeDefinitions[f]
```

```
Out[84] = {f}
```

```
In[85] := Parallelize[Table[{f[n], $KernelID}, {n, 0, 7}]]
```

```
Out[85] = {{2, 4}, {π, 4}, {-4 + π^2, 3}, {π(-6 + π^2), 3}, {48 - 12π^2 + π^4, 2},
  {π(120 - 20π^2 + π^4), 2}, {-1440 + 360π^2 - 30π^4 + π^6, 1},
  {π(-5040 + 840π^2 - 42π^4 + π^6), 1}}
```

```
In[86] := Clear[f]
```

6.8 Functions with an Index

Something like an array of functions can be constructed. Let $f[1]$ be the function adding 1 to its argument and $f[2]$ —the function adding 2 to its argument; $f[n]$ is undefined for other values of n .

```
In[87] := f[1] = Function[x, x + 1]
```

```
Out[87] = Function[x, x + 1]
```

```
In[88] := f[2] = Function[x, x + 2]
```

```
Out[88] = Function[x, x + 2]
```

```
In[89] := {f[1][a], f[2][a], f[n][a]}
```

```
Out[89] = {1 + a, 2 + a, f[n][a]}
```

```
In[90] := Clear[f]
```

6.9 Hold and Evaluate

Assignment

`Assignment` does not evaluate its left-hand side. This is natural: the value of the right-hand side is assigned to the variable given by the left-hand side, not to its value.

```
In[91] := a = x
```

```
Out[91] = x
```

```
In[92] := a = y
```

```
Out[92] = y
```

```
In[93] := a
```

```
Out[93] = y
```

```
In[94] := x
```

```
Out[94] = x
```

The attribute `HoldFirst` is responsible for this.

```
In[95] := Attributes[Set]
```

```
Out[95] = {HoldFirst, Protected, SequenceHold}
```

Evaluate

It is possible to assign a value to *the value* of the left-hand side. To this end the function `Evaluate` is used.

```
In[96] := a = x
```

```
Out[96] = x
```

```
In[97] := Evaluate[a] = y
```

```
Out[97] = y
```

```
In[98] := x
```

```
Out[98] = y
```

```
In[99] := a
```

```
Out[99] = y
```

```
In[100] := Clear[x]; a
```

```
Out[100] = x
```

```
In[101] := Clear[a]
```

Delayed Assignment

Delayed assignment does not evaluate also its right-hand side. The attribute `HoldAll` is responsible for this.

```
In[102] := Attributes[SetDelayed]
```

```
Out[102] = {HoldAll, Protected, SequenceHold}
```

You can use these attributes for your functions, too.

```
In[103] := x = 1; y = 2; z = 3;
```

```
In[104] := Attributes[f] = {HoldAll}; f[x,y,z]
```

```
Out[104] = f[x,y,z]
```

```
In[105] := Attributes[f] = {HoldFirst}; f[x,y,z]
```

```
Out[105] = f[x,2,3]
```

```
In[106] := Clear[x,y,z]
```

```
In[107] := ClearAll[f]
```

The First Argument of the Function Plot

The function `Plot` does not evaluate its arguments.

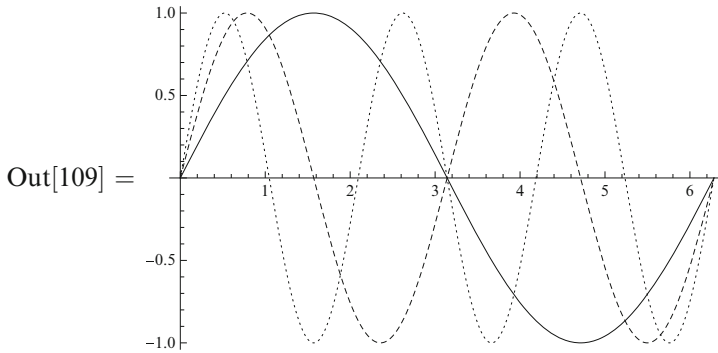
```
In[108] := Attributes[Plot]
```

```
Out[108] = {HoldAll, Protected}
```

The first argument $f[x]$ can be meaningful only for numerical values of x , not for symbolic x . Therefore it is better not to evaluate $f[x]$ before the function `Plot` will

call it for numerical values of x . But if the first argument is a command which generates the list of expressions to draw, it will not work. We want the command to be executed; to this end Evaluate is used.

In[109] := Plot[Evaluate[Table[Sin[n * x], {n, 1, 3}]], {x, 0, 2 * Pi}]



Hold

The function Hold suppresses evaluation of its argument.

In[110] := a = x

Out[110] = x

In[111] := b = Hold[a]

Out[111] = Hold[a]

This suppression can be removed by the function ReleaseHold.

In[112] := ReleaseHold[b]

Out[112] = x

The function Hold is simple—it has the attribute HoldAll, i.e., it does not evaluate its arguments.

In[113] := Attributes[Hold]

Out[113] = {HoldAll, Protected}

In[114] := Clear[a, b]

Chapter 7

Gröbner Bases

7.1 Statement of the Problem

In this lecture we shall consider (in a slightly vulgarized form, without rigorous mathematical terms) an important mathematical achievement of the second half of the last century—Gröbner bases, the Buchberger algorithm (which constructs them), and their applications (see [12, 13] for an introduction).

Suppose we have n variables x_1, \dots, x_n . They are not independent, but satisfy some polynomial equations $p_1 = 0, \dots, p_m = 0$ (p_j are polynomials of x_i). Let's consider some polynomial q of the same variables. It is natural to ask if this polynomial is equal to 0 due to the constraints on our variables or not. If there is another polynomial q_2 , there is the question of their equality.

These questions would become very easy if we had an algorithm reducing polynomials of dependent variables to a canonical form. Two equal polynomials reduce to the same canonical form; a polynomial equal to 0 reduces to the canonical form 0.

We can try to use the equations $p_j = 0$ for simplifying the polynomial q , i.e., for replacing its more complicated terms by combinations of simpler ones. But to do so we first have to accept some convention in which terms are more complicated and which are more simple.

7.2 Monomial Orders

We need a total order of monomials (i.e., products of powers of the variables $x_1^{n_1} \cdots x_n^{n_n}$). An order is total if for any monomials s and t either $s < t$ or $s > t$ or $s = t$ is true. An order is admissible if two properties are satisfied:

- $1 \leq s$ for any monomial s .
- If $s < t$ then $su < tu$ for any monomial u .

Three admissible orders are most popular.

Lexicographic

Anybody who has ever seen a dictionary knows what is lexicographic order. We are comparing two monomials: $s = x_1^{n_1} x_2^{m_2} \cdots x_n^{m_n}$ and $t = x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}$. If the degree of the main variable x_1 in s is larger than in t ($n_1 > m_1$), then $s > t$. If it is smaller ($n_1 < m_1$), then $s < t$. If $n_1 = m_1$, we compare the degrees of the next variable x_2 : if $n_2 > m_2$, then $s > t$; if $n_2 < m_2$, then $s < t$; if $n_2 = m_2$, we compare the degrees of x_3 ; and so on.

By Total Degree than Lexicographic

First we compare the total degree $n = n_1 + n_2 + \cdots + n_n$ of the monomial s and the total degree $m = m_1 + m_2 + \cdots + m_n$ of the monomial t . If $n > m$ then $s > t$; if $n < m$ then $s < t$; if the total degrees are equal, we compare s and t lexicographically.

By Total Degree than Reverse Lexicographic

First we compare the total degrees. If they are equal, then we begin from the junior variable x_n : if its degree in s is larger than in t ($n_n > m_n$), then $s < t$; if it is smaller ($n_n < m_n$), then $s > t$; if $n_n = m_n$, we compare the degrees of the previous variable x_{n-1} ; and so on, that is, this is (within some total degree) the reverse lexicographic order with respect to the reverse list of variables.

7.3 Reduction of Polynomials

Let's fix some admissible monomial order. We'll write polynomials in descending order: the leading term first, followed by the rest ones. We'll normalize all polynomials p_i in such a way that the coefficient of the leading term is 1. Now they can be used as substitutions which replace the leading term by minus sum of the remaining ones, that is, if some term of a polynomial q is divisible by the leading term of some polynomial p_i , we remove this leading term and insert minus sum of the remainder terms of p_i instead. This is called reduction of the polynomial q with respect to the set of polynomials p_i ; if none of the substitutions is applicable, the polynomial q is called reduced. For example, let's consider a set of polynomials

$$\mathbf{In}[1] := \mathbf{p1} = x^2 + y^2 - 1; \mathbf{p2} = x * y - 1/4;$$

Let's try to reduce the polynomial

$$\mathbf{In}[2] := \mathbf{q} = x^2 * y;$$

(we use the lexicographic order with $x > y$). This can be done in different ways.

Let's first reduce q with respect to p_1 :

In[3] := PolynomialReduce[q , { p_1 }, { x, y }]

Out[3] = { { y }, $y - y^3$ }

The result means that if we subtract the polynomial p_1 multiplied by y from q , then the reduced polynomial $y - y^3$ is obtained. This is what we are interested in.

In[4] := $q_1 = \%$ [[2]]

Out[4] = $y - y^3$

Now let's reduce q with respect to p_2 :

In[5] := PolynomialReduce[q , { p_2 }, { x, y }]

Out[5] = { { x }, $\frac{x}{4}$ }

In[6] := $q_2 = \%$ [[2]]

Out[6] = $\frac{x}{4}$

So, we have obtained two different results, q_1 and q_2 . In fact they are equal due to $p_1 = 0$ and $p_2 = 0$, but this is not evident. Every time when more than one substitution can be applied to a term of a polynomial q (in this particular case, we can replace either x^2 or xy in x^2y), a fork appears; maybe, its branches join later, but maybe, they don't (as in this case).

A set of polynomials p_1, \dots, p_n is called a **Gröbner basis** (for a given monomial order) if reduction of any polynomial q with respect to this set is unique.

This definition is not constructive: it does not say how to check if a given set of polynomials forms a Gröbner basis. Presently we shall formulate Buchberger algorithm which transforms a set of polynomials (constraints on variables) into an equivalent system of constraints which is a Gröbner basis.

7.4 S-Polynomials

In our example, the constraints $p_1 = 0$ and $p_2 = 0$ allow us to simplify the monomials x^2 and xy . Do these constraints contain an extra information usable for simplification but not obvious? Yes, they do! Let's multiply p_1 and p_2 by monomials (i.e., products of powers of variables) in such a way that their leading terms become identical (equal to the least common multiple of the leading terms of p_1 and p_2). Then we subtract the second polynomial from the first one. The leading terms cancel, and we get a new polynomial with a new leading term which can be used for simplifying terms in q (because this new polynomial also vanishes). This polynomial is called the S-polynomial $S[p_1, p_2]$ (from the word subtraction). In our example

In[7] := $S = \text{Expand}[y * p_1 - x * p_2]$

Out[7] = $\frac{x}{4} - y + y^3$

This polynomial can be added to the system of constraints $p_1 = 0$, $p_2 = 0$. Let's normalize its leading coefficient to 1:

In[8] := $p_3 = \text{Expand}[4 * S]$

Out[8] = $x - 4y + 4y^3$

In[9] := Clear[S]

Now we have a new possibility for reduction:

In[10] := PolynomialReduce[q2, {p3}, {x, y}]

$$\text{Out[10]} = \left\{ \left\{ \frac{1}{4} \right\}, y - y^3 \right\}$$

Now we've got the same result q_1 . The polynomials $\{p_1, p_2, p_3\}$ form a Gröbner basis. This set can be simplified by reducing them with respect to each other:

In[11] := PolynomialReduce[p1, {p3}, {x, y}]

$$\text{Out[11]} = \left\{ \{x + 4y - 4y^3\}, -1 + 17y^2 - 32y^4 + 16y^6 \right\}$$

In[12] := p1a = Expand[%[[2]]/16]

$$\text{Out[12]} = -\frac{1}{16} + \frac{17y^2}{16} - 2y^4 + y^6$$

In[13] := PolynomialReduce[p2, {p3}, {x, y}]

$$\text{Out[13]} = \left\{ \{y\}, \frac{1}{4}(-1 + 16y^2 - 16y^4) \right\}$$

In[14] := p2a = Expand[-%[[2]]/4]

$$\text{Out[14]} = \frac{1}{16} - y^2 + y^4$$

In[15] := PolynomialReduce[p1a, p2a, {x, y}]

$$\text{Out[15]} = \left\{ \{-1 + y^2\}, 0 \right\}$$

The polynomial p_{1a} reduces to 0, and hence it can be excluded from the system of constraints on our variables x, y . The polynomials p_{2a} and p_3 form a reduced Gröbner basis (with respect to the lexicographic order with $x > y$). Reduced Gröbner basis is unique (for a given monomial order), if we accept the convention that the coefficients of the leading terms are 1.

7.5 Buchberger Algorithm

Generalizing this example, we can formulate an algorithm for construction of the Gröbner basis of a set of n polynomials $P = \{p_i\}$:

1. $S = \{\text{the set of pairs } (p_i, p_j) \text{ of these polynomials with } i < j \leq n\}$
2. **while** S is not empty
3. choose and remove some pair (p_i, p_j) from S ;
4. calculate the S-polynomial $S[p_i, p_j]$;
5. reduce it with respect to P ;
6. if the result is not 0, add this polynomial to P ,
and the corresponding pairs to S .

The set of pairs S alternately shrinks and grows. But it can be proved that this process terminates after a finite number of steps and produces a Gröbner basis P . Reducing these polynomials with respect to each other and throwing zeros away, one can get the reduced Gröbner basis. Some variations can improve the efficiency of the algorithm. For example, when adding a new polynomial to the set P , we can reduce all polynomials already in P with respect to the new one; if some of

them changes, reduce other ones with respect to them, and so on (throwing zeros away while doing so). The order in which pairs are selected from the set S is very important—a good choice can reduce the amount of computations drastically.

Let's ask *Mathematica* to construct the Gröbner basis for the system $\{p_1, p_2\}$ with respect to the lexicographic order with $x > y$:

In[16] := B = GroebnerBasis[{p1, p2}, {x, y}]

Out[16] = $\{1 - 16y^2 + 16y^4, x - 4y + 4y^3\}$

Let's reduce the polynomial q to the canonical form, i.e., reduce it with respect to the Gröbner basis (the result is unique).

In[17] := PolynomialReduce[q, B, {x, y}]

Out[17] = $\left\{ \left\{ -\frac{x}{4}, \frac{1}{4} + xy \right\}, y - y^3 \right\}$

It is difficult to predict the complexity of the Buchberger algorithm. In worst cases it can be very high, i.e., constructing the Gröbner basis of a moderately large system can require a huge amount of calculations. The complexity strongly depends on the monomial order being used. In the case of ordering by the total degree (and then something) reduction tries to lower the total degree of a polynomial. The number of possible terms in a polynomial of a low total degree is small. In the case of the lexicographic order, a polynomial of y of an arbitrarily large degree is considered simpler than x to the first power. Therefore reduction does not lower the number of terms in a polynomial as strongly as in the case of total-degree orders, and the complexity of Gröbner basis calculations is higher. On the other hand, a reduced Gröbner basis with respect to a lexicographic order provides more information useful for solving the system, as we shall see soon. *Mathematica* knows how to construct Gröbner bases with respect to monomial orders we discussed.

In[18] := B = GroebnerBasis[{p1, p2}, {x, y},

MonomialOrder → DegreeLexicographic]

Out[18] = $\{-1 + 4xy, -1 + x^2 + y^2, x - 4y + 4y^3\}$

In[19] := PolynomialReduce[q, B, MonomialOrder → DegreeLexicographic]

Out[19] = $\left\{ \left\{ \frac{x}{4}, 0, 0 \right\}, \frac{x}{4} \right\}$

In[20] := Clear[p1, p2, p3, p1a, p2a, q, q1, q2, B]

7.6 Is the System Compatible?

Consider the system

In[21] := p1 = x^2 * y + 4 * y^2 - 17; p2 = 2 * x * y - 3 * y^3 + 8;

p3 = x * y^2 - 5 * x * y + 1;

Let's construct its Gröbner basis—an equivalent system of equations.

In[22] := GroebnerBasis[{p1, p2, p3}, {x, y, z}]

Out[22] = $\{1\}$

This system contains the equation $1 = 0$. This means that it has no solutions. If the Gröbner basis contains 1, the system is incompatible. The inverse statement can be

also proved—the Gröbner basis of an incompatible system always contains 1 (if we normalize all leading coefficients to 1; otherwise, just some nonzero constant).

In[23] := Clear[p1, p2, p3]

7.7 Gröbner Bases with Respect to Lexicographic Order

Reduction with respect to the lexicographic order first of all tries to lower the degree of the main variable (x in our examples), and if possible, down to 0. Therefore usually there is a subset of polynomials in a reduced Gröbner bases which don't contain x . When x is absent, reduction tries to lower the degree of y , and if possible, down to 0. Therefore usually among these polynomials there are those which don't contain y , and so on. In other words, a lexicographic Gröbner bases has a triangular structure. For example,

In[24] := B = GroebnerBasis[{x^2 + y^2 + z^2, x + y - z, y + z^2}, {x, y, z}]

Out[24] = {z^2 + z^3 + z^4, y + z^2, x - z - z^2}

The polynomial

In[25] := p1 = B[[1]]

Out[25] = z^2 + z^3 + z^4

depends only on the most junior variable z . This means that projections of all solutions of our system on the z axis form a finite set of points—roots of this equation. In our example, they are $z = 0$ and

In[26] := p1 = Expand[p1/z^2]; s = Solve[p1 == 0, z]

Out[26] = {{z -> -(-1)^(1/3)}, {z -> (-1)^(2/3)}}

In[27] := z1 = ComplexExpand[z/.s[[1]]]

Out[27] = -1/2 - i*sqrt(3)/2

In[28] := z2 = ComplexExpand[z/.s[[2]]]

Out[28] = -1/2 + i*sqrt(3)/2

Substituting any of these z values to

In[29] := p2 = B[[2]]

Out[29] = y + z^2

we find the corresponding y value. Substituting these z and y into

In[30] := p3 = B[[3]]

Out[30] = x - z - z^2

we find the corresponding x value. Thus solving any system of polynomial equations with several unknowns reduces to solving single-variable polynomial equations sequentially, thanks to lexicographic Gröbner bases. Even when some of them cannot be solved in radicals, it is easy to solve them numerically to any desired precision.

In[31] := Clear[B, p1, p2, p3, z1, z2]

And here is another example.

In[32] := B = GroebnerBasis[{x^2 - 2*x*y + 2*y^2 - 1, x*y - y*z + z^2 - 1, x*z + y^2 - y*z - 1}, {x, y, z}]

Out[32] = $\{1 - y^2 - 2z^2 + y^2z^2 + z^4, -y + y^3 + z - y^2z + yz^2 - z^3, x - y - 2z + y^2z + z^3\}$

Now we have no equations with a single variable z ; there are 2 equations containing z and y :

In[33] := p1 = Factor[B[[1]]]

Out[33] = $(-1 + z)(1 + z)(-1 + y^2 + z^2)$

In[34] := p2 = Factor[B[[2]]]

Out[34] = $(y - z)(-1 + y^2 + z^2)$

The common set of their solutions is the circle $y^2 + z^2 = 1$. Substituting a point on this circle into

In[35] := p3 = B[[3]]

Out[35] = $x - y - 2z + y^2z + z^3$

we find the corresponding x value, that is, the set of solutions of this system is one-dimensional.

In[36] := Clear[B, p1, p2, p3]

For solving a system of polynomial equations it is useful to construct its Gröbner basis and then to factorize its elements.

7.8 Is the Number of Solutions Finite?

Gröbner bases with respect to other monomial orders don't have such simple triangular structure. But any Gröbner basis can tell us not only if the system is compatible but also if the number of its solutions is finite. Let's consider the same examples.

**In[37] := GroebnerBasis[{x^2 + y^2 + z^2, x + y - z, y + z^2}, {x, y, z},
MonomialOrder → DegreeLexicographic]**

Out[37] = $\{x + y - z, y + z^2, -y + y^2 - yz\}$

The leading terms of the polynomials forming this basis are x , z^2 , and y^2 . What is the dimensionality of the space of polynomials which cannot be reduced with respect to this basis? Only monomials which are not divisible by these leading terms cannot be reduced, namely, 1 , y , and z . So the space of polynomials reduced to the canonical form is three-dimensional for our system of constraints on the variables. Therefore our system has 3 solutions (there explicit form can be obtained more easily from the lexicographic Gröbner basis, as we have seen).

If each variable raised to some power is the leading term of some element of a Gröbner basis, then any monomials with this (or higher) degree of this variable are reducible. Irreducible monomials are inside the parallelepiped bounded by these powers, and their number is finite. Therefore the space of polynomials reduced to the canonical form is finite-dimensional, and the system has a finite number of solutions.

And here is our second example:

```
In[38] := GroebnerBasis[{x^2 - 2*x*y + 2*y^2 - 1, x*y - y*z + z^2 - 1,
  x*z + y^2 - y*z - 1}, {x, y, z}, MonomialOrder -> DegreeLexicographic]
Out[38] = {-1 + y^2 + xz - yz, -1 + xy - yz + z^2, -3 + x^2 + 2y^2 - 2yz + 2z^2,
  x - y - 2z + y^2z + z^3, x - 2y + y^3 - z + yz^2}
```

The leading terms are xz , xy , x^2 , y^2z , and y^3 . Among them there are powers of x and of y , but not of z . Therefore the space of polynomials in the canonical form (i.e., reduced with respect to this basis) is infinite-dimensional. This space contains, e.g., the directions $1, z, z^2, z^3, \dots$ (and not only them). This means that the set of solutions of our system is infinite.

So, the criterion works in the opposite direction, too. If there exists a variable no power of which appears as the leading term of some element of the Gröbner basis (not being multiplied by some other variable), then all powers of this variable are irreducible, and the space of polynomials in the canonical form is infinite-dimensional. And hence the set of solutions of the equation system is infinite.

Knowing the reduced Gröbner basis (for any monomial order) one can also find the dimensionality of the set of solutions [14]. Consider sets of variables satisfying the following condition: none of the leading terms of the elements of the basis is a product of powers of these variables. The number of variables in the longest set gives the dimensionality of the set of solutions. In our example there is just one such set— $\{z\}$. Therefore the set of solutions of this system is one-dimensional.

Chapter 8

Calculus

8.1 Series

Let's expand a function in x at the point $x = 0$ up to the fifth order.

In[1] := s = Series[Exp[x], {x, 0, 5}]

$$\text{Out[1]} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + O[x]^6$$

How are series represented in *Mathematica*? By the function SeriesData. Its first argument is the expansion variable; the second one—the expansion point; the third one—the list of coefficients; the fourth one—the minimum degree (here 0); the fifth one—the power of $O[x]$; the sixth one is 1 for series with integer degrees (all degrees are divided by it if it's not 1). Thus a series is not a sum (Plus) in spite of its appearance.

In[2] := FullForm[s]

Out[2]//FullForm =

SeriesData[x, 0, List[1, 1, Rational[1, 2], Rational[1, 6], Rational[1, 24],
Rational[1, 120]], 0, 6, 1]

Coefficients are extracted by the function SeriesCoefficient.

In[3] := Do[Print[SeriesCoefficient[s, n]], {n, 0, 5}]

1
1
 $\frac{1}{2}$
 $\frac{1}{6}$
 $\frac{1}{24}$
 $\frac{1}{120}$

This series begins with degree -1 .

In[4] := s = Series[Cot[x], {x, 0, 5}]

$$\text{Out[4]} = \frac{1}{x} - \frac{x}{3} - \frac{x^3}{45} - \frac{2x^5}{945} + O[x]^6$$

In[5] := FullForm[s]

Out[5]//FullForm =

SeriesData[x, 0, List[1, 0, Rational[-1, 3], 0, Rational[-1, 45], 0,
Rational[-2, 945]], -1, 6, 1]

This is a series with half-integer degrees.

In[6] := s = Series[Sqrt[x*(1-x)], {x, 0, 5}]

Out[6] = $\sqrt{x} - \frac{x^{3/2}}{2} - \frac{x^{5/2}}{8} - \frac{x^{7/2}}{16} - \frac{5x^{9/2}}{128} + O[x]^{11/2}$

In[7] := FullForm[s]

Out[7]//FullForm =

SeriesData[x, 0, List[1, 0, Rational[-1, 2], 0, Rational[-1, 8], 0,
Rational[-1, 16], 0, Rational[-5, 128]], 1, 11, 2]

This is an expansion at infinity.

In[8] := s = Series[Log[x + 1], {x, Infinity, 4}]

Out[8] = $-\text{Log}\left[\frac{1}{x}\right] + \frac{1}{x} - \frac{1}{2x^2} + \frac{1}{3x^3} - \frac{1}{4x^4} + O\left[\frac{1}{x}\right]^5$

In[9] := FullForm[s]

Out[9]//FullForm =

SeriesData[x, DirectedInfinity[1], List[Times[-1, Log[Power[x, -1]]], 1,
Rational[-1, 2], Rational[1, 3], Rational[-1, 4]], 0, 5, 1]

Coefficients of a series in x may depend on x , but only weakly, weaker than any degree.

In[10] := s = Series[x^x, {x, 0, 3}]

Out[10] = $1 + \text{Log}[x]x + \frac{1}{2}\text{Log}[x]^2x^2 + \frac{1}{6}\text{Log}[x]^3x^3 + O[x]^4$

In[11] := FullForm[s]

Out[11]//FullForm =

SeriesData[x, 0, List[1, Log[x], Times[Rational[1, 2], Power[Log[x], 2]],
Times[Rational[1, 6], Power[Log[x], 3]]], 0, 4, 1]

In[12] := Clear[s]

Operations with Series

Let's take three series.

In[13] := sinx = Series[Sin[x], {x, 0, 7}]

Out[13] = $x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + O[x]^8$

In[14] := cosx = Series[Cos[x], {x, 0, 7}]

Out[14] = $1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + O[x]^8$

In[15] := tanx = Series[Tan[x], {x, 0, 7}]

Out[15] = $x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315} + O[x]^8$

Series can be added, multiplied, divided, etc.

In[16] := tanx * cosx

$$\text{Out[16]} = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + O[x]^8$$

In[17] := sinx/cosx

$$\text{Out[17]} = x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315} + O[x]^8$$

In[18] := sinx^2 + cosx^2

$$\text{Out[18]} = 1 + O[x]^8$$

If a series occurs as an argument of a function, the function is expanded automatically.

In[19] := Exp[sinx]

$$\text{Out[19]} = 1 + x + \frac{x^2}{2} - \frac{x^4}{8} - \frac{x^5}{15} - \frac{x^6}{240} + \frac{x^7}{90} + O[x]^8$$

In[20] := (1 - cosx)/x^2

$$\text{Out[20]} = \frac{1}{2} - \frac{x^2}{24} + \frac{x^4}{720} + O[x]^6$$

Here is an interesting method to expand a function in x .

In[21] := X = Series[x, {x, 0, 7}]

$$\text{Out[21]} = x + O[x]^8$$

In[22] := Sin[X]

$$\text{Out[22]} = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + O[x]^8$$

In[23] := Clear[X]

Series can be differentiated and integrated.

In[24] := D[cosx, x]

$$\text{Out[24]} = -x + \frac{x^3}{6} - \frac{x^5}{120} + O[x]^7$$

In[25] := Integrate[tanx, x]

$$\text{Out[25]} = \frac{x^2}{2} + \frac{x^4}{12} + \frac{x^6}{45} + \frac{17x^8}{2520} + O[x]^9$$

A series (beginning from a small term) can be substituted for the expansion variable of another series. This is $\text{Sin}[\text{Tan}[x]]$.

In[26] := st = sinx/.x->tanx

$$\text{Out[26]} = x + \frac{x^3}{6} - \frac{x^5}{40} - \frac{55x^7}{1008} + O[x]^8$$

An alternative syntax.

In[27] := ComposeSeries[sinx, tanx]

$$\text{Out[27]} = x + \frac{x^3}{6} - \frac{x^5}{40} - \frac{55x^7}{1008} + O[x]^8$$

Let's subtract $\text{Tan}[\text{Sin}[x]]$; this expression is expanded automatically, i.e., series are contagious.

In[28] := st - Tan[Sin[x]]

$$\text{Out[28]} = -\frac{x^7}{30} + O[x]^8$$

In[29] := Clear[st]

Series inversion—solving the equation $\tan x = y$ for x as a series in y .

In[30] := atany = InverseSeries[tanx, y]

$$\text{Out[30]} = y - \frac{y^3}{3} + \frac{y^5}{5} - \frac{y^7}{7} + O[y]^8$$

The result should be the arctangent.

In[31] := Series[ArcTan[y], {y, 0, 7}]

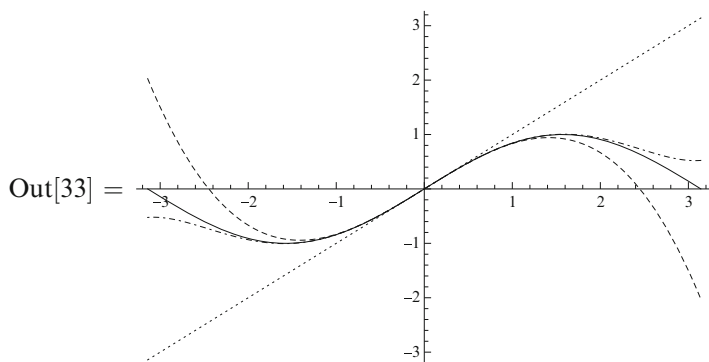
$$\text{Out[31]} = y - \frac{y^3}{3} + \frac{y^5}{5} - \frac{y^7}{7} + O[y]^8$$

In[32] := ComposeSeries[tanx, atany]

$$\text{Out[32]} = y + O[y]^8$$

It is not allowed to substitute a numerical value for the expansion variable into a series. The function `Normal` converts a series into a normal expression by dropping $+O[x]^n$. Here is a plot of sine and a few truncations of its series.

In[33] := Plot[Evaluate[Prepend[Table[Normal[Series[Sin[x], {x, 0, n}]], {n, 1, 5, 2}], Sin[x]], {x, -Pi, Pi}]



In[34] := Clear[sinx, cosx, tanx, atany]

You should work with series as long as possible, converting them into normal polynomials only at the very end. Then terms of too high orders of smallness are dropped automatically. At any moment you know exactly what is the order of the neglected term $O[x]^n$.

Arbitrary-Order Term

The function `SeriesCoefficient` can also be used in this way.

In[35] := SeriesCoefficient[Exp[x], {x, 0, 4}]

$$\text{Out[35]} = \frac{1}{24}$$

This is the 4th coefficient in the expansion of $\text{Exp}[x]$ in x at 0. The number of the series term can be given symbolically.

In[36] := cn = SeriesCoefficient[Exp[x], {x, 0, n}]

$$\text{Out[36]} = \begin{cases} \frac{1}{n!} & n \geq 0 \\ 0 & \text{True} \end{cases}$$

In[37] := Sum[cn * x^n, {n, 0, Infinity}]

$$\text{Out[37]} = e^x$$

In[38] := cn = SeriesCoefficient[Cos[x], {x, 0, n}]

$$\text{Out[38]} = \begin{cases} \frac{i^{n(1+(-1)^n)}}{2n!} & n \geq 0 \\ 0 & \text{True} \end{cases}$$

In[39] := Sum[cn * x^n, {n, 0, Infinity}]

$$\text{Out[39]} = \text{Cos}[x]$$

In[40] := Clear[cn]

8.2 Differentiation

In[41] := f = x * Sin[x + y]

$$\text{Out[41]} = x \text{Sin}[x + y]$$

The derivative in x ; in y ; in x and y ; the second derivative in x ; the second derivative in x and the first one in y :

In[42] := {D[f, x], D[f, y], D[f, x, y], D[f, {x, 2}], D[f, {x, 2}, y]}

$$\text{Out[42]} = \{x \text{Cos}[x + y] + \text{Sin}[x + y], x \text{Cos}[x + y], \text{Cos}[x + y] - x \text{Sin}[x + y], 2 \text{Cos}[x + y] - x \text{Sin}[x + y], -x \text{Cos}[x + y] - 2 \text{Sin}[x + y]\}$$

In[43] := Clear[f]

Unknown Functions

Expressions with unknown functions can be differentiated.

In[44] := D[x * f[x^2], x]

$$\text{Out[44]} = f[x^2] + 2x^2 f'[x^2]$$

Mathematica represents the first derivative of an unknown function f as the operator `Derivative[1]` applied to f .

In[45] := FullForm[%]

$$\text{Out[45]} // \text{FullForm} =$$

$$\text{Plus}[f[\text{Power}[x, 2]], \text{Times}[2, \text{Power}[x, 2], \text{Derivative}[1][f][\text{Power}[x, 2]]]]$$

And this is the second derivative.

In[46] := Expand[D[x * f[x^2], {x, 2}]]

$$\text{Out[46]} = 6x f'[x^2] + 4x^3 f''[x^2]$$

In[47] := FullForm[%]

$$\text{Out[47]} // \text{FullForm} =$$

$$\text{Plus}[\text{Times}[6, x, \text{Derivative}[1][f][\text{Power}[x, 2]]], \text{Times}[4, \text{Power}[x, 3], \text{Derivative}[2][f][\text{Power}[x, 2]]]]$$

Derivative[2, 3] means the second derivative in the first argument and the third one in the second.

In[48] := D[f[x, y], {x, 2}, {y, 3}]

Out[48] = $f^{(2,3)}[x, y]$

In[49] := FullForm[%]

Out[49] // FullForm =
Derivative[2, 3][f][x, y]

Defining Derivatives

Let's tell *Mathematica* that the derivative of the function f is g .

In[50] := f'[x_] := g[x]

In[51] := D[x * f[x^2], x]

Out[51] = $f[x^2] + 2x^2 g[x^2]$

The second derivative is not substituted automatically.

In[52] := Expand[D[x * f[x^2], {x, 2}]]

Out[52] = $6xg[x^2] + 4x^3 f''[x^2]$

we can tell *Mathematica* that $\frac{\partial^3 f[x, y]}{\partial x \partial^2 y}$ is a function g .

In[53] := Derivative[1, 2][f][x_, y_] := g[x, y]

In[54] := D[x * f[x, y], {x, 2}, {y, 2}]

Out[54] = $2g[x, y] + x f^{(2,2)}[x, y]$

8.3 Integration

Indefinite Integrals

In[55] := Integrate[1/(x*(x^2-2)^2), x]

Out[55] = $-\frac{1}{4(-2+x^2)} + \frac{\text{Log}[x]}{4} - \frac{1}{8}\text{Log}[2-x^2]$

In[56] := Integrate[1/(Exp[x]+1), x]

Out[56] = $x - \text{Log}[1 + e^x]$

In[57] := Integrate[x/(Exp[x]+1), x]

Out[57] = $\frac{x^2}{2} - x \text{Log}[1 + e^x] - \text{PolyLog}[2, -e^x]$

In[58] := Integrate[Log[x], x]

Out[58] = $-x + x \text{Log}[x]$

In[59] := Integrate[1/Log[x], x]

Out[59] = $\text{LogIntegral}[x]$

In[60] := Integrate[Exp[x^2],x]

$$\text{Out[60]} = \frac{1}{2} \sqrt{\pi} \text{Erfi}[x]$$

In[61] := Integrate[x * Exp[x^2],x]

$$\text{Out[61]} = \frac{e^{x^2}}{2}$$

In[62] := Integrate[1/Sqrt[(1-x^2)*(1-k^2*x^2)],x]

$$\text{Out[62]} = \frac{\sqrt{1-x^2} \sqrt{1-k^2x^2} \text{EllipticF}[\text{ArcSin}[x],k^2]}{\sqrt{(-1+x^2)(-1+k^2x^2)}}$$

In[63] := Simplify[Integrate[x/Sqrt[(1-x^2)*(1-k^2*x^2)],x],x > 1]

$$\text{Out[63]} = \frac{\text{Log}\left[k\left(k\sqrt{-1+x^2} + \sqrt{-1+k^2x^2}\right)\right]}{k}$$

Definite Integrals

Here *Mathematica* produces a result with some assumptions about the parameter n .

In[64] := Integrate[x^n, {x, 0, 1}]

$$\text{Out[64]} = \text{ConditionalExpression}\left[\frac{1}{1+n}, \text{Re}[n] > -1\right]$$

Let's tell it that $n > -1$.

In[65] := Integrate[x^n, {x, 0, 1}, Assumptions -> {n > -1}]

$$\text{Out[65]} = \frac{1}{1+n}$$

In[66] := Integrate[Exp[a * Sin[x]], {x, 0, 2 * Pi}]

$$\text{Out[66]} = 2\pi \text{BesselI}[0, a]$$

In[67] := Integrate[Log[x]/(1-x), {x, 0, 1}]

$$\text{Out[67]} = -\frac{\pi^2}{6}$$

The default value of the option `Assumptions` for `Simplify`, `Integrate`, etc. can be given in the variable `$Assumptions`.

In[68] := \$Assumptions = {t > 0, t < 1, a > -1, b > -1};

In[69] := Integrate[x^a * (1-x)^b * (1-t*x)^c, {x, 0, 1}]

$$\text{Out[69]} = -(\pi \text{Csc}[a\pi] \text{Gamma}[1+b] \text{Hypergeometric2F1Regularized}[1+a, -c, 2+a+b, t]) / \text{Gamma}[-a]$$

Now we can clear `$Assumptions`.

In[70] := \$Assumptions = True;

Multiple integral

In[71] := Integrate[1/(1+x*y), {x, 0, 1}, {y, 0, 1}]

$$\text{Out[71]} = \frac{\pi^2}{12}$$

8.4 Summation

Finite Sums

$$\text{In}[72] := \text{Sum}[n, \{n, 0, k\}]$$

$$\text{Out}[72] = \frac{1}{2}k(1+k)$$

$$\text{In}[73] := \text{Sum}[n^2, \{n, 0, k\}]$$

$$\text{Out}[73] = \frac{1}{6}k(1+k)(1+2k)$$

$$\text{In}[74] := \text{Sum}[x^n, \{n, 0, k\}]$$

$$\text{Out}[74] = \frac{-1+x^{1+k}}{-1+x}$$

$$\text{In}[75] := \text{Sum}[\text{Binomial}[k, n], \{n, 0, k\}]$$

$$\text{Out}[75] = 2^k$$

$$\text{In}[76] := \text{Sum}[(-1)^n * \text{Binomial}[k, n], \{n, 0, k\}]$$

$$\text{Out}[76] = \text{KroneckerDelta}[k]$$

$$\text{In}[77] := \text{Sum}[\text{Binomial}[k, n]^2, \{n, 0, k\}]$$

$$\text{Out}[77] = \text{Binomial}[2k, k]$$

Series

$$\text{In}[78] := \text{Sum}[1/n^2, \{n, 1, \text{Infinity}\}]$$

$$\text{Out}[78] = \frac{\pi^2}{6}$$

$$\text{In}[79] := \text{Sum}[1/n^4, \{n, 1, \text{Infinity}\}]$$

$$\text{Out}[79] = \frac{\pi^4}{90}$$

$$\text{In}[80] := \text{Sum}[(-1)^n/n^2, \{n, 1, \text{Infinity}\}]$$

$$\text{Out}[80] = -\frac{\pi^2}{12}$$

$$\text{In}[81] := \text{Sum}[x^n/n!, \{n, 0, \text{Infinity}\}]$$

$$\text{Out}[81] = e^x$$

8.5 Differential Equations

A first-order differential equation.

$$\text{In}[82] := \text{DSolve}[D[x[t], t] + x[t] == 0, x[t], t]$$

$$\text{Out}[82] = \left\{ \left\{ x[t] \rightarrow e^{-t} C[1] \right\} \right\}$$

The solution contains an arbitrary constant $C[1]$. Let's add an initial condition:

In[83] := DSolve[{D[x[t], t] + x[t] == 0, x[0] == 1}, x[t], t]

Out[83] = {{x[t] → e^{-t}}}

A second-order differential equation.

In[84] := DSolve[D[x[t], {t, 2}] + x[t] == 0, x[t], t]

Out[84] = {{x[t] → C[1] Cos[t] + C[2] Sin[t]}}

Initial conditions.

In[85] := DSolve[{D[x[t], {t, 2}] + x[t] == 0, x[0] == 0, x'[0] == 1}, x[t], t]

Out[85] = {{x[t] → Sin[t]}}

Boundary conditions.

In[86] := DSolve[{D[x[t], {t, 2}] + x[t] == 0, x[0] == 0, x[1] == 1}, x[t], t]

Out[86] = {{x[t] → Csc[1] Sin[t]}}

A system of differential equations.

In[87] := DSolve[{D[x[t], t] == p[t], D[p[t], t] == -x[t]}, {x[t], p[t]}, t]

Out[87] = {{p[t] → C[1] Cos[t] - C[2] Sin[t], x[t] → C[2] Cos[t] + C[1] Sin[t]}}

Chapter 9

Numerical Calculations

9.1 Approximate Numbers in *Mathematica*

Mathematica usually works with exact numbers, either symbolic (π , e) or rational, and derives exact analytical results. However, it can also perform approximate numerical calculations. Many problems cannot be solved analytically, but numerical solution is possible. On the other side, an analytical result can depend on symbolic parameters; in order to do a numerical calculation, you have to substitute some numerical values for all parameters. It is often useful to check the correctness of a complicated analytical derivation by a direct numerical calculation for a few sets of values of the parameters.

There are two kinds of approximate real numbers in *Mathematica*. The first one is *machine numbers*.

In[1] := $p = N[\text{Pi}]$

Out[1] = 3.14159

In[2] := FullForm[p]

Out[2]//FullForm =
3.141592653589793

Precision is the number of significant decimal digits. For machine numbers it is a symbolic constant:

In[3] := Precision[p]

Out[3] = MachinePrecision

In[4] := N[MachinePrecision]

Out[4] = 15.9546

It is 53 bits (or about 16 decimal digits) of precision. In other languages (C, Fortran) such numbers are usually called double precision. Operations with such numbers in *Mathematica* are performed by hardware, as in C or Fortran. They are less efficient than in these languages, but nevertheless rather efficient.

In[5] := MachineNumberQ[p]

Out[5] = True

There are also *arbitrary-precision* numbers. Operations with them are implemented in software and are far less efficient.

In[6] := $p = N[\text{Pi}, 25]$

Out[6] = 3.141592653589793238462643

In[7] := FullForm[p]

Out[7]//FullForm =
3.141592653589793238462643²⁵.

In[8] := Precision[p]

Out[8] = 25.

Precision is a part of a value, not of a variable, as in some other languages. If it is equal to n , then the estimated relative error of the value is 10^{-n} . There is also *accuracy*—the number of significant decimal digits after the point. If it is equal to m , then the estimated absolute error of the value is 10^{-m} .

In[9] := Accuracy[p]

Out[9] = 24.5029

When approximate numbers are added or subtracted, the absolute errors are added. The difference of two approximately equal numbers has a lower precision than the operands.

In[10] := $q = N[355/113, 20]$

Out[10] = 3.1415929203539823009

In[11] := Accuracy[q]

Out[11] = 19.5029

In[12] := $d = p - q$

Out[12] = $-2.667641890624 \times 10^{-7}$

In[13] := {Precision[d], Accuracy[d]}

Out[13] = {12.929, 19.5028}

When approximate numbers are multiplied or divided, the relative errors are added.

In[14] := $r = p/q$

Out[14] = 0.9999999150863285520

In[15] := {Precision[r], Accuracy[r]}

Out[15] = {20., 20.}

In[16] := Clear[p, q, d, r]

This error handling sometimes may be too pessimistic. Let's consider an example [15]. The sequence $x_n = f(x_{n-1})$, $x_1 = 1$,

In[17] := $f[x_] := (x^2 + 4)/(2 * x)$

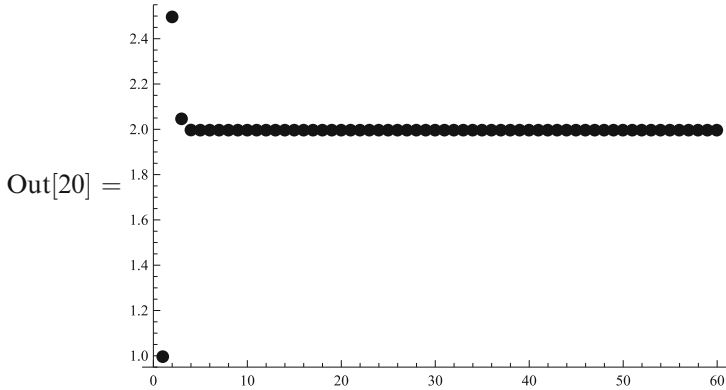
converges to 2. With machine numbers

In[18] := $f[x_, n_] := Module[{t = Table[x, {n}]},$

Do[t[[i]] = f[t[[i - 1]]], {i, 2, n}; t]

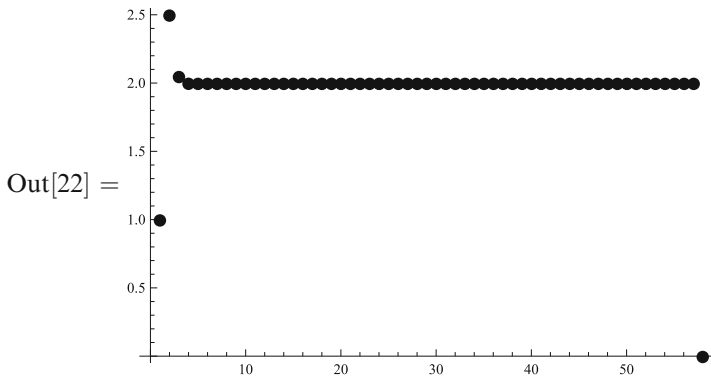
In[19] := $x = f[1.0, 60];$

```
In[20] := ListPlot[x, PlotRange -> {0.95, 2.55},  
PlotMarkers -> {Automatic, Medium}]
```

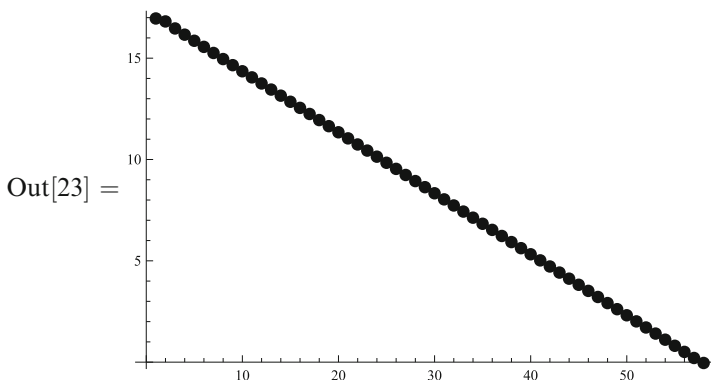


this is indeed so. Now let x_1 be 1.0 with 17-digits precision:

```
In[21] := x = f[1.017, 60];  
In[22] := ListPlot[x, PlotRange -> {-0.05, 2.55},  
PlotMarkers -> {Automatic, Medium}]
```



```
In[23] := ListPlot[Map[Precision, x],  
PlotMarkers -> {Automatic, Medium}]
```



The tail of this list is

In[24] := x[[55;;60]]

Out[24] = {2., 2., 0., 0., ComplexInfinity, Indeterminate}

In[25] := Map[Precision, x[[55;;60]]]

Out[25] = {0.84866, 0.54763, 0.2466, 0., ∞, ∞}

The initial precision is completely lost in 58 iterations.

In[26] := Clear[f, x]

9.2 Solving Equations

NSolve tries to solve equations numerically. They must not contain symbolic parameters, only numbers and unknowns. Only very limited classes of equations can be solved analytically; numerical solution is possible nearly always. The option Reals says to find only real roots.

In[27] := NSolve[x^5 + x + 1 == 0, x]

Out[27] = {{x → -0.754878}, {x → -0.5 - 0.866025i}, {x → -0.5 + 0.866025i},
{x → 0.877439 - 0.744862i}, {x → 0.877439 + 0.744862i}}

In[28] := NSolve[x^5 + x + 1 == 0, x, Reals]

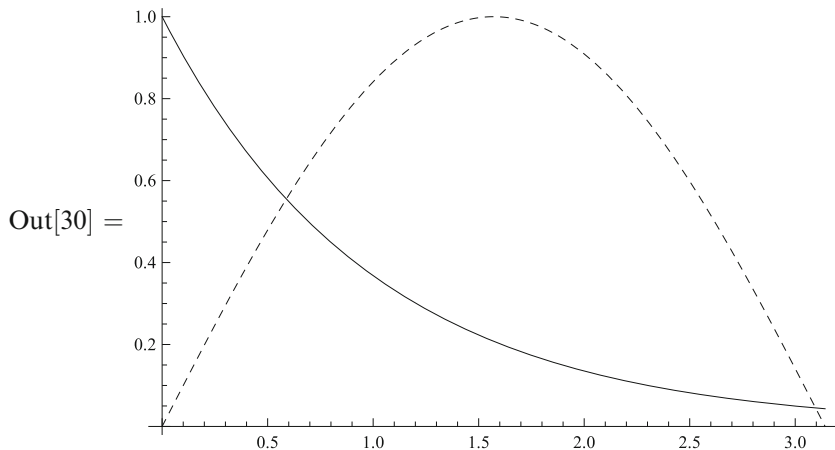
Out[28] = {{x → -0.754878}}

We can add an interval in which we want to find solutions.

In[29] := NSolve[{Exp[-x] == Sin[x], 0 < x < Pi}, x]

Out[29] = {{x → 0.588533}, {x → 3.09636}}

In[30] := Plot[{Exp[-x], Sin[x]}, {x, 0, Pi}]



9.3 Numerical Integration and Summation

The function `NIntegrate` integrates numerically, without trying to do it analytically first—it uses an appropriate numerical method right away. Of course, integration limits must be numbers, and there must be no symbolic parameters.

```
In[31] := NIntegrate[Sin[x]/x, {x, 0, Infinity}]
```

```
Out[31] = 1.5708
```

The option `PrecisionGoal` states the desired precision of the result. If the result is close to 0 due to strong cancellations, it may be difficult to attain a high precision (i.e., a small relative error). Then it is better to specify `AccuracyGoal`, i.e., the desired absolute error. The option `WorkingPrecision` specifies the precision level at which internal calculations are done; it must be \geq `PrecisionGoal`.

```
In[32] := NIntegrate[Exp[-x^2], {x, 0, Infinity}, WorkingPrecision->30]
```

```
Out[32] = 0.886226925452758013649083741785
```

The integration method is selected automatically; however, we can specify it:

```
In[33] := NIntegrate[1/(1 + x*y), {x, 0, 1}, {y, 0, 1},
```

```
Method->"AdaptiveMonteCarlo"]
```

```
Out[33] = 0.822237
```

```
In[34] := i = NIntegrate[Log[x]^2/(x + 1), {x, 0, 1}, PrecisionGoal->30,
```

```
WorkingPrecision->35]
```

```
Out[34] = 1.8030853547393914280996072422671750
```

Suppose we suspect that the integral i is a linear combination of $\zeta(3)$ and 1 with rational coefficients. `FindIntegerNullVector` tries to find integer coefficients such that the linear combination vanishes:

```
In[35] := FindIntegerNullVector[{i, Zeta[3], 1}]
```

```
Out[35] = {2, -3, 0}
```

This means that $2i - 3\zeta(3) = 0$, i.e., our integral is $\frac{3}{2}\zeta(3)$. Of course, this is not a mathematical proof. However, if we increase precision, and the linear combination stays the same, we can be practically sure that the result is correct (this is called *experimental mathematics*).

`NSum` is similar.

```
In[36] := s = NSum[1/n^4, {n, 1, Infinity}, PrecisionGoal->30,
```

```
WorkingPrecision->35, NSumTerms->30]
```

```
Out[36] = 1.0823232337111381915160036965412
```

If we suspect that the sum s is a linear combination of π^4 and 1 with rational coefficients, we can do

```
In[37] := FindIntegerNullVector[{s, Pi^4, 1}]
```

```
Out[37] = {90, -1, 0}
```

This means that our sum is, probably, $\frac{\pi^4}{90}$.

```
In[38] := Clear[i, s]
```

9.4 Differential Equations

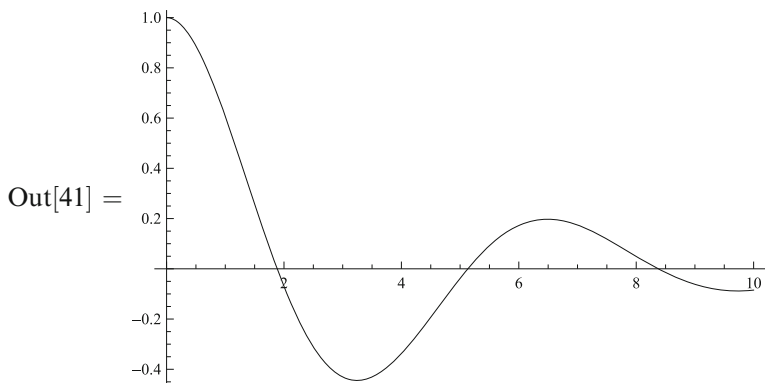
NDSolve solves differential equations numerically, for a finite interval of the independent variable. It returns results in terms of InterpolatingFunction; this result can be numerically evaluated for any value of the independent variable in the given interval.

In[39] := a = 1/2;

In[40] := ns = NDSolve[{y''[t] + a * y'[t] + y[t] == 0, y'[0] == 0, y[0] == 1}, y[t], {t, 0, 10}]

Out[40] = {{y[t] → InterpolatingFunction[{{0., 10.}}, <>][t]}}

In[41] := Plot[y[t]/.ns[[1]], {t, 0, 10}]

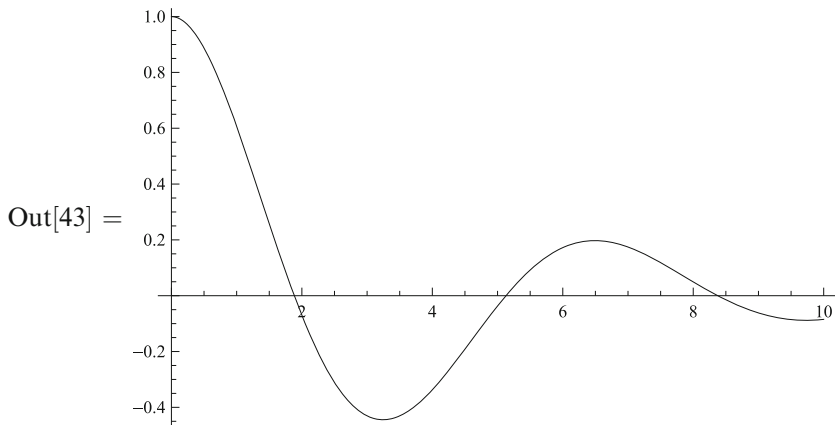


Let's compare with the analytical solution.

In[42] := s = DSolve[{y''[t] + a * y'[t] + y[t] == 0, y'[0] == 0, y[0] == 1}, y[t], t]

Out[42] = $\left\{ \left\{ y[t] \rightarrow \frac{1}{15} e^{-t/4} \left(15 \cos \left[\frac{\sqrt{15}t}{4} \right] + \sqrt{15} \sin \left[\frac{\sqrt{15}t}{4} \right] \right) \right\} \right\}$

In[43] := Plot[y[t]/.s[[1]], {t, 0, 10}]



In[44] := Clear[a, s, ns]

Chapter 10

Risch Algorithm

We were taught at calculus classes that integration is an art, not a science (in contrast to differentiation—even a monkey can be trained to take derivatives). And we were taught wrong. The Risch algorithm (which is known for decades) allows one to find, in a finite number of steps, if a given indefinite integral can be taken in elementary functions, and if so, to calculate it. This algorithm has been constructed in works by an American mathematician Risch near 1970; many cases were not analyzed completely in these works and were later considered by other mathematicians. The algorithm is very complicated, and no computer algebra system implements it fully. Its implementation in *Mathematica* is rather complete, even with extensions to some classes of special functions, but details are not publicly known. Strictly speaking, it is not quite an algorithm, because it contains algorithmically unsolvable subproblems, such as finding out if a given combination of elementary functions vanishes. But in practice computer algebra systems are quite good in solving such problems. Here we shall consider, at a very elementary level, the main ideas of the Risch algorithm; see [16] for more details.

10.1 Rational Functions

We begin with a very simple case—integration of rational functions. Better methods than the partial fraction decomposition exist for this problem. And these methods can be generalized to much wider classes of integrands. Let's consider an integral

$$\int \frac{N(x)}{D(x)} dx,$$

where $N(x)$ and $D(x)$ are polynomials. If $\deg N \geq \deg D$, we can divide with remainder; integration of a polynomial is trivial. Therefore we'll assume $\deg N < \deg D$. The integration result consists of a rational part and a logarithmic one:

$$\int \frac{N(x)}{D(x)} dx = \frac{P(x)}{\hat{D}(x)} + \sum c_i \log(x - a_i),$$

where a_i are the roots of the denominator $D(x)$, and c_i are constants. If

$$D(x) = \prod (x - a_i)^{d_i},$$

then

$$\hat{D}(x) = \prod (x - a_i)^{d_i - 1}.$$

Indeed, at $x \rightarrow a_i$ the rational part has a pole of the order $d_i - 1$; when differentiated, it becomes a pole of the order d_i , as needed. The numerator $P(x)$ is a polynomial of degree $\deg P < \deg \hat{D}$:

$$P(x) = \sum p_n x^n.$$

Substituting all these parts and differentiating, we can find the unknown coefficients c_i and p_n by solving a linear system.

For example, let's calculate

$$\int \frac{dx}{x^2(x-1)} = \frac{p_0}{x} + c_1 \log(x) + c_2 \log(x-1).$$

In[1] := Res = p[0]/x + c[1] * Log[x] + c[2] * Log[x - 1]

Out[1] = c[2]Log[-1 + x] + c[1]Log[x] + $\frac{p[0]}{x}$

In[2] := Eq = Together[x^2 * (x - 1) * D[Res, x] - 1]

Out[2] = -1 - xc[1] + x^2c[1] + x^2c[2] + p[0] - xp[0]

In[3] := Eqs = Table[Coefficient[Eq, x, n] == 0, {n, 0, 2}]

Out[3] = {-1 + p[0] == 0, -c[1] - p[0] == 0, c[1] + c[2] == 0}

In[4] := Sol = Solve[Eqs, {p[0], c[1], c[2]}][[1]]

Out[4] = {p[0] -> 1, c[1] -> -1, c[2] -> 1}

In[5] := Res /. Sol

Out[5] = $\frac{1}{x}$ + Log[-1 + x] - Log[x]

In[6] := Clear[Res, Eq, Eqs, Sol]

10.2 Logarithmic Extension

Now we begin to extend the class of integrands and consider

$$\int \frac{N(x, y)}{D(x, y)} dx,$$

where y depends on x (in a nonrational way). The extension is called algebraic if y is a root of a polynomial equation $p(x, y) = 0$. For example, $y = \sqrt[n]{p(x)/q(x)}$ is a root of the equation $q(x)y^n - p(x) = 0$. An algorithm for integration of expressions belonging to algebraic extensions has been constructed, but it requires an advanced mathematical apparatus [17], and we shall not discuss it here.

Extensions which are not algebraic are called transcendental. There are two important classes of such extensions. A logarithmic extension $y = \log r(x)$ (where $r(x)$ is a rational function) is characterized by the property $y' = r'/r$. An exponential extension $y = \exp r(x)$ —by $y' = r'y$.

If an integral of an expression from a logarithmic extension with some $y = \log r(x)$ can be taken in elementary functions, it has the form

$$\int \frac{N(x, y)}{D(x, y)} dx = \frac{P(x, y)}{\hat{D}(x, y)} + \sum c_i \log q_i,$$

where

$$D = \prod D_i^{d_i} \Rightarrow \hat{D} = \prod D_i^{d_i-1},$$

q_i are the irreducible factors of all D_i , c_i are constants,

$$P(x, y) = \sum p_n(x)y^n = \sum p_{mn}x^m y^n$$

is a polynomial with unknown (so far) coefficients. Differentiating this general form of the result, putting everything over a common denominator, and equating coefficients of $x^m y^n$, we obtain a linear system for finding all unknown coefficients. If this system is incompatible, this means that the integral cannot be taken in elementary functions.

Example 1

We shall consider several examples. Let $y = \log x$ so that $y' = 1/x$. Let's calculate the integral

$$\int y dx = p_2(x)y^2 + p_1(x)y + p_0(x).$$

When differentiated, the degree in y reduces by 1, so that the result is quadratic in y (there is no denominator, and hence no q_i).

In[7] := y'[x_] := 1/x

In[8] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 2}]

Out[8] = p[0][x] + y[x]p[1][x] + y[x]^2 p[2][x]

In[9] := Eq = D[Res, x] - y[x]

Out[9] = -y[x] + $\frac{p[1][x]}{x}$ + $\frac{2y[x]p[2][x]}{x}$ + p[0]'[x] + y[x]p[1]'[x] + y[x]^2 p[2]'[x]

In[10] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 2}]

$$\text{Out[10]} = \left\{ \frac{p[1][x]}{x} + p[0]'[x] == 0, -1 + \frac{2p[2][x]}{x} + p[1]'[x] == 0, p[2]'[x] == 0 \right\}$$

We see that $p_2(x)$ is a constant:

In[11] := p[2][x.] := p[2, 0]; Eqs

$$\text{Out[11]} = \left\{ \frac{p[1][x]}{x} + p[0]'[x] == 0, -1 + \frac{2p[2, 0]}{x} + p[1]'[x] == 0, \text{True} \right\}$$

Since $p_1(x)$ is a polynomial, the second equation can be satisfied only if $p_{20} = 0$:

In[12] := p[2, 0] = 0; Eqs

$$\text{Out[12]} = \left\{ \frac{p[1][x]}{x} + p[0]'[x] == 0, -1 + p[1]'[x] == 0, \text{True} \right\}$$

The second equation gives

In[13] := p[1][x.] := x + p[1, 0]

Now the first equation

In[14] := Eq1 = ExpandAll[Eqs[[1]]]

$$\text{Out[14]} = 1 + \frac{p[1, 0]}{x} + p[0]'[x] == 0$$

gives $p_{10} = 0$:

In[15] := p[1, 0] = 0; Eq1

$$\text{Out[15]} = 1 + p[0]'[x] == 0$$

Therefore $p_0(x) = -x$ (omitting the integration constant):

In[16] := p[0][x.] := -x; Eq1

$$\text{Out[16]} = \text{True}$$

In[17] := Res

$$\text{Out[17]} = -x + xy[x]$$

In[18] := Clear[Res, Eq, Eqs, Eq1, p]

We have derived the well-known result

$$\int \log(x) dx = x \log(x) - x.$$

Consider the integrals

$$\int x \log(x) dx, \quad \int \log^2(x) dx$$

in a similar way.

Example 2

Let's calculate the integral

$$\int \frac{y}{x} dx = p_2(x)y^2 + p_1(x)y + p_0(x).$$

Here $\hat{D} = 1$; it seems that there is a single q , namely x , but $\log(x) = y$, so that the logarithmic part contributes nothing new.

In[19] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 2}];

In[20] := Eq = D[Res, x] - y[x]/x

Out[20] = $-\frac{y[x]}{x} + \frac{p[1][x]}{x} + \frac{2y[x]p[2][x]}{x} + p[0]'[x] + y[x]p[1]'[x] + y[x]^2p[2]'[x]$

In[21] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 2}]

Out[21] = $\left\{ \frac{p[1][x]}{x} + p[0]'[x] == 0, -\frac{1}{x} + \frac{2p[2][x]}{x} + p[1]'[x] == 0, p[2]'[x] == 0 \right\}$

In[22] := p[2][x_] := p[2, 0]; Eqs

Out[22] = $\left\{ \frac{p[1][x]}{x} + p[0]'[x] == 0, -\frac{1}{x} + \frac{2p[2, 0]}{x} + p[1]'[x] == 0, \text{True} \right\}$

From the second equation, $p_{20} = 1/2$; then p_1 is a constant:

In[23] := p[2, 0] = 1/2; Eqs[[2]]

Out[23] = $p[1]'[x] == 0$

In[24] := p[1][x_] = p[1, 0]; Eqs

Out[24] = $\left\{ \frac{p[1, 0]}{x} + p[0]'[x] == 0, \text{True}, \text{True} \right\}$

From the first equation, $p_{10} = 0$; then p_0 is a constant (which may be omitted):

In[25] := p[1, 0] = 0; Eqs[[1]]

Out[25] = $p[0]'[x] == 0$

In[26] := p[0][x_] := 0; Res

Out[26] = $\frac{y[x]^2}{2}$

In[27] := Clear[Res, Eq, Eqs, p]

We have derived the well-known result

$$\int \frac{\log(x)}{x} dx = \frac{1}{2} \log^2(x).$$

Example 3

Let's change the previous integral a little:

$$\int \frac{y}{x+1} dx = p_2(x)y^2 + p_1(x)y + p_0(x) + c \log(x+1).$$

In[28] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 2}] + c * Log[x + 1]

Out[28] = $c, \text{Log}[1 + x] + p[0][x] + y[x]p[1][x] + y[x]^2p[2][x]$

In[29] := Eq = D[Res, x] - y[x]/(x + 1)

Out[29] = $\frac{c}{1+x} - \frac{y[x]}{1+x} + \frac{p[1][x]}{x} + \frac{2y[x]p[2][x]}{x} + p[0]'[x] + y[x]p[1]'[x] + y[x]^2p[2]'[x]$

In[30] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 2}]

$$\text{Out[30]} = \left\{ \frac{c}{1+x} + \frac{p[1][x]}{x} + p[0]'[x] == 0, -\frac{1}{1+x} + \frac{2p[2][x]}{x} + p[1]'[x] == 0, \right. \\ \left. p[2]'[x] == 0 \right\}$$

As in the previous examples, $p_2(x)$ is a constant:

In[31] := p[2][x.] := p[2, 0]; Eqs

$$\text{Out[31]} = \left\{ \frac{c}{1+x} + \frac{p[1][x]}{x} + p[0]'[x] == 0, -\frac{1}{1+x} + \frac{2p[2, 0]}{x} + p[1]'[x] == 0, \right. \\ \left. \text{True} \right\}$$

A polynomial $p_1(x)$ satisfying the second equation does not exist. Therefore, this integral cannot be taken in elementary functions.

In[32] := Clear[Res, Eq, Eqs, p]

Example 4

Let's consider

$$\int \frac{dx}{y} = p_1(x)y + p_0(x) + c \log(y)$$

(it is not quite clear what the degree of the right-hand side in y should be; we shall see in a moment that this is irrelevant).

In[33] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}] + c * Log[y[x]]

$$\text{Out[33]} = c, \text{Log}[y[x]] + p[0][x] + y[x]p[1][x]$$

In[34] := Eq = Expand[y[x] * D[Res, x] - 1]

$$\text{Out[34]} = -1 + \frac{c}{x} + \frac{y[x]p[1][x]}{x} + y[x]p[0]'[x] + y[x]^2p[1]'[x]$$

In[35] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 2}]

$$\text{Out[35]} = \left\{ -1 + \frac{c}{x} == 0, \frac{p[1][x]}{x} + p[0]'[x] == 0, p[1]'[x] == 0 \right\}$$

From the last equation, $p_1(x)$ is a constant. The previous equation shows that it is 0, and $p_0(x)$ is a constant (it may be set to 0). It is clear that if we started from some other degree in y , we would all the same find that all $p_n(x) = 0$. And the first equation cannot be solved for c .

In[36] := Clear[Res, Eq, Eqs]

Consider the integrals

$$\int \frac{dx}{xy}, \quad \int \frac{dx}{y+1}$$

in a similar way.

10.3 Exponential Extension

Now we shall consider an exponential extension with some $y = \exp r(x)$. If an integral of an expression from this extension can be taken in elementary functions, it has the form

$$\int \frac{N(x,y)}{D(x,y)} dx = \frac{P(x,y)}{\hat{D}(x,y)} + \sum c_i \log q_i,$$

$$D = \prod D_i^{d_i} \quad \Rightarrow \quad \hat{D} = \prod D_i^{\hat{d}_i},$$

where $\hat{d}_i = d_i - 1$ always except the case $D_i = y$ in which $\hat{d}_i = d_i$. This is because the derivative of $1/y$ is proportional to $1/y$, the degree of y in the denominator does not increase. We exclude y from the list of the factors q_i (if it was present, of course) because $\log y = r(x)$ is a rational function, and such a contribution is already accounted for. As usual, we differentiate the result and equate to the integrand to obtain a linear system. If it cannot be solved, then the integral does not exist in elementary functions.

Example 1

Let $y = e^x$:

In[37] := $y'[x.] := y[x]$

Let's calculate

$$\int y dx = p_1(x)y + p_0(x)$$

(the degree in y does not change when differentiating; therefore the polynomial in y in the right-hand side should have the same degree as the integrand).

In[38] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}]

Out[38] = $p[0][x] + y[x]p[1][x]$

In[39] := Eq = D[Res, x] - y[x]

Out[39] = $-y[x] + y[x]p[1][x] + p[0]'[x] + y[x]p[1]'[x]$

In[40] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]

Out[40] = $\{p[0]'[x] == 0, -1 + p[1][x] + p[1]'[x] == 0\}$

In[41] := p[1][x.] := Sum[p[1, m] * x^m, {m, 0, 1}]; Eqs[[2]]

Out[41] = $-1 + p[1, 0] + p[1, 1] + xp[1, 1] == 0$

From this equation:

In[42] := p[1, 1] = 0; p[1, 0] = 1; Eqs

Out[42] = $\{p[0]'[x] == 0, \text{True}\}$

Therefore $p_0(x)$ is a constant (which may be omitted):

In[43] := p[0][x.] := 0; Res

Out[43] = $y[x]$

We've got the expected result.

In[44] := Clear[Res, Eq, Eqs, p]

Example 2

Now let's calculate

$$\int xy \, dx = p_1(x)y + p_0(x).$$

In[45] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}]

Out[45] = $p[0][x] + y[x]p[1][x]$

In[46] := Eq = D[Res, x] - x * y[x]

Out[46] = $-xy[x] + y[x]p[1][x] + p[0]'[x] + y[x]p[1]'[x]$

In[47] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]

Out[47] = $\{p[0]'[x] == 0, -x + p[1][x] + p[1]'[x] == 0\}$

In[48] := p[1][x_] := Sum[p[1, m] * x^m, {m, 0, 1}]; Eqs[[2]]

Out[48] = $-x + p[1, 0] + p[1, 1] + xp[1, 1] == 0$

From this equation:

In[49] := p[1, 1] = 1; p[1, 0] = -1; Eqs

Out[49] = $\{p[0]'[x] == 0, \text{True}\}$

In[50] := p[0][x_] := 0; Res

Out[50] = $(-1 + x)y[x]$

And without integration by parts!

In[51] := Clear[Res, Eq, Eqs, p]

Consider

$$\int x^2 y \, dx$$

in a similar way.

Example 3

Now let's try to calculate

$$\int \frac{y}{x} \, dx = p_1(x)y + p_0(x) + c \log x.$$

In[52] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}] + c * Log[x]

Out[52] = $c, \text{Log}[x] + p[0][x] + y[x]p[1][x]$

In[53] := Eq = D[Res, x] - y[x]/x

Out[53] = $\frac{c}{x} - \frac{y[x]}{x} + y[x]p[1][x] + p[0]'[x] + y[x]p[1]'[x]$

In[54] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]
Out[54] = $\left\{ \frac{c}{x} + p[0]'[x] == 0, -\frac{1}{x} + p[1][x] + p[1]'[x] == 0 \right\}$

A polynomial $p_1(x)$ satisfying the second equation does not exist. Therefore this integral cannot be taken in elementary functions.

In[55] := Clear[Res, Eq, Eqs]

Example 4

Let's calculate

$$\int \frac{dx}{y} = \frac{p_1(x)y + p_0(x)}{y}$$

(here $\hat{D} = y$, and there are no q_i). Of course, we could denote e^{-x} as y , and the problem would reduce to the Example 1 with trivial modifications; but we want to observe how the algorithm works in this new case.

In[56] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}]/y[x]

Out[56] = $\frac{p[0][x] + y[x]p[1][x]}{y[x]}$

In[57] := Eq = Expand[y[x] * D[Res, x] - 1]

Out[57] = $-1 - p[0][x] + p[0]'[x] + y[x]p[1]'[x]$

In[58] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]

Out[58] = $\{-1 - p[0][x] + p[0]'[x] == 0, p[1]'[x] == 0\}$

From the second equation, $p_1(x)$ is a constant, which may be omitted—this is the integration constant.

In[59] := p[1][x_] := 0

In[60] := p[0][x_] := Sum[p[0, m] * x^m, {m, 0, 1}]; Eqs[[1]]

Out[60] = $-1 - p[0, 0] + p[0, 1] - xp[0, 1] == 0$

Therefore

In[61] := p[0, 1] = 0; p[0, 0] = -1; Res

Out[61] = $-\frac{1}{y[x]}$

In[62] := Clear[Res, Eq, Eqs, p]

Example 5

Let's consider

$$\int \frac{dx}{y-1} = p_1(x)y + p_0(x) + c \log(y-1)$$

(now $\hat{D} = 1$, and there is a single q , namely $y-1$).

In[63] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}] + c * Log[y[x] - 1]

Out[63] = c, Log[-1 + y[x]] + p[0][x] + y[x]p[1][x]

In[64] := Eq = Cancel[(y[x] - 1) * D[Res, x]] - 1

Out[64] = -1 + cy[x] - y[x]p[1][x] + y[x]^2p[1][x] - p[0]'[x] + y[x]p[0]'[x] - y[x]p[1]'[x] + y[x]^2p[1]'[x]

In[65] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 2}]

Out[65] = {-1 - p[0]'[x] == 0, c - p[1][x] + p[0]'[x] - p[1]'[x] == 0, p[1][x] + p[1]'[x] == 0}

In[66] := p[1][x_] := Sum[p[1, m] * x^m, {m, 0, 1}]; Eqs[[3]]

Out[66] = p[1, 0] + p[1, 1] + xp[1, 1] == 0

Therefore

In[67] := p[1, 1] = 0; p[1, 0] = 0; Eqs

Out[67] = {-1 - p[0]'[x] == 0, c + p[0]'[x] == 0, True}

From the first equation, $p_0(x) = -x$ (omitting the integration constant).

In[68] := p[0][x_] := -x; Eqs

Out[68] = {True, -1 + c == 0, True}

In[69] := c = 1; Res

Out[69] = -x + Log[-1 + y[x]]

In[70] := Clear[Res, Eq, Eqs, p, c]

Consider

$$\int \frac{x}{y-1} dx$$

in a similar way and demonstrate that this integral does not exist in elementary functions.

Example 6

Of course, the method can be also used for other exponential extensions. For example, let $y = \exp x^2$:

In[71] := y'[x_] := 2 * x * y[x]

Let's consider

$$\int y dx = p_1(x)y + p_0(x).$$

In[72] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}];

In[73] := Eq = D[Res, x] - y[x]

Out[73] = -y[x] + 2xy[x]p[1][x] + p[0]'[x] + y[x]p[1]'[x]

In[74] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]

Out[74] = {p[0]'[x] == 0, -1 + 2xp[1][x] + p[1]'[x] == 0}

In[75] := p[1][x_] := Sum[p[1, m] * x^m, {m, 0, 1}]; ExpandAll[Eqs[[2]]]

Out[75] = -1 + 2xp[1, 0] + p[1, 1] + 2x^2p[1, 1] == 0

This equation cannot be solved. A larger degree of $p_1(x)$ does not help. Therefore this integral cannot be taken in elementary functions.

In[76] := Clear[Res, Eq, Eqs, p]

Example 7

And what about

$$\int xy \, dx = p_1(x)y + p_0(x) ?$$

In[77] := Res = Sum[p[n][x] * y[x]^n, {n, 0, 1}];

In[78] := Eq = D[Res, x] - x * y[x]

Out[78] = $-xy[x] + 2xy[x]p[1][x] + p[0]'[x] + y[x]p[1]'[x]$

In[79] := Eqs = Table[Coefficient[Eq, y[x], n] == 0, {n, 0, 1}]

Out[79] = $\{p[0]'[x] == 0, -x + 2xp[1][x] + p[1]'[x] == 0\}$

In[80] := p[1][x_] := Sum[p[1, m] * x^m, {m, 0, 1}]; ExpandAll[Eqs[[2]]]

Out[80] = $-x + 2xp[1, 0] + p[1, 1] + 2x^2p[1, 1] == 0$

From this equation, $p_{11} = 0$, $p_{10} = 1/2$. The first equation says that $p_0(x)$ is a constant (it may be omitted).

In[81] := p[1, 1] = 0; p[1, 0] = 1/2; p[0][x_] := 0; Res

Out[81] = $\frac{y[x]}{2}$

In[82] := Clear[Res, Eq, Eqs, p]

Consider

$$\int x^2 y \, dx, \quad \int \frac{y}{x} \, dx$$

in a similar way.

10.4 Elementary Functions

A tower of extensions can be constructed. We start from the set of rational functions $N(x)/D(x)$. Then we introduce y_1 , which is either a root of a polynomial equation $p(x, y_1) = 0$, or logarithm or exponent of some rational function of x , and we obtain a first extension—the set of functions $N(x, y_1)/D(x, y_1)$. Then we introduce y_2 , which is either a root of a polynomial equation $p(x, y_1, y_2) = 0$, or logarithm or exponent of some function from the previous extension, and we get a next extension—the set of rational functions of x, y_1, y_2 . And so on. We should take care that an extension which seems transcendental is not in fact algebraic; if we neglect this, the methods designed for transcendental extensions may break down, e.g., produce divisions by 0. For example, if $y_1 = e^x$, then it would not be a good idea to introduce the exponential extension with $y_2 = e^{2x}$, because y_2 is not algebraically independent: $y_2 = y_1^2$. Similarly, if $y_1 = \log x$, then it's not reasonable to introduce the logarithmic extension with $y_2 = \log 2x$, because $y_2 = y_1 + \log 2$ (in this case the field of constants needs to be extended by the transcendental number $\log 2$). In simple cases such restrictions are obvious, but in complicated ones it is necessary to decide if some function from some extension is identical 0, and this problem is algorithmically unsolvable in general.

Such towers of algebraic, logarithmic, and exponential extensions include all functions called elementary. And even some extra ones: an algebraic extension can be defined, e.g., by a root of a fifth degree polynomial unsolvable in radicals. Indeed, trigonometric functions reduce to exponentials, and inverse trigonometric ones—to logarithms. For each elementary function there exists a tower of extensions to which it belongs (it is not unique).

Suppose we want to integrate an elementary function. We construct a tower of extensions to which it belongs. If the indefinite integral exists in elementary functions, it belongs to some further extension of our tower by some extra logarithms (their number may be zero). The Risch algorithm allows one to decide in a finite number of steps if the result exists in this further extension, and if so, to find it; if it does not exist, the algorithm proves this fact. In its classical form, the algorithm is recursive in extensions—it calls itself for solving integration subproblems in previous (smaller) extensions, until rational functions. There is a simpler and more efficient version of the Risch algorithm—to write down the general form of the result with unknown coefficients, differentiate it and equate to the integrand. Then the problem reduces to solving a linear system. This approach is guaranteed to be correct if we know upper bounds on the degrees of the polynomial $P(x, y_1, y_2, \dots)$ in its variables. But such upper bounds are not always known (as we have seen in the examples, they are known if there is no denominator). Therefore some heuristic rules to bound the degrees of P are used. This can give a situation when no result is found, though it really exists (but has a larger degree in some variables).

The Risch algorithm is an outstanding achievement of mathematics in the twentieth century. But it does not solve all problems with indefinite integration. The answer that no result exists in elementary functions is not very useful. It would be much better to get the result with some special functions. There were attempts to generalize the Risch algorithm to some special functions (the error function, polylogarithms). Some of them are implemented in *Mathematica*.

Chapter 11

Linear Algebra

11.1 Constructing Matrices

A matrix in *Mathematica* is a list of lists; all the lists—rows of the matrix—must have the same length.

In[1] := M = {{a,b},{c,d}}

Out[1] = {{a,b},{c,d}}

MatrixForm is used to print a matrix nicely.

In[2] := MatrixForm[M]

Out[2]//MatrixForm =

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

If the (i, j) th element of a matrix is given by an expression depending on i and j , this matrix can be constructed by the function Table.

In[3] := A = Table[a[i,j],{i,1,2},{j,1,2}]

Out[3] = {{a[1,1],a[1,2]},{a[2,1],a[2,2]}}

In[4] := B = Table[1/(i+j+1),{i,1,2},{j,1,2}]

Out[4] = $\left\{ \left\{ \frac{1}{3}, \frac{1}{4} \right\}, \left\{ \frac{1}{4}, \frac{1}{5} \right\} \right\}$

In[5] := MatrixForm[A]

Out[5]//MatrixForm =

$$\begin{pmatrix} a[1,1] & a[1,2] \\ a[2,1] & a[2,2] \end{pmatrix}$$

In[6] := MatrixForm[B]

Out[6]//MatrixForm =

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{5} \\ \frac{1}{3} & \frac{1}{4} \end{pmatrix}$$

The function Array is similar, but its first argument is a function of two parameters, not an expression. It may be just a symbol or an anonymous function (Function).

In[7] := A = Array[a, {2,2}]

Out[7] = {{a[1,1],a[1,2]},{a[2,1],a[2,2]}}

In[8] := B = Array[Function[{i, j}, 1/(i + j + 1)], {2, 2}]

Out[8] = $\left\{ \left\{ \frac{1}{3}, \frac{1}{4} \right\}, \left\{ \frac{1}{4}, \frac{1}{5} \right\} \right\}$

Mathematica does not distinguish column vectors and row vectors; both are just lists.

In[9] := V = Table[v[i], {i, 1, 2}]

Out[9] = $\{v[1], v[2]\}$

In[10] := V = Array[v, 2]

Out[10] = $\{v[1], v[2]\}$

In[11] := U = Array[u, 2]

Out[11] = $\{u[1], u[2]\}$

11.2 Parts of a Matrix

A matrix element.

In[12] := A[[1, 2]]

Out[12] = $a[1, 2]$

A row.

In[13] := A[[1]]

Out[13] = $\{a[1, 1], a[1, 2]\}$

A column.

In[14] := A[[All, 2]]

Out[14] = $\{a[1, 2], a[2, 2]\}$

A new value can be assigned to a matrix element.

In[15] := M[[1, 2]] = 0

Out[15] = 0

In[16] := MatrixForm[M]

Out[16]//MatrixForm =

$$\begin{pmatrix} a & 0 \\ c & d \end{pmatrix}$$

Add 1 to the first row.

In[17] := M[[1]] +=

Out[17] = $\{a, 0\}$

In[18] := MatrixForm[M]

Out[18]//MatrixForm =

$$\begin{pmatrix} 1 + a & 1 \\ c & d \end{pmatrix}$$

Add the second column to the first one.

In[19] := M[[All, 1]] += M[[All, 2]]

Out[19] = $\{2 + a, c + d\}$

In[20] := MatrixForm[M]

Out[20]//MatrixForm =

$$\begin{pmatrix} 2 + a & 1 \\ c + d & d \end{pmatrix}$$

11.3 Queries

The function `VectorQ` checks if its argument is a vector, i.e., a list whose elements are not lists.

In[21] := {VectorQ[V], VectorQ[M]}

Out[21] = {True, False}

The function `MatrixQ` checks if its argument is a matrix, i.e., a list of same-length lists.

In[22] := {MatrixQ[V], MatrixQ[M], MatrixQ[{{a, b}, {x, y, z}]}

Out[22] = {False, True, False}

The argument of the function `Dimensions` must be a matrix. This function returns a two-element list—the numbers of rows and columns of the matrix.

In[23] := Dimensions[M]

Out[23] = {2, 2}

It can be conveniently used for simultaneous assignment to two variables.

In[24] := {n1, n2} = Dimensions[{{a, b, c}, {x, y, z}]

Out[24] = {2, 3}

In[25] := n1

Out[25] = 2

In[26] := n2

Out[26] = 3

In[27] := Clear[M, n1, n2]

11.4 Operations with Matrices and Vectors

Vectors can be added and multiplied by scalar expressions.

In[28] := 2 * V + U

Out[28] = {u[1] + 2v[1], u[2] + 2v[2]}

Matrices can be added and multiplied by scalar expressions.

In[29] := MatrixForm[A + 2 * B]

Out[29] // MatrixForm =

$$\begin{pmatrix} \frac{2}{3} + a[1, 1] & \frac{1}{2} + a[1, 2] \\ \frac{1}{2} + a[2, 1] & \frac{2}{5} + a[2, 2] \end{pmatrix}$$

The scalar product of two vectors.

In[30] := V.U

Out[30] = u[1]v[1] + u[2]v[2]

The product of a matrix and a column vector.

In[31] := A.V

Out[31] = {a[1, 1]v[1] + a[1, 2]v[2], a[2, 1]v[1] + a[2, 2]v[2]}

The product of a row vector and a matrix.

In[32] := V.A

Out[32] = {a[1, 1]v[1] + a[2, 1]v[2], a[1, 2]v[1] + a[2, 2]v[2]}

The product of two matrices.

In[33] := MatrixForm[A.B]

Out[33]//MatrixForm =

$$\begin{pmatrix} \frac{1}{3}a[1,1] + \frac{1}{4}a[1,2] & \frac{1}{4}a[1,1] + \frac{1}{5}a[1,2] \\ \frac{1}{3}a[2,1] + \frac{1}{4}a[2,2] & \frac{1}{4}a[2,1] + \frac{1}{5}a[2,2] \end{pmatrix}$$

It is not commutative.

In[34] := MatrixForm[A.B - B.A]

Out[34]//MatrixForm =

$$\begin{pmatrix} \frac{1}{4}a[1,2] - \frac{1}{4}a[2,1] & \frac{1}{4}a[1,1] - \frac{2}{15}a[1,2] - \frac{1}{4}a[2,2] \\ -\frac{1}{4}a[1,1] + \frac{2}{15}a[2,1] + \frac{1}{4}a[2,2] & -\frac{1}{4}a[1,2] + \frac{1}{4}a[2,1] \end{pmatrix}$$

Determinant (the matrix must be square).

In[35] := Det[A]

Out[35] = $-a[1,2]a[2,1] + a[1,1]a[2,2]$

Trace (the matrix must be square).

In[36] := Tr[A]

Out[36] = $a[1,1] + a[2,2]$

Transposing.

In[37] := MatrixForm[Transpose[A]]

Out[37]//MatrixForm =

$$\begin{pmatrix} a[1,1] & a[2,1] \\ a[1,2] & a[2,2] \end{pmatrix}$$

The inverse matrix.

In[38] := MatrixForm[Inverse[A]]

Out[38]//MatrixForm =

$$\begin{pmatrix} \frac{a[2,2]}{-a[1,2]a[2,1] + a[1,1]a[2,2]} & -\frac{a[1,2]}{-a[1,2]a[2,1] + a[1,1]a[2,2]} \\ -\frac{a[2,1]}{-a[1,2]a[2,1] + a[1,1]a[2,2]} & \frac{a[1,1]}{-a[1,2]a[2,1] + a[1,1]a[2,2]} \end{pmatrix}$$

A square matrix can be raised to an integer power.

In[39] := MatrixForm[MatrixPower[B, 3]]

Out[39]//MatrixForm =

$$\begin{pmatrix} 197 & 1009 \\ 2160 & 14400 \\ 1009 & 323 \\ 14400 & 6000 \end{pmatrix}$$

The power -1 is the inverse matrix.

In[40] := MatrixForm[MatrixPower[B, -1]]

Out[40]//MatrixForm =

$$\begin{pmatrix} 48 & -60 \\ -60 & 80 \end{pmatrix}$$

This is the solution of the linear system $A.X = V$.

In[41] := Together[Inverse[A].V]

Out[41] = $\left\{ \frac{a[2,2]v[1] - a[1,2]v[2]}{-a[1,2]a[2,1] + a[1,1]a[2,2]}, \frac{a[2,1]v[1] - a[1,1]v[2]}{a[1,2]a[2,1] - a[1,1]a[2,2]} \right\}$

The same can be done using LinearSolve.

In[42] := LinearSolve[A, V]

Out[42] = $\left\{ \frac{a[2,2]v[1] - a[1,2]v[2]}{-a[1,2]a[2,1] + a[1,1]a[2,2]}, \frac{a[2,1]v[1] - a[1,1]v[2]}{a[1,2]a[2,1] - a[1,1]a[2,2]} \right\}$

In[43] := Clear[A, B, U, V]

11.5 Eigenvalues and Eigenvectors

Here is some symbolic matrix.

```
In[44] := MatrixForm[M =
  {{(1-x)^3*(3+x), 4*x*(1-x^2), -2*(1-x^2)*(3-x)},
  {4*x*(1-x^2), -(1+x)^3*(3-x), 2*(1-x^2)*(3+x)},
  {-2*(1-x^2)*(3-x), 2*(1-x^2)*(3+x), 16*x}}
```

```
Out[44] // MatrixForm =
  ( (1-x)^3(3+x)   4x(1-x^2)   -2(3-x)(1-x^2) )
  ( 4x(1-x^2)     -(3-x)(1+x)^3  2(3+x)(1-x^2) )
  ( -2(3-x)(1-x^2) 2(3+x)(1-x^2)   16x )
```

It is singular.

```
In[45] := Det[M]
```

```
Out[45] = 0
```

Its rank.

```
In[46] := MatrixRank[M]
```

```
Out[46] = 2
```

The function `NullSpace` returns a list of vectors forming a basis of the null space of the matrix, i.e., the subspace of vectors nullified by the matrix.

```
In[47] := s = NullSpace[M]
```

```
Out[47] = { { -2/(1+x), 2/(1+x), 1 } }
```

In this case, the null space is one-dimensional—it has a single basis vector. Let's check it.

```
In[48] := Together[M.s[[1]]]
```

```
Out[48] = {0, 0, 0}
```

The function `Eigenvalues` returns a list of eigenvalues of a matrix.

```
In[49] := Simplify[Eigenvalues[M], Element[x, Reals]]
```

```
Out[49] = { 0, (3+x^2)^2, -(3+x^2)^2 }
```

We have added the second argument to `Simplify` which informs *Mathematica* that the variable x is real. The function `Eigenvectors` returns the list of the corresponding eigenvectors (in the same order).

```
In[50] := Simplify[Eigenvectors[M], Element[x, Reals]]
```

```
Out[50] = { { -2/(1+x), 2/(1+x), 1 }, { (-1+x)/(1+x), (1-x)/2, 1 }, { (1+x)/2, (1+x)/(-1+x), 1 } }
```

The function `Eigensystem` returns both. It is convenient for simultaneous assignment to two variables.

```
In[51] := {val, vec} = Simplify[Eigensystem[M], Element[x, Reals]];
```

Let's check.

```
In[52] := Do[Print[Simplify[M.vec[[i]] - val[[i]] * vec[[i]]], {i, 1, 3}]
```

```
{0, 0, 0}
```

```
{0, 0, 0}
```

```
{0, 0, 0}
```

```
In[53] := Clear[M, s, val, vec]
```

11.6 Jordan Form

Here is a matrix of rational numbers.

```
In[54] := MatrixForm[M =
  {{13/9, -2/9, 1/3, 4/9, 2/3},
   {-2/9, 10/9, 2/15, -2/9, -11/15},
   {1/5, -2/5, 41/25, -2/5, 12/25},
   {4/9, -2/9, 14/15, 13/9, -2/15},
   {-4/15, 8/15, 12/25, 8/15, 34/25}}]
```

```
Out[54] // MatrixForm =
  ( 13/9  -2/9  1/3  4/9  2/3
    -2/9  10/9  2/15 -2/9 -11/15
     1/5  -2/5  41/25 -2/5  12/25
     4/9  -2/9  14/15 13/9  -2/15
    -4/15 8/15  12/25 8/15  34/25 )
```

The function `JordanDecomposition` returns a pair of matrices—the Jordan form J and the transformation matrix P which reduces our matrix to its Jordan form.

```
In[55] := {P, J} = JordanDecomposition[M];
```

```
In[56] := MatrixForm[J]
```

```
Out[56] // MatrixForm =
  ( 1 0 0 0 0
    0 2 1 0 0
    0 0 2 0 0
    0 0 0 1 - i  0
    0 0 0 0 1 + i )
```

```
In[57] := MatrixForm[P]
```

```
Out[57] // MatrixForm =
  ( -2  1  0  5i/12  -5i/12
    -2 -1/2 0 -5i/6  5i/6
     0  0  6/5  -3/4  -3/4
     1  1  0 -5i/6  5i/6
     0  0  9/10 1  1 )
```

Let's check.

```
In[58] := MatrixForm[P.J.Inverse[P] - M]
```

```
Out[58] // MatrixForm =
  ( 0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0
    0 0 0 0 0 )
```

Here are the eigenvalues and the eigenvectors of our matrix. Note that only one eigenvector corresponds to the eigenvalue 2, because the corresponding Jordan block has the size 2×2 . The eigenvectors are the columns of the transformation matrix P .

```

In[59] := {val, vec} = Eigensystem[M];
In[60] := val
Out[60] = {2, 2, 1 + i, 1 - i, 1}
In[61] := vec
Out[61] =  $\left\{ \left\{ 1, -\frac{1}{2}, 0, 1, 0 \right\}, \{0, 0, 0, 0, 0\}, \left\{ -\frac{5i}{12}, \frac{5i}{6}, -\frac{3}{4}, \frac{5i}{6}, 1 \right\}, \right.$ 
 $\left. \left\{ \frac{5i}{12}, -\frac{5i}{6}, -\frac{3}{4}, -\frac{5i}{6}, 1 \right\}, \{-2, -2, 0, 1, 0\} \right\}$ 
In[62] := Clear[M, J, P, val, vec]

```

11.7 Symbolic Vectors, Matrices, and Tensors

Let's inform *Mathematica* that u , v , and w are symbolic three-dimensional vectors with real components. The scalar product is $u.v$, and the vector product is $\text{Cross}[u, v]$. The function TensorReduce simplifies expressions with vectors.

```

In[63] := $Assumptions = Element[u|v|w, Vectors[3, Reals]]
Out[63] = (u|v|w) ∈ Vectors[3, Reals]
In[64] := TensorReduce[u.v - v.u]
Out[64] = 0
In[65] := TensorReduce[Cross[u, v] + Cross[v, u]]
Out[65] = 0
In[66] := TensorReduce[u.Cross[v, w] + v.Cross[w, u] + w.Cross[u, v]]
Out[66] =  $3u \times v.w$ 
In[67] := TensorReduce[Cross[u, Cross[v, w]]]
Out[67] =  $-wu.v + vu.w$ 
In[68] := TensorReduce[u.(2 * v + 3 * w)]
Out[68] =  $2u.v + 3u.w$ 

```

Now let's say that u and v are d -dimensional vectors, and S and A are $d \times d$ matrices, S symmetric and A antisymmetric.

```

In[69] := $Assumptions = {Element[u|v, Vectors[d, Reals]],
Element[S, Matrices[{d, d}, Reals, Symmetric[{1, 2}]]],
Element[A, Matrices[{d, d}, Reals, Antisymmetric[{1, 2}]]]};
In[70] := TensorReduce[v.A.(u + v)]
Out[70] =  $-u.A.v$ 
In[71] := TensorReduce[u.S.v + v.S.u]
Out[71] =  $2u.S.v$ 

```

The tensor product $S_{ij}A_{kl}$ is contracted in j and k and in i and l .

```

In[72] := TensorReduce[TensorContract[TensorProduct[S, A], {{2, 3}, {1, 4}}]]
Out[72] = 0

```

The Riemann curvature tensor has the properties $R_{ijkl} = -R_{jikl}$ and $R_{ijkl} = R_{klij}$.

```

In[73] := $Assumptions = Element[R, Arrays[{4, 4, 4, 4}, Reals,
  {{{2, 1, 3, 4}, -1}, {{3, 4, 1, 2}, 1}}]]
Out[73] = R ∈ Arrays[{4, 4, 4, 4}, Reals, {{Cycles[{{1, 2}}, -1],
  {Cycles[{{1, 3}, {2, 4}}, 1], {Cycles[{{3, 4}}, -1}}]}
In[74] := TensorReduce[TensorContract[R, {{1, 2}}]]
Out[74] = 0
The Ricci tensor.
In[75] := R2 = TensorContract[R, {{1, 3}}]
Out[75] = TensorContract[R, {{1, 3}}]
In[76] := {TensorRank[R2], TensorDimensions[R2], TensorSymmetry[R2]}
Out[76] = {2, {4, 4}, Symmetric[{{1, 2}}]}
 $R_{ijkl}R_{ijkl} + R_{ijkl}R_{klji} + R_{ijkl}R_{ikjl}$ .
In[77] := TensorReduce[
  TensorContract[TensorProduct[R, R], {{1, 5}, {2, 6}, {3, 7}, {4, 8}}] +
  TensorContract[TensorProduct[R, R], {{1, 7}, {2, 8}, {3, 6}, {4, 5}}] +
  TensorContract[TensorProduct[R, R], {{1, 5}, {2, 7}, {3, 6}, {4, 8}}]]
Out[77] = TensorContract[R ⊗ R, {{1, 5}, {2, 7}, {3, 6}, {4, 8}}]
Unfortunately, Mathematica cannot take  $R_{ijkl} + R_{iklj} + R_{iljk} = 0$  into account.
In[78] := $Assumptions = True;

```

Chapter 12

Input–Output and Strings

12.1 Reading and Writing .m Files

When developing any nontrivial *Mathematica* program, it is better to write it in a text file, using a text editor, and then to read it into a fresh *Mathematica* session. In this way, your actions will be reproducible. You can fix a bug and see what has changed. Suppose we have a text file called `wrong.m`. It contains the text

```
In[1] := FilePrint["wrong.m"]
```

```
a=x^2+2*x*y+y^2;
```

```
b=x^3+3*x^2*y
```

```
+3*x*y^2+y^3;
```

We can read it. Now the variables a and b have values:

```
In[2] := <<wrong.m
```

```
In[3] := {a,b}
```

```
Out[3] = {x^2 + 2xy + y^2, x^3 + 3x^2y}
```

The value of b is not what we expected. Why? When *Mathematica* sees an end of a line, it checks if the text read so far forms a syntactically correct expression. If so, the expression is considered complete; the next line starts a new expression. Otherwise, the next line is considered a continuation of the current expression. Thus our file `wrong.m` contains not two but three expressions: two assignments (to a and b) and a separate polynomial consisting of two terms. One way to prevent such unintended splitting of multiline expressions is to place a binary operator (e.g., $+$ or $-$) at the end of each line. *Mathematica* always writes its result in such a way. But it is easy to forget about it when writing a long expression in a text editor. Also, if we paste results from some other system into a *Mathematica* program, it would be tedious and error-prone to bring long expressions to such a form by hand. It is better to enclose each multiline expression in extra parentheses, then any incomplete subset of lines is not syntactically correct. Therefore we edit our `wrong.m` and obtain a file `right.m`:

In[4] := FilePrint["right.m"]

$a = x^2 + 2xy + y^2$;

$b = (x^3 + 3x^2y$

$+ 3xy^2 + y^3)$;

Now everything's all right:

In[5] := <<right.m

In[6] := {a,b}

Out[6] = $\{x^2 + 2xy + y^2, x^3 + 3x^2y + 3xy^2 + y^3\}$

In[7] := Clear[a,b]

Let us modify the file:

In[8] := FilePrint["right2.m"]

$a = x^2 + 2xy + y^2$;

$(x^3 + 3x^2y$

$+ 3xy^2 + y^3)$

Now the last expression is not an assignment, but just a polynomial. What happens if we read it?

In[9] := <<right2.m

Out[9] = $x^3 + 3x^2y + 3xy^2 + y^3$

In[10] := a

Out[10] = $x^2 + 2xy + y^2$

The operator << (its full name is Get) returns the value of the last expression.

Therefore, we can use it in an assignment (or as an argument of any other function):

In[11] := b = <<right2.m

Out[11] = $x^3 + 3x^2y + 3xy^2 + y^3$

In[12] := {a,b}

Out[12] = $\{x^2 + 2xy + y^2, x^3 + 3x^2y + 3xy^2 + y^3\}$

In[13] := Clear[b]

The function Get tries to find the file in all directories in the list \$Path. Its default value contains, in particular, the current directory “.”. You can add more directories to it using list operations, such as Append, Prepend, or Join.

And this operator (its full name is Put) writes the value of an expression into a file.

The value is written in the input form, and hence can be later read by *Mathematica*.

In[14] := a >> result.m

In[15] := FilePrint["result.m"]

$x^2 + 2xy + y^2$

In[16] := Clear[a]

In[17] := a = <<result.m

Out[17] = $x^2 + 2xy + y^2$

In[18] := a

Out[18] = $x^2 + 2xy + y^2$

The function >> writes just an expression, not an assignment to some variable.

Often it is more useful to write an assignment (or several of them) which defines some variable (or function). Suppose we have

In[19] := f[0] = 1; f[n_] := n * f[n - 1]

We can save the definition of the function f into a file:

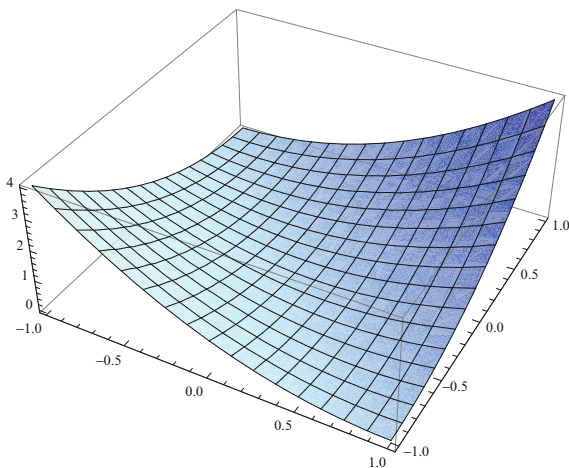
```
In[20] := Save["f.m", f]
In[21] := FilePrint["f.m"]
f[0] = 1
f[n_] := n * f[n - 1]
In[22] := Clear[f]
In[23] := <<"f.m"
In[24] := f[10]
Out[24] = 3628800
In[25] := Clear[f]
```

12.2 Output

The function Print prints expressions (including strings, plots, etc.). It does not separate them by spaces, so that it is usually a good idea to insert "" between expressions. It adds a newline at the end.

```
In[26] := s = "A strings\nwith a newline";
In[27] := p = Plot3D[a, {x, -1, 1}, {y, -1, 1}];
In[28] := Print[s, "", a, "", p]
Out[28] = A strings
```

with a newline $x^2 + 2xy + y^2$



```
In[29] := Clear[p]
```

The function Print is most useful when you want to know what happens at some point deep inside your own function when it is called. If you want to write expressions to a file instead, use Write:

```
In[30] := f = OpenWrite["res.m"]
Out[30] = OutputStream[res.m, 19]
In[31] := Write[f, s]; Write[f, a]
In[32] := Close[f]
Out[32] = res.m
In[33] := FilePrint["res.m"]
```

A strings\nwith a newline

$$x^2 + 2*x*y + y^2$$

Expressions are written in the input form and can be read later.

In[34] := Clear[s,a]

12.3 C, Fortran, and TeX Forms

Suppose you have derived analytically a valuable expression a :

In[35] := a = Sin[x]^2/x^2 - 1

$$\text{Out[35]} = -1 + \frac{\text{Sin}[x]^2}{x^2}$$

Now you want to do some large-scale numerical calculations with it. In order to avoid possible errors when translating this expression into a form suitable to inclusion into a numerical program, it is a good idea to do this step automatically. The function CForm converts an expression into a form which can be inserted into a C (or C++) program.

In[36] := CForm[a]

$$\text{Out[36]} = -1 + \text{Power}(\text{Sin}(x),2)/\text{Power}(x,2)$$

The macros Power, Sin, and friends are defined in the file mdefs.h which is supplied with *Mathematica*. If the expression contains special functions, you will need some extra C libraries which can calculate them numerically. Of course, it is better to write the C form of an expression into a file and then insert it into your program. For those old-fashioned enough to do numerical computations in Fortran, there is also FortranForm.

In[37] := FortranForm[a]

$$\text{Out[37]} = -1 + \text{Sin}(x)**2/x**2$$

At last, all computations are done, and you are writing an article to be submitted to a scientific journal. You want to include your valuable expression a and to avoid errors in process of doing so. Scientific articles are written in L^AT_EX (in most cases). The function TeXForm converts an expression into a form which can be inserted into a L^AT_EX file.

In[38] := TeXForm[a]

$$\text{Out[38]} = \frac{\{\text{sin}^2(x)\}}{\{x^2\}} - 1$$

In[39] := Clear[a]

12.4 Strings

The simplest string operation is concatenation:

In[40] := s1 = "This"; s2 = "is"; s3 = "a string";

In[41] := s = s1 <> " " <> s2 <> " " <> s3

Out[41] = This is a string

In[42] := FullForm[s]

Out[42]//FullForm =
"This is a string"

In[43] := StringLength[s]

Out[43] = 16

In[44] := Clear[s, s1, s2, s3]

Mathematica has string patterns and substitutions.

In[45] := StringReplace["abcabd", "ab" → "AB"]

Out[45] = ABcABd

$x_.$ means an arbitrary (single) character.

In[46] := StringReplace["a1b a2b a3c", a ~~ x_ ~~ b → A ~~ x ~~ B]

Out[46] = A1B A2B a3c

In[47] := StringReplace["a1b2 a12b",

a ~~ x_ ~~ b ~~ y_ → A ~~ y ~~ B ~~ x]

Out[47] = A2B1 a12b

Internally, these patterns are the function StringExpression.

In[48] := FullForm[a ~~ x_]

Out[48]//FullForm =

StringExpression[a, Pattern[x, Blank[]]]

A pattern with two identical arbitrary characters.

In[49] := StringReplace["a1b1 a1b2 a2b2",

a ~~ x_ ~~ b ~~ x_ → A ~~ x ~~ x ~~ B]

Out[49] = A11B a1b2 A22B

b|c means b or c.

In[50] := StringReplace["abcd abcd", b|c → X]

Out[50] = aXXd aXXd

$x_..$ means any nonempty sequence of characters.

In[51] := StringReplace["ab", a ~~ x_.. ~~ b ~~ x_.. → A ~~ x ~~ B ~~ x]

Out[51] = ab

In[52] := StringReplace["a12b12",

a ~~ x_.. ~~ b ~~ x_.. → A ~~ x ~~ B ~~ x]

Out[52] = A12B12

$x_...$ means any sequence of characters (including empty).

In[53] := StringReplace["ab", a ~~ x_... ~~ b ~~ x_... → A ~~ x ~~ B ~~ x]

Out[53] = AB

In[54] := StringReplace["a12b12",

a ~~ x_... ~~ b ~~ x_... → A ~~ x ~~ B ~~ x]

Out[54] = A12B12

As in the case of general patterns, /; means a condition (such that).

In[55] := StringReplace["a1b1 a1b2",

a ~~ x_ ~~ b ~~ y_ /; x! = y → A ~~ x ~~ y ~~ B]

Out[55] = a1b1 A12B

The function StringMatchQ tests if a string matches a pattern.

```
In[56] := {StringMatchQ["a1b1", a ~~ x_ ~~ b ~~ x_],  
StringMatchQ["a1b2", a ~~ x_ ~~ b ~~ x_],  
StringMatchQ["a1b1c", a ~~ x_ ~~ b ~~ x_]}
```

```
Out[56] = {True, False, False}
```

The function `StringFreeQ` tests if there are no substrings matching a pattern.

```
In[57] := {StringFreeQ["a1b1", a ~~ x_ ~~ b ~~ x_],  
StringFreeQ["a1b2", a ~~ x_ ~~ b ~~ x_],  
StringFreeQ["a1b1c", a ~~ x_ ~~ b ~~ x_]}
```

```
Out[57] = {False, True, False}
```

The function `StringSplit` splits a string at each occurrence of a pattern; if the second argument is not given, then at each white space.

```
In[58] := StringSplit["xxa1byya2bzz", a ~~ x_ ~~ b]
```

```
Out[58] = {xx, yy, zz}
```

```
In[59] := StringSplit["xx yy zz\nuu\tvv\t\t ww"]
```

```
Out[59] = {xx, yy, zz, uu, vv, ww}
```

Mathematica contains many more string manipulation functions and additional powerful features of string patterns and can be used for text processing instead of Perl. For more details, see the online help.

Chapter 13

Packages

13.1 Contexts

When writing a large program, it is easy to accidentally use one symbol for two different quantities in different parts of the program. This leads to difficult-to-find bugs. This is especially true if parts of the program are written by different persons (in particular, when some packages from the standard library, or third-party packages, are used). To avoid such problems, contexts are used.

In *Mathematica*, a full symbol name consists of two parts: the *context* and the *short name*. Two symbols in different contexts may have the same short name. For example, the global symbol x and the symbol x in the contexts a and b are unrelated.

In[1] := $x = 1; a\ x = 2; \{x, a\ x, b\ x\}$

Out[1] = $\{1, 2, b\ x\}$

Contexts may be nested. Here the variable x lives in the context b which lives in the context a (and thus is unrelated to the global context b used above).

In[2] := $a\ b\ x$

Out[2] = $a\ b\ x$

The current default context is held in the variable $\$Context$. It is used when a new symbol is created without specifying its context.

In[3] := $\$Context$

Out[3] = Global

When a symbol without an explicit context is used, it is being searched in contexts specified in $\$ContextPath$.

In[4] := $\$ContextPath$

Out[4] = $\{\text{PacletManager}, \text{QuantityUnits}, \text{WebServices}, \text{System}, \text{Global}\}$

Built-in symbols live in the context System .

In[5] := $\{\text{Context}[x], \text{Context}[a\ x], \text{Context}[b\ x], \text{Context}[a\ b\ x], \text{Context}[\text{Pi}]\}$

Out[5] = $\{\text{Global}, a, b, a\ b, \text{System}\}$

You can change $\$ContextPath$ using standard list functions. The function `Remove` removes symbols from the system.

In[6] := $\text{Clear}[x, a\ x]; \text{Remove}[a\ x, b\ x, a\ b\ x]$

13.2 Packages

Mathematica comes with a library of packages extending its built-in functionality. A package can be loaded by

```
In[7] := <<Quaternions`
```

Now this context is prepended to `$ContextPath`.

```
In[8] := $ContextPath
```

```
Out[8] = {Quaternions`, PacletManager`, QuantityUnits`, WebServices`,
          System`, Global` }
```

Short names will be searched in this context first, possibly shadowing variables and functions from `Global``.

```
In[9] := Context[Quaternion]
```

```
Out[9] = Quaternions`
```

The package `Quaternions` lives in the directory `Quaternions`, which lives in the standard library directory.

Many additional packages are available at <http://library.wolfram.com/infocenter/MathSource/>. You can download them and install somewhere in your `$Path`.

You can instruct *Mathematica* to load a package whenever any function defined in it is used.

```
In[10] := DeclarePackage["NumericalCalculus",
                        {"EulerSum", "NLimit", "ND", "NSeries", "NResidue"}]
```

```
Out[10] = NumericalCalculus`
```

```
In[11] := ND[x^2, x, 1.0]
```

```
Out[11] = 2.
```

13.3 Writing Your Own Package

Begin, End

`Begin["a"]` changes the default `$Context`; all symbols defined after this will live in this context.

```
In[12] := Begin["a"]
```

```
Out[12] = a`
```

```
In[13] := $Context
```

```
Out[13] = a`
```

```
In[14] := z = 0; Context[z]
```

```
Out[14] = a`
```

`End[]` restores the previous `$Context`.

```
In[15] := End[]
```

```
Out[15] = a`
```

In[16] := \$Context

Out[16] = Global`

In[17] := a`z

Out[17] = 0

BeginPackage, EndPackage

`BeginPackage["a`"]` sets `$Context` and changes `$ContextPath` in such a way that only the contexts `a`` and `System`` are available.

In[18] := BeginPackage["a`"]

Out[18] = a`

In[19] := \$Context

Out[19] = a`

In[20] := \$ContextPath

Out[20] = {a`, System`}

In[21] := u = 1;

`EndPackage[]` restores the previous `$Context`; `ContextPath` gets its old value prepended by `a``, so that symbols defined after `BeginPackage[a`]` remain available.

In[22] := EndPackage[]

In[23] := \$Context

Out[23] = Global`

In[24] := \$ContextPath

Out[24] = {a`, NumericalCalculus`, Quaternions`, PacletManager`,
QuantityUnits`, WebServices`, System`, Global`}

In[25] := u

Out[25] = 1

In[26] := Clear[z, u]

A Typical Package

A simple package looks like this.

In[27] := FilePrint["APackage.m"]

`BeginPackage["APackage"]`

`f::usage = "f squares its argument"`

`Begin["Private"]`

`g[x_] := x^2`

`f[x_] := Expand[g[x]]`

`End[]`

`EndPackage[]`

After `BeginPackage["APackage"]`, publicly available functions and variables of the package are introduced, usually by assignments to their usage messages.

Implementation of the package is done in the context `APackage`Private``, which may contain additional functions and variables (not seen by users of the package).

In[28] := `<<APackage``

In[29] := `?f`

Out[29] = f squares its argument

In[30] := `f[a + b]`

Out[30] = $a^2 + 2ab + b^2$

Part II

Computer Classes

Before spending your time and effort on catching a lion, check: maybe, somebody has already caught it, and it is available for download at <http://library.wolfram.com/infocenter/MathSource/>

Chapter 14

Plots

After typing a *Mathematica* command in its notebook interface, you can send it to the kernel (which performs calculations) by pressing Shift-Enter. The result appears in the output cell which follows your input cell. Both are nested in an outer cell representing a calculation step. Later you can return to this input cell, edit the command, and execute it again. The old output will be replaced by the new result. It is allowed to type several commands in a single input cell, but this is not convenient—don't do so unless you have good reasons.

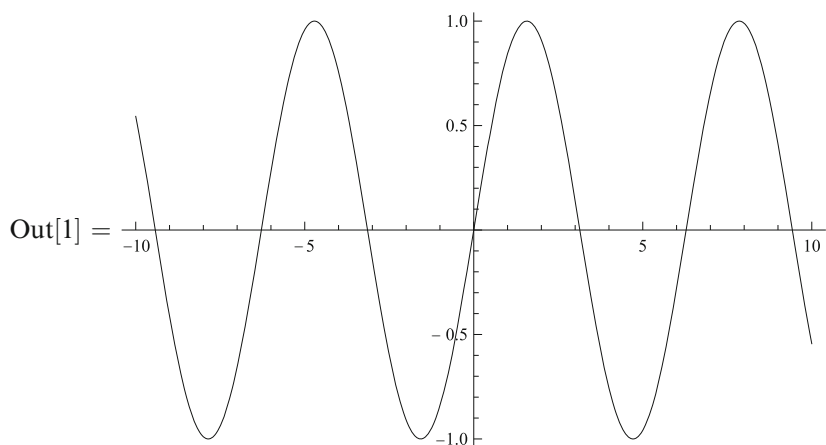
Mathematica Help contains all the necessary information. The Help menu contains Documentation Center, Function Navigator, and Virtual Book (among other things). You can quickly get help for a specific function if you select it with the mouse and press F1.

14.1 2D Plots

Function Plot

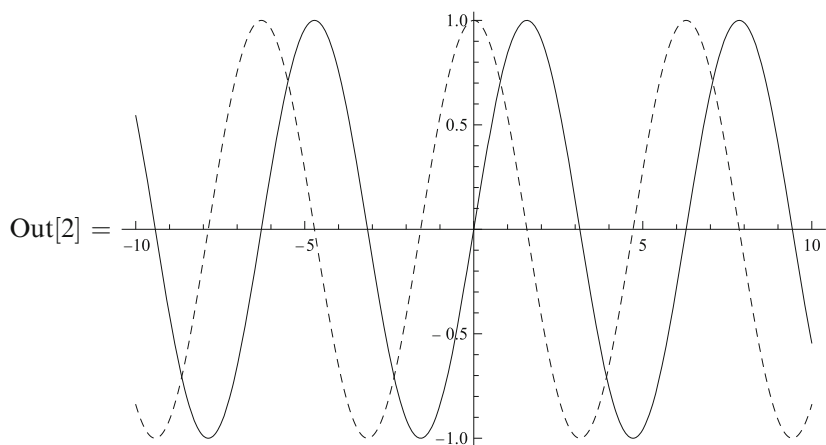
See Help → Virtual Book → Visualization and Graphics → Graphics and Sound → Basic Plotting, and Help → Function Navigator → Visualization and Graphics → Function Visualization → Plot for more details.

In[1] := Plot[Sin[x], {x, -10, 10}]



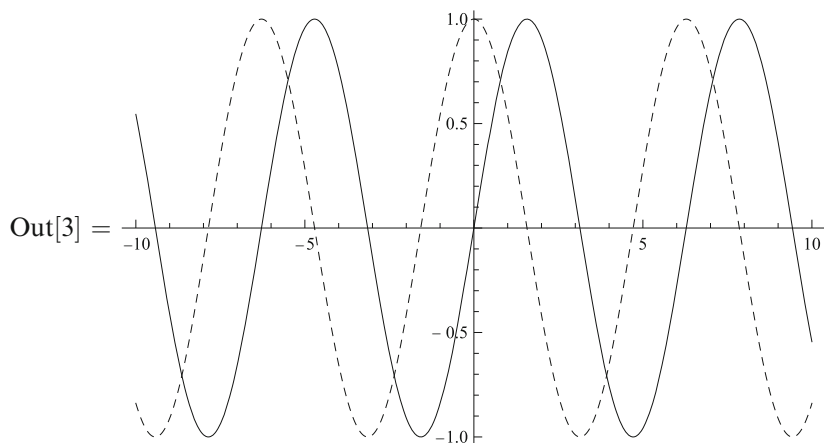
Several Functions

In[2] := Plot[{Sin[x], Cos[x]}, {x, -10, 10}]



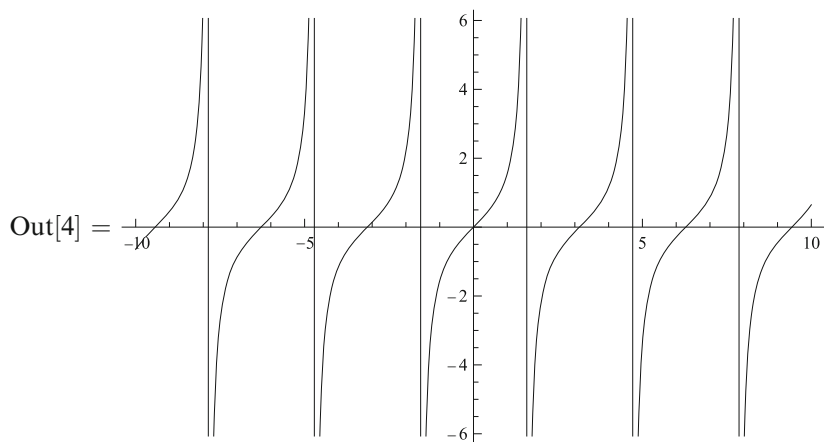
You can set colors and styles of the curves (Virtual Book → Visualization and Graphics → Graphics and Sound → Options for Graphics; Function Navigator → Visualization and Graphics → Options and Styling → Plotting Options → Plot-Style).

```
In[3] := Plot[{Sin[x], Cos[x]}, {x, -10, 10}, PlotStyle->{Red, {Blue, Dashed}}]
```



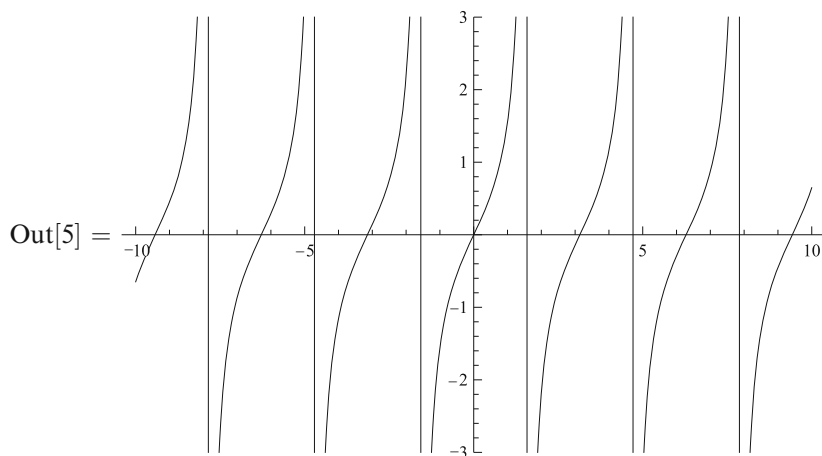
Unbounded Function

```
In[4] := Plot[Tan[x], {x, -10, 10}]
```



Mathematica has chosen some y scale. How to set it? Find in the Help.

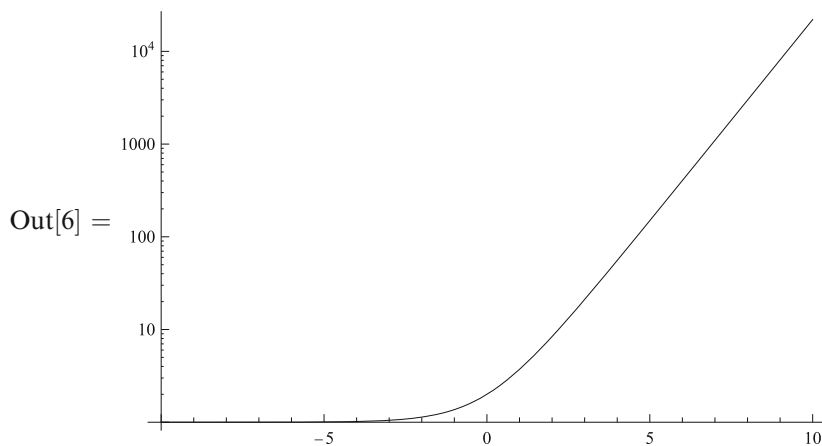
In[5] := Plot[Tan[x], {x, -10, 10}, PlotRange->{-3, 3}]



Logarithmic Scale

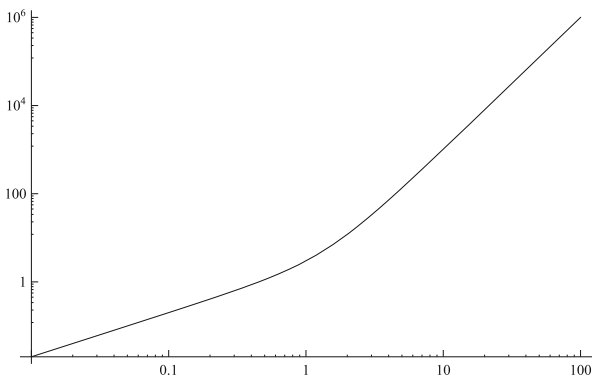
If our function is positive and varies by orders of magnitude in our region, it is convenient to use logarithmic scale in y . If the independent variable also varies by orders of magnitude, the x -axis scale also should be logarithmic (Function Navigator → Visualization and Graphics → Function Visualization → LogPlot, LogLogPlot).

In[6] := LogPlot[Exp[x] + 1, {x, -10, 10}]



In[7] := `LogLogPlot[x^3 + 2 * x, {x, 10^-2, 10^2}]`

Out[7] =

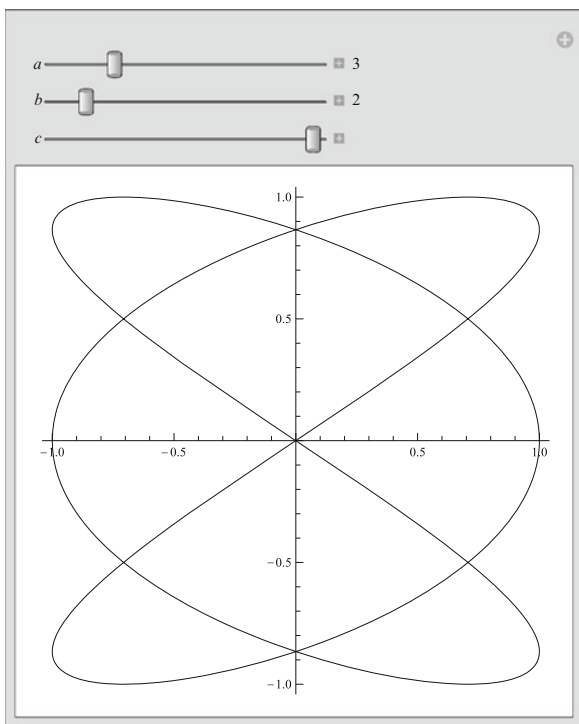


Parametric Curve

Lissajous figures.

In[8] := `Manipulate[ParametricPlot[{Sin[a * t + c], Sin[b * t]}, {t, 0, 2 * Pi}],
{a, 1, 10, 1, Appearance -> "Labeled"},
{b, 1, 10, 1, Appearance -> "Labeled"}, {{c, Pi/2}, 0, Pi/2}]`

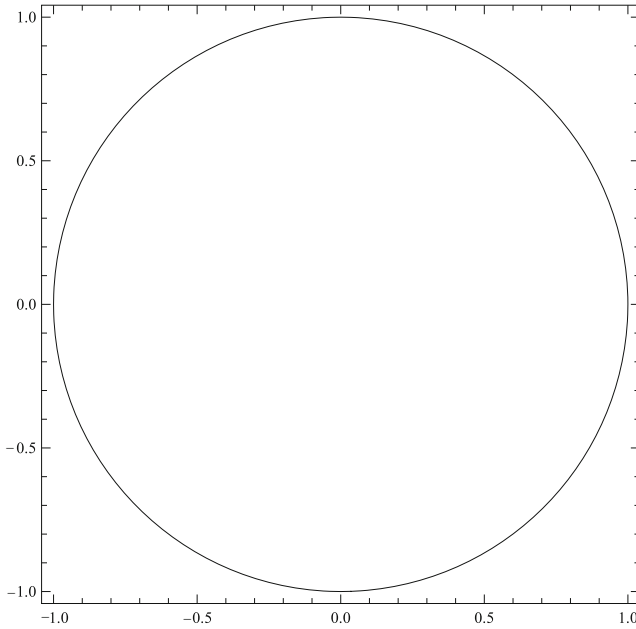
Out[8] =



Implicit Plots

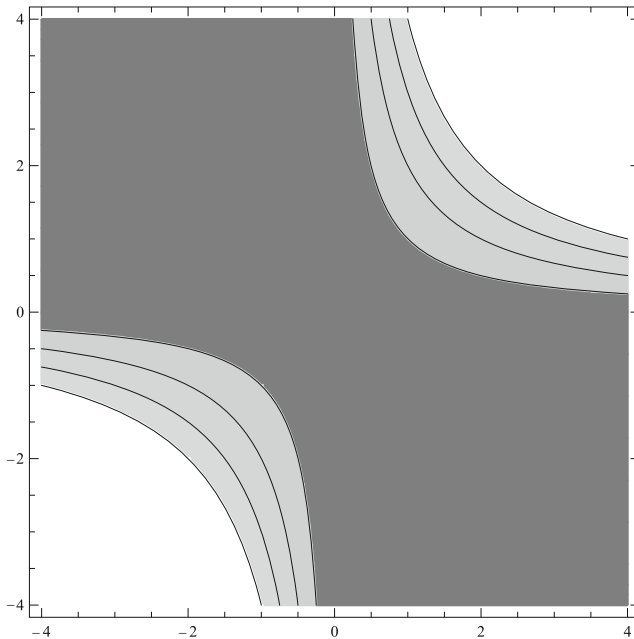
In[9] := ContourPlot[x^2 + y^2 == 1, {x, -1, 1}, {y, -1, 1}]

Out[9] =



In[10] := ContourPlot[x * y, {x, -4, 4}, {y, -4, 4}, Contours -> {1, 2, 3, 4}]

Out[10] =



Experimental Points

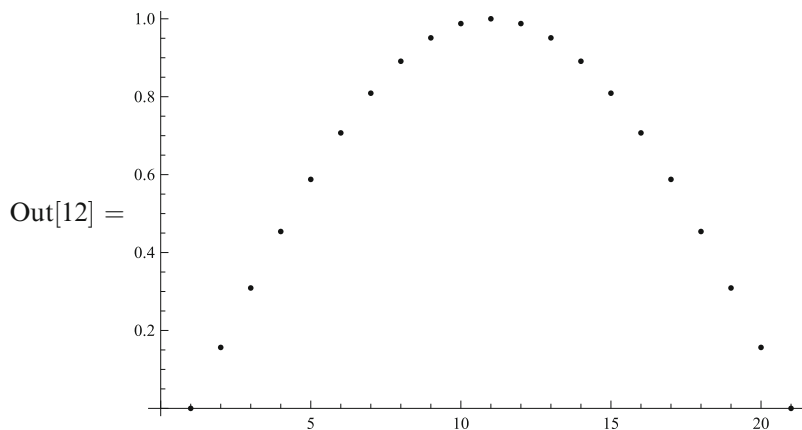
In real life they are being read from a file. We don't have such a file at hand, and therefore we'll generate a list of "experimental" points according to some formulas.

In[11] := l = Table[Sin[Pi * n/20], {n, 0, 20}]

Out[11] = {0., 0.156434, 0.309017, 0.45399, 0.587785, 0.707107, 0.809017, 0.891007, 0.951057, 0.987688, 1., 0.987688, 0.951057, 0.891007, 0.809017, 0.707107, 0.587785, 0.45399, 0.309017, 0.156434, 0.}

Function Navigator → Visualization and Graphics → Data Visualization → List-Plot.

In[12] := p1 = ListPlot[l]

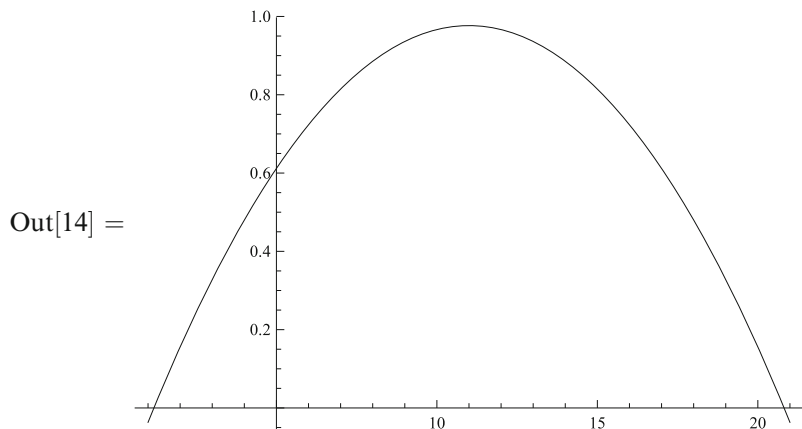


Let's try to fit these points by a quadratic polynomial.

In[13] := f = Fit[l, {1, x, x^2}, x]

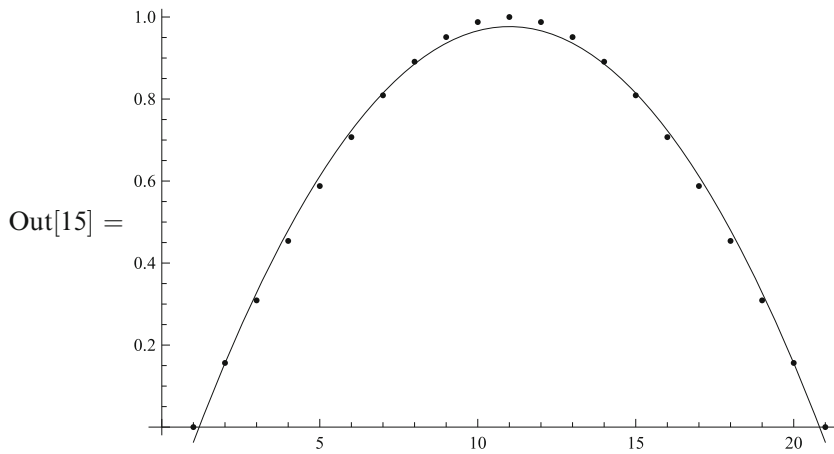
Out[13] = $-0.24953 + 0.222936x - 0.0101334x^2$

In[14] := p2 = Plot[f, {x, 1, 21}]



And now the curve and the points on a single plot (Function Navigator → Visualization and Graphics → Data Visualization → Annotation & Combination → Show).

In[15] := Show[p1,p2]

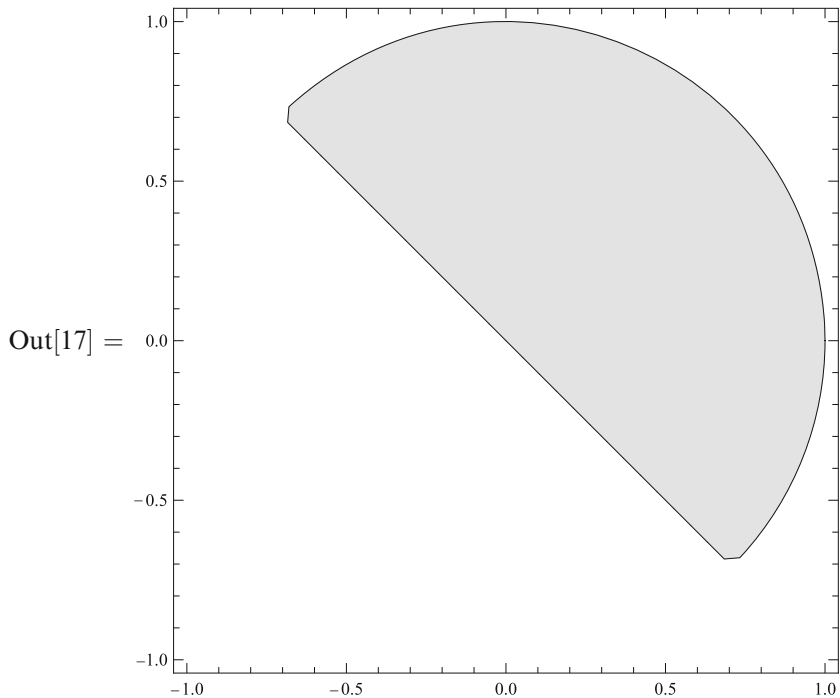


In[16] := Clear[l,f,p1,p2]

Inequalities

Function Navigator → Visualization and Graphics → Function Visualization → RegionPlot.

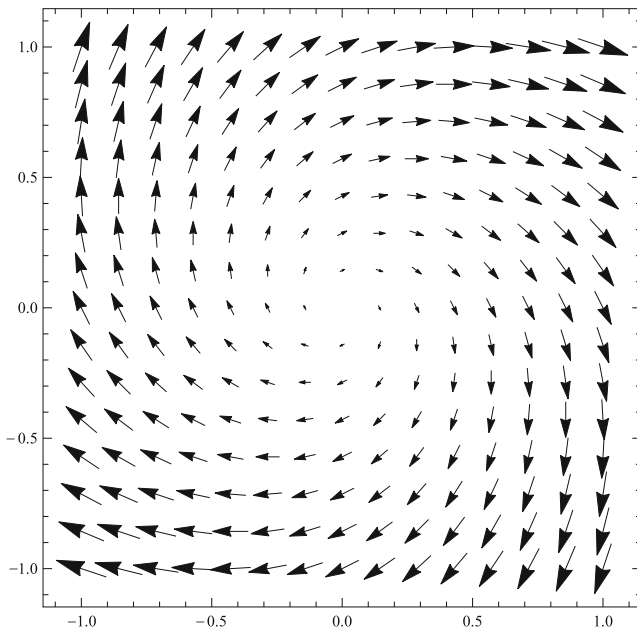
In[17] := RegionPlot[x^2 + y^2 < 1 && x + y > 0, {x, -1, 1}, {y, -1, 1}]



Vector Fields

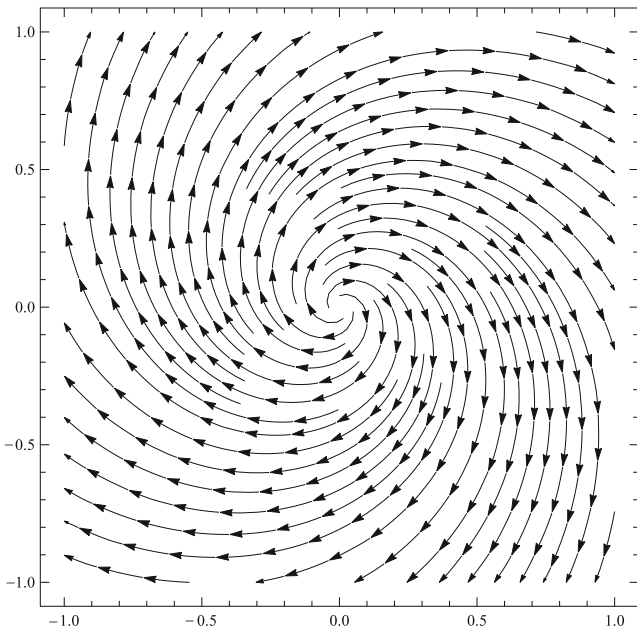
```
In[18] := VectorPlot[{y + 0.5 * x, -x + 0.5 * y}, {x, -1, 1}, {y, -1, 1}]
```

Out[18] =



```
In[19] := StreamPlot[{y + 0.5 * x, -x + 0.5 * y}, {x, -1, 1}, {y, -1, 1}]
```

Out[19] =



14.2 3D Plots

Hat

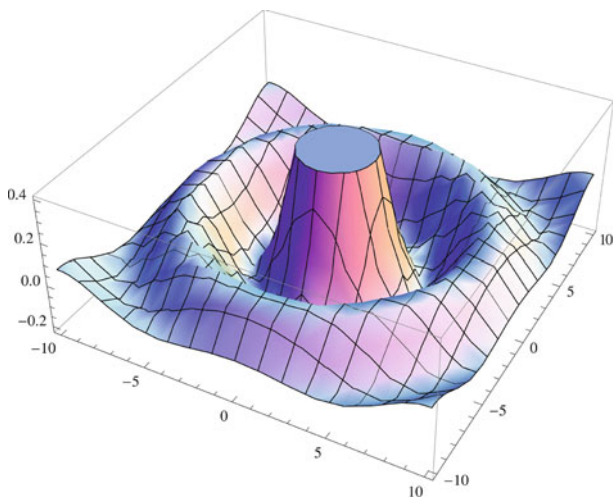
Let's draw a hat with a wavy pent. First define a function.

```
In[20] := f[x_,y_] := With[{r = Sqrt[x^2 + y^2]}, Sin[r]/r]
```

Virtual Book → Visualization and Graphics → Three-Dimensional Surface Plots;
Function Navigator → Visualization and Graphics → Function Visualization → Plot3D.

```
In[21] := Plot3D[f[x,y], {x, -10, 10}, {y, -10, 10}]
```

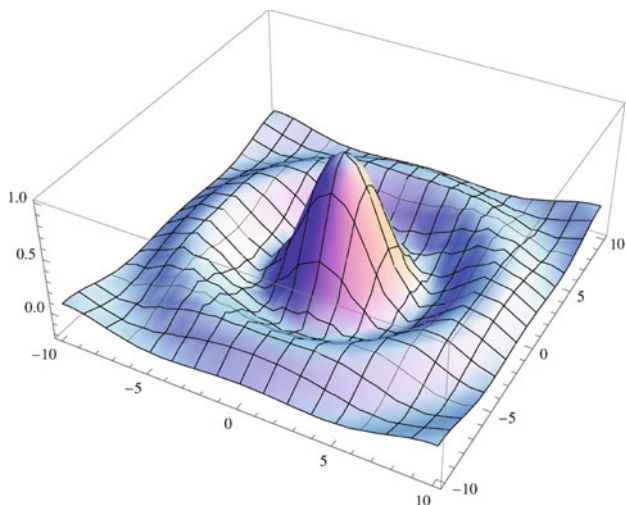
Out[21] =



And why is the top of the hat cut off?

```
In[22] := Plot3D[f[x,y], {x, -10, 10}, {y, -10, 10}, PlotRange->All]
```

Out[22] =



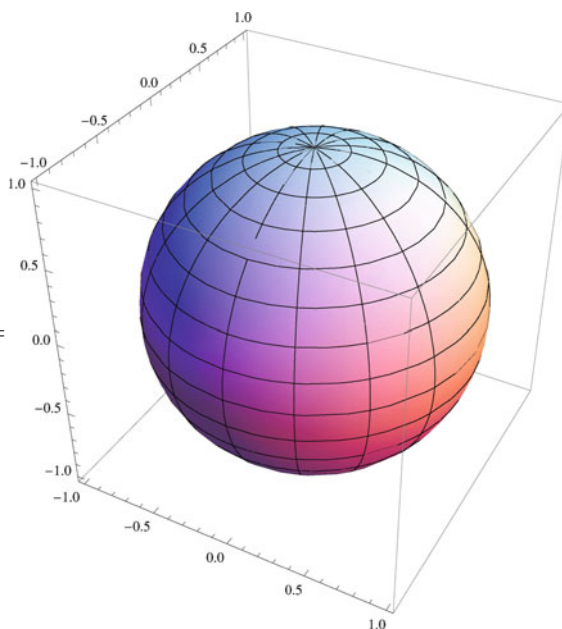
All 3D plots can be rotated by the mouse. If you press Shift, the mouse will move the plot; and if you press Ctrl, then it will resize it.

In[23] := Clear[f]

Sphere

**In[24] := ParametricPlot3D[{Sin[θ] * Cos[ϕ], Sin[θ] * Sin[ϕ], Cos[θ]},
{ θ , 0, Pi}, { ϕ , 0, 2 * Pi}]**

Out[24] =



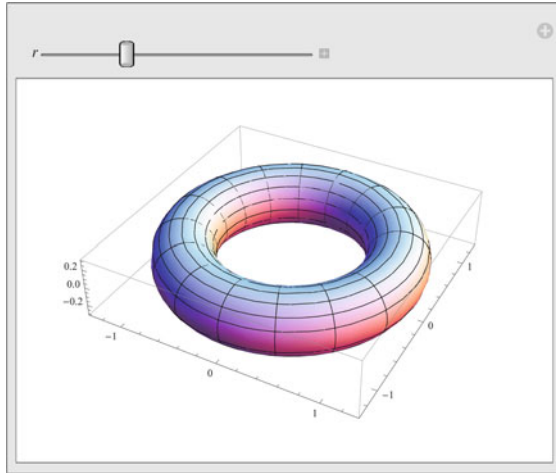
Donut

Take a point on the x -axis at a distance R from the origin; draw a circle of a radius r around it in the x, z plane; and rotate it around the z axis. You will get a donut (torus). Let R be 1; r can be tuned by the mouse from 0 to 1, with the initial value 0.3.

In[25] := R = 1;

**In[26] := Manipulate[
ParametricPlot3D[
 {(R + r * Cos[θ]) * Cos[ϕ], (R + r * Cos[θ]) * Sin[ϕ], r * Sin[θ]},
 { θ , 0, 2 * Pi}, { ϕ , 0, 2 * Pi}],
 {{r, 0.3}, 0, 1}]**

Out[26] =



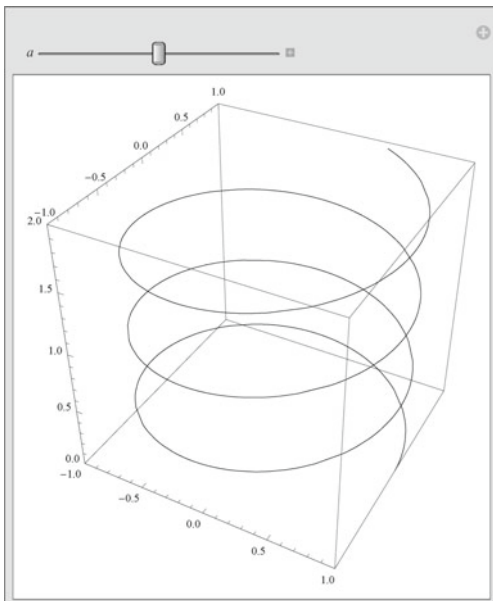
In[27] := Clear[R]

Spiral

ParametricPlot3D can draw curves, too.

```
In[28] := Manipulate[ParametricPlot3D[{Cos[t], Sin[t], a*t}, {t, 0, 20},
  PlotRange -> {{-1, 1}, {-1, 1}, {0, 2}},
  {{a, 0.1}, 0, 0.2}]
```

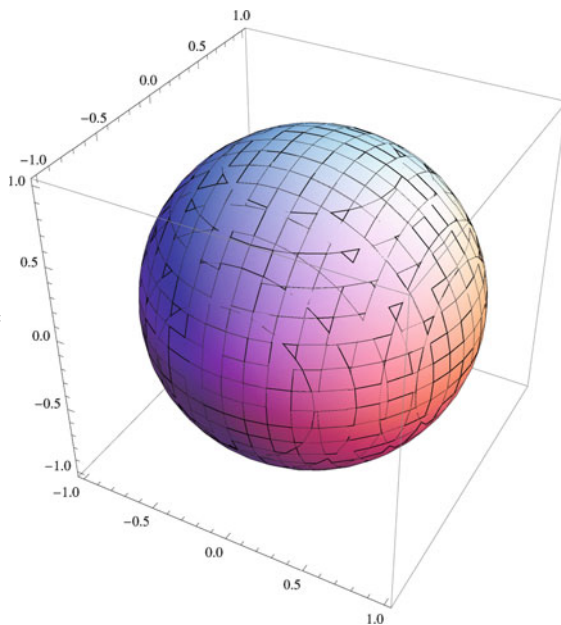
Out[28] =



Implicit Surface

```
In[29] := ContourPlot3D[x^2 + y^2 + z^2 == 1, {x, -1, 1}, {y, -1, 1}, {z, -1, 1}]
```

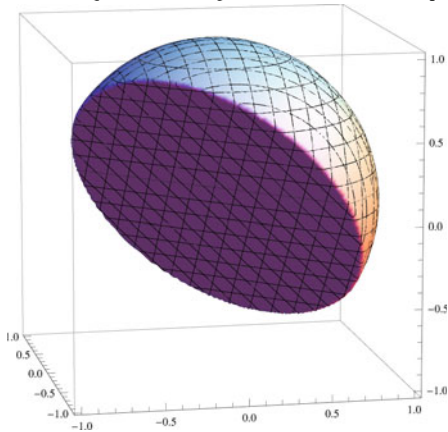
Out[29] =



Inequalities

```
In[30] := RegionPlot3D[x^2 + y^2 + z^2 < 1 && x + y + z > 0, {x, -1, 1}, {y, -1, 1}, {z, -1, 1}, ViewPoint -> {-2, -10, 2}, PlotPoints -> 100]
```

Out[30] =



Chapter 15

Trigonometric Functions

Statement of the Problem

Mathematica can calculate Cos and Sin for many arguments equal to π times rational numbers. For example,

In[1] := Table[Cos[Pi/n], {n, 1, 12}]

$$\text{Out[1]} = \left\{ -1, 0, \frac{1}{2}, \frac{1}{\sqrt{2}}, \frac{1}{4} (1 + \sqrt{5}), \frac{\sqrt{3}}{2}, \text{Cos} \left[\frac{\pi}{7} \right], \text{Cos} \left[\frac{\pi}{8} \right], \text{Cos} \left[\frac{\pi}{9} \right], \right. \\ \left. \sqrt{\frac{5}{8} + \frac{\sqrt{5}}{8}}, \text{Cos} \left[\frac{\pi}{11} \right], \frac{1 + \sqrt{3}}{2\sqrt{2}} \right\}$$

But it does not apply the half-angle formulas in all possible cases. We'll write our own cos function which does this; then

In[2] := sin[x_] := cos[x - Pi/2]

Simple Cases

These cases should be considered separately because *Mathematica* does not treat them as rational numbers times π .

In[3] := cos[0] = 1; cos[Pi] = -1;

In[4] := cos[n_Integer * Pi] := (-1)^n

General Case

**In[5] := cos[r_Rational * Pi] := Which[r < 0, cos[-r * Pi],
r > 2, cos[FractionalPart[r/2] * 2 * Pi], r > 1, cos[(2 - r) * Pi],
r > 1/2, -cos[(1 - r) * Pi],**

**EvenQ[Denominator[r]], Simplify[Sqrt[(1 + cos[2 * r * Pi])/2]],
True, Cos[r * Pi]]**

For example,

In[6] := {cos[Pi/32], cos[-65 * Pi/32], cos[3 * Pi/32], cos[33 * Pi/32]}

$$\text{Out[6]} = \left\{ \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}}, \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}}, \right. \\ \left. \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 - \sqrt{2}}}}, -\frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2}}}} \right\}$$

In[7] := {cos[Pi/48], cos[5 * Pi/48], cos[7 * Pi/48]}

$$\text{Out[7]} = \left\{ \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{3}}}}, \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{2 - \sqrt{3}}}}, \right. \\ \left. \frac{1}{2} \sqrt{2 + \sqrt{2 - \sqrt{2 - \sqrt{3}}}} \right\}$$

In[8] := {cos[Pi/40], cos[3 * Pi/40]}

$$\text{Out[8]} = \left\{ \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{\frac{1}{2} (5 + \sqrt{5})}}}, \frac{1}{2} \sqrt{2 + \sqrt{2 + \sqrt{\frac{1}{2} (5 - \sqrt{5})}}} \right\}$$

Check

**In[9] := check[d_, n_] := Module[{e = 0, x, ex},
Do[x = i/d * Pi; ex = Abs[N[cos[x] - Cos[x]]]; If[ex > e, e = ex,
{i, -d * n, d * n}];
e]**

In[10] := {check[128, 5], check[192, 5], check[320, 5]}

$$\text{Out[10]} = \left\{ 2.220446049250313 \times 10^{-16}, 6.106226635438361 \times 10^{-16}, \right. \\ \left. 1.861358289723114 \times 10^{-15} \right\}$$

Chapter 16

Quantum Oscillator

Catching a lion, the Schrödinger's method: At any moment, there is a nonzero probability that a lion is inside the cage. Sit and wait.

16.1 Lowering and Raising Operators

The Hamiltonian of the harmonic oscillator is [18]

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{m\omega^2 \hat{x}^2}{2}.$$

There is a dimensionful constant \hbar in quantum mechanics; therefore, two quantities m and ω define a scale of length $\sqrt{\hbar/(m\omega)}$, momentum $\sqrt{\hbar m\omega}$, energy $\hbar\omega$, and any other quantity of any dimensionality. They have the meaning of the characteristic amplitude, momentum, and energy of zero oscillations. We shall put $\hbar = 1$, $m = 1$, and $\omega = 1$, thus choosing these characteristic scales as units for measurement of corresponding quantities. Then

$$\hat{H} = \frac{\hat{p}^2 + \hat{x}^2}{2}.$$

Let's introduce the operator

$$\hat{a} = \frac{\hat{x} + i\hat{p}}{\sqrt{2}}$$

and its Hermitian conjugate

$$\hat{a}^+ = \frac{\hat{x} - i\hat{p}}{\sqrt{2}}.$$

The commutation relation $[\hat{p}, \hat{x}] = -i$ implies for them

$$[\hat{a}, \hat{a}^+] = 1.$$

The Hamiltonian is expressed via these operators as

$$\hat{H} = \hat{a}^+ \hat{a} + \frac{1}{2},$$

from where we obtain $[\hat{H}, \hat{a}] = -\hat{a}$, $[\hat{H}, \hat{a}^+] = \hat{a}^+$. Therefore, if $|\psi\rangle$ is an eigenstate of \hat{H} with the energy E : $\hat{H}|\psi\rangle = E|\psi\rangle$, then $\hat{a}|\psi\rangle$ and $\hat{a}^+|\psi\rangle$ are also eigenstates of \hat{H} with the energies $E - 1$ and $E + 1$:

$$\hat{H}\hat{a}|\psi\rangle = (\hat{a}\hat{H} - \hat{a})|\psi\rangle = (E - 1)\hat{a}|\psi\rangle, \quad \hat{H}\hat{a}^+|\psi\rangle = (E + 1)\hat{a}^+|\psi\rangle$$

(if only these states don't vanish). Hence the eigenvalues of \hat{H} form an arithmetic progression with step equal to 1. It is bounded from below because \hat{H} is a positive definite operator. Therefore, there exists a state $|0\rangle$ with the lowest energy that cannot be lowered any more:

$$\hat{a}|0\rangle = 0.$$

Its energy is equal to $\frac{1}{2}$:

$$\hat{H}|0\rangle = \left(\hat{a}^+ \hat{a} + \frac{1}{2}\right)|0\rangle = \frac{1}{2}|0\rangle$$

(this is the zero oscillations energy). Acting on $|0\rangle$ by the raising operator \hat{a}^+ n times, we obtain a state $|n\rangle$ with the energy

$$E_n = n + \frac{1}{2}.$$

Hence, $\hat{H}|n\rangle = (\hat{a}^+ \hat{a} + \frac{1}{2})|n\rangle = (n + \frac{1}{2})|n\rangle$ or

$$\hat{a}^+ \hat{a}|n\rangle = n|n\rangle,$$

i.e., $\hat{a}^+ \hat{a}$ acts as an operator of the level number.

We have $\hat{a}|n\rangle = c_n|n-1\rangle$; it is possible to make c_n real and positive by tuning the phases of the states $|n\rangle$. These coefficients can be found from the normalization condition: $|c_n|^2 = \langle n|\hat{a}^+ \hat{a}|n\rangle = n$. The action of the operator \hat{a}^+ follows from Hermitian conjugation:

$$\hat{a}|n\rangle = \sqrt{n}|n-1\rangle, \quad \hat{a}^+|n\rangle = \sqrt{n+1}|n+1\rangle.$$

From this we again have $\hat{a}^+ \hat{a}|n\rangle = n|n\rangle$.

In the coordinate representation

$$\hat{x} = x, \quad \hat{p} = -i \frac{d}{dx}.$$

Let's implement the operators \hat{a} and \hat{a}^+ in *Mathematica*.

In[1] := a[f_.] := Together[(x * f + D[f, x])/Sqrt[2]]

In[2] := ac[f_.] := Together[(x * f - D[f, x])/Sqrt[2]]

16.2 Ground State

This is the state which cannot be lowered by \hat{a} .

In[3] := Eq = a[ψ₀[x]] == 0

Out[3] = $\frac{x\psi_0[x] + \psi_0'[x]}{\sqrt{2}} == 0$

In[4] := s = DSolve[Eq, ψ₀[x], x]

Out[4] = $\left\{ \left\{ \psi_0[x] \rightarrow e^{-\frac{x^2}{2}} C[1] \right\} \right\}$

In[5] := ψ₀ = ψ₀[x]/.s[[1]]

Out[5] = $e^{-\frac{x^2}{2}} C[1]$

In[6] := Clear[Eq]

The normalization integral.

In[7] := NI = Integrate[ψ₀², {x, -Infinity, Infinity}]

Out[7] = $\sqrt{\pi} C[1]^2$

In[8] := s = Solve[NI == 1, C[1]]

Out[8] = $\left\{ \left\{ C[1] \rightarrow -\frac{1}{\pi^{1/4}} \right\}, \left\{ C[1] \rightarrow \frac{1}{\pi^{1/4}} \right\} \right\}$

In[9] := ψ₀ = ψ₀/.s[[2]]

Out[9] = $\frac{e^{-\frac{x^2}{2}}}{\pi^{1/4}}$

In[10] := Clear[NI, s]

16.3 Excited States

In[11] := ψ[0] = ψ₀;

In[12] := ψ[n_] := ψ[n] = ac[ψ[n - 1]]/Sqrt[n]

The wave functions of a few first states.

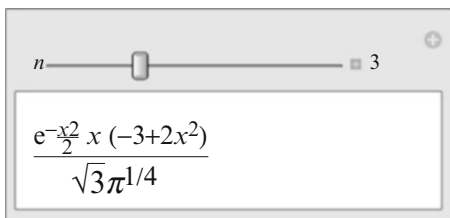
In[13] := Table[ψ[n], {n, 0, 10}]

Out[13] = $\left\{ \frac{e^{-\frac{x^2}{2}}}{\pi^{1/4}}, \frac{\sqrt{2}e^{-\frac{x^2}{2}}x}{\pi^{1/4}}, \frac{e^{-\frac{x^2}{2}}(-1+2x^2)}{\sqrt{2}\pi^{1/4}}, \frac{e^{-\frac{x^2}{2}}x(-3+2x^2)}{\sqrt{3}\pi^{1/4}}, \right.$
 $\frac{e^{-\frac{x^2}{2}}(3-12x^2+4x^4)}{2\sqrt{6}\pi^{1/4}}, \frac{e^{-\frac{x^2}{2}}x(15-20x^2+4x^4)}{2\sqrt{15}\pi^{1/4}},$
 $\frac{e^{-\frac{x^2}{2}}(-15+90x^2-60x^4+8x^6)}{12\sqrt{5}\pi^{1/4}}, \frac{e^{-\frac{x^2}{2}}x(-105+210x^2-84x^4+8x^6)}{6\sqrt{70}\pi^{1/4}},$
 $\left. \frac{e^{-\frac{x^2}{2}}(105-840x^2+840x^4-224x^6+16x^8)}{24\sqrt{70}\pi^{1/4}} \right\}$

$$\left. \begin{aligned} & \frac{e^{-\frac{x^2}{2}} x (945 - 2520x^2 + 1512x^4 - 288x^6 + 16x^8)}{72\sqrt{35}\pi^{1/4}}, \\ & \frac{e^{-\frac{x^2}{2}} (-945 + 9450x^2 - 12600x^4 + 5040x^6 - 720x^8 + 32x^{10})}{720\sqrt{7}\pi^{1/4}} \end{aligned} \right\}$$

And here the level number can be set by the mouse.

In[14] := Manipulate[$\psi[n]$, { n , 0, 10, 1, Appearance -> "Labeled"}]

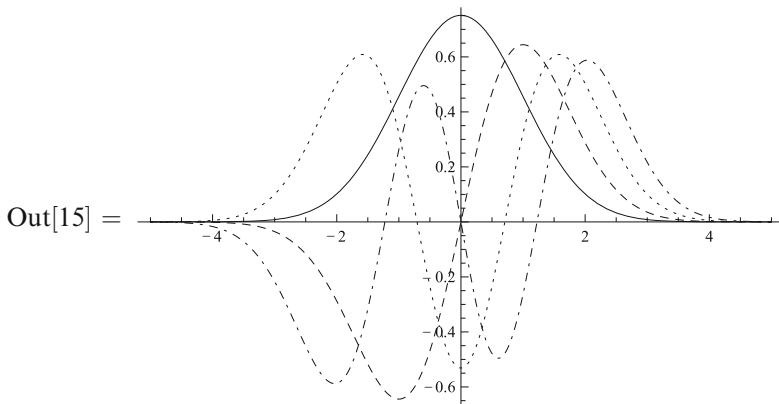


Out[14] =

$$\frac{e^{-\frac{x^2}{2}} x (-3 + 2x^2)}{\sqrt{3}\pi^{1/4}}$$

The wave functions of a few first states.

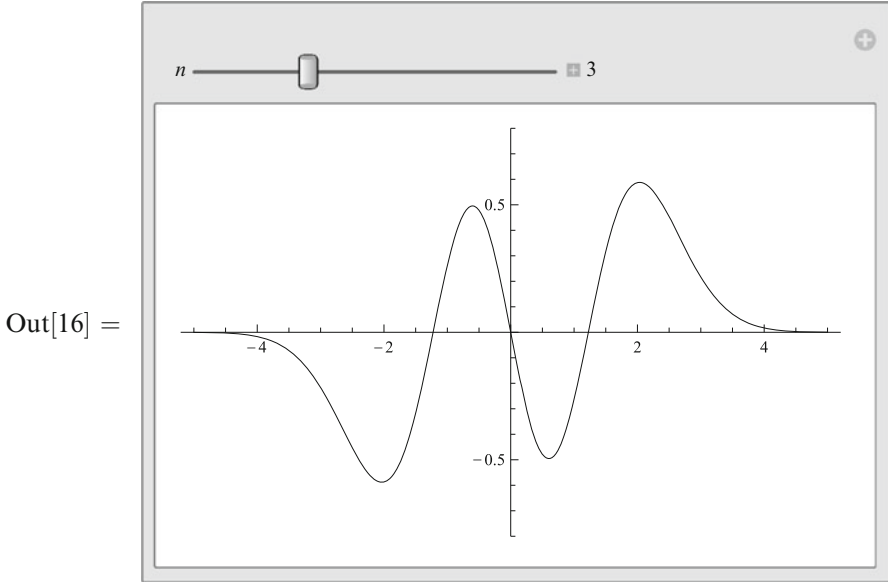
In[15] := Plot[Evaluate[Table[$\psi[n]$, { n , 0, 3}], { x , -5, 5}]



Out[15] =

And this is a live plot: the level number can be set by the mouse.

In[16] := Manipulate[Plot[$\psi[n]$, { x , -5, 5}, PlotRange -> {-0.8, 0.8}], { n , 0, 10, 1, Appearance -> "Labeled"}]



16.4 Some Properties

Orthogonality and normalization.

```
In[17] := Distribute[ψ]
```

Out[17] = ψ

```
In[18] := Parallelize[Table[Table[Integrate[ψ[n] * ψ[m], {x, -Infinity, Infinity}],
    {n, 0, 10}], {m, 0, 10}]]
```

```
Out[18] = {{1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0}, {0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1}}
```

Wave functions in the momentum representation (Fourier transforms) are the same as in the coordinate one, up to phase factors.

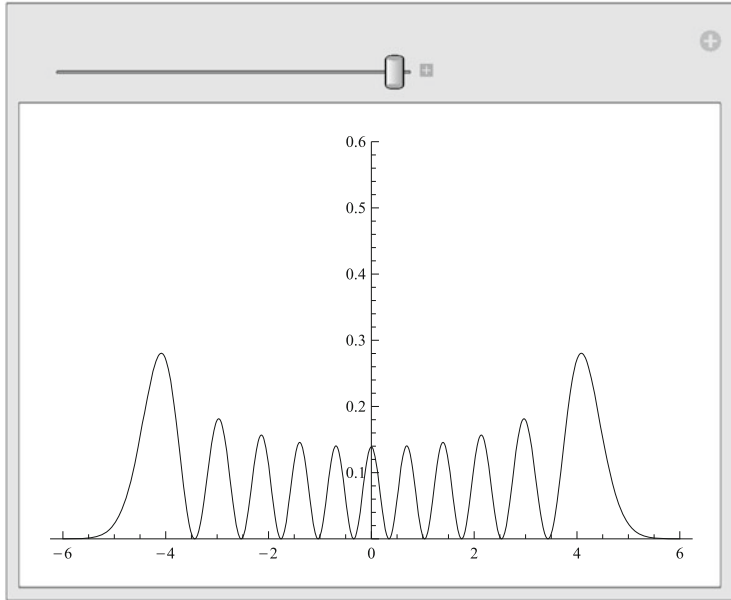
```
In[19] := Parallelize[Table[
    Cancel[Integrate[ψ[n] * Exp[-I * p * x], {x, -Infinity, Infinity}]] /
    Sqrt[2 * Pi] / (ψ[n] /. x -> p),
    {n, 0, 10}]]
```

```
Out[19] = {1, -i, -1, i, 1, -i, -1, i, 1, -i, -1}
```

The probability density.

```
In[20] := Manipulate[Plot[ψ[n]^2, {x, -6, 6}, PlotRange -> {0, 0.6}],
    {{n, 10}, 0, 10, 1, Appearance -> "Labeled"}]
```

Out[20] =



Why is it larger near the boundaries?

Chapter 17

Spherical Harmonics

17.1 Angular Momentum in Quantum Mechanics

The angular momentum operator $\hat{\mathbf{J}}$ is defined [18] in such a way that $\hat{U} = \exp(-i\hat{\mathbf{J}} \cdot \delta\boldsymbol{\varphi})$ is the operator of an infinitesimal rotation with the angle $\delta\boldsymbol{\varphi}$: if $|\psi\rangle$ is a state, then $\hat{U}|\psi\rangle$ is the same state rotated by $\delta\boldsymbol{\varphi}$. Therefore, the average value \mathbf{V}' of a vector operator $\hat{\mathbf{V}}$ over $\hat{U}|\psi\rangle$ is related to its average value \mathbf{V} over $|\psi\rangle$ by the formula $\mathbf{V}' = \mathbf{V} + \delta\boldsymbol{\varphi} \times \mathbf{V}$ and hence $\hat{U}^+\hat{\mathbf{V}}\hat{U} = \hat{\mathbf{V}} + i[\hat{\mathbf{J}} \cdot \delta\boldsymbol{\varphi}, \hat{\mathbf{V}}] = \hat{\mathbf{V}} + \delta\boldsymbol{\varphi} \times \hat{\mathbf{V}}$. Therefore, for any vector operator $\hat{\mathbf{V}}$ the commutation relation $[\hat{V}_i, \hat{J}_j] = i\epsilon_{ijk}\hat{V}_k$ holds. The average value of a scalar operator \hat{S} does not change at rotations; hence $[\hat{S}, \hat{J}_i] = 0$. In particular, the angular momentum $\hat{\mathbf{J}}$ is a vector operator, and its square $\hat{\mathbf{J}}^2 = \hat{J}_x^2 + \hat{J}_y^2 + \hat{J}_z^2$ is a scalar one:

$$[\hat{J}_i, \hat{J}_j] = i\epsilon_{ijk}\hat{J}_k, \quad [\hat{\mathbf{J}}^2, \hat{J}_i] = 0.$$

Therefore, a system of common eigenstates of $\hat{\mathbf{J}}^2$ and \hat{J}_z exists. Let's introduce the operators $\hat{J}_{\pm} = \hat{J}_x \pm i\hat{J}_y$, $\hat{J}_{\pm}^{\pm} = \hat{J}_{\mp}$; we have $[\hat{J}_z, \hat{J}_{\pm}] = \pm\hat{J}_{\pm}$. This means that if $|\psi\rangle$ is an eigenstate of \hat{J}_z ($\hat{J}_z|\psi\rangle = m|\psi\rangle$), then $\hat{J}_{\pm}|\psi\rangle$ are also eigenstates of \hat{J}_z : $\hat{J}_z\hat{J}_{\pm}|\psi\rangle = (m \pm 1)\hat{J}_{\pm}|\psi\rangle$ (if they don't vanish). Therefore, eigenvalues m of \hat{J}_z form a progression with unit step, and the ladder operators \hat{J}_{\pm} increase and decrease m .

Let's consider states with a given eigenvalue of $\hat{\mathbf{J}}^2$. For these states, eigenvalues of \hat{J}_z are bounded from above and from below because the operator $\hat{\mathbf{J}}^2 - \hat{J}_z^2 = \hat{J}_x^2 + \hat{J}_y^2$ is positive definite. Let $|m_{\pm}\rangle$ be the eigenstates with the maximum and the minimum eigenvalues of \hat{J}_z equal to m_{\pm} . Then these eigenvalues cannot be further increased and decreased by the operators \hat{J}_{\pm} correspondingly: $\hat{J}_{\pm}|m_{\pm}\rangle = 0$. We have

$$\hat{J}_{\pm}\hat{J}_{\mp} = \hat{\mathbf{J}}^2 - \hat{J}_z^2 \pm \hat{J}_z.$$

Therefore, $\hat{J}_\mp \hat{J}_\pm |m_\pm\rangle = 0 = [\hat{J}^2 - m_\pm(m_\pm \pm 1)] |m_\pm\rangle$, i.e., the eigenvalue of the operator \hat{J}^2 for these states (as well as for all the other states being considered) is $m_+(m_+ + 1) = m_-(m_- - 1)$. Hence $(m_+ + m_-)(m_+ - m_- + 1) = 0$; taking into account $m_+ \geq m_-$ we obtain $m_- = -m_+$ or $m_\pm = \pm j$. The number j must be integer or half-integer because m_+ and m_- differ by an integer.

Finally, we have a system of common eigenstates $|j, m\rangle$ of the operators \hat{J}^2 and \hat{J}_z :

$$\hat{J}^2 |j, m\rangle = j(j+1) |j, m\rangle, \quad \hat{J}_z |j, m\rangle = m |j, m\rangle,$$

where j is integer or half-integer, and m varies from $-j$ to j by 1. The operators \hat{J}_\pm increase and decrease m correspondingly: $\hat{J}_\pm |j, m\rangle = a_\pm(j, m) |j, m \pm 1\rangle$. Tuning the phases of $|j, m\rangle$ we can make $a_\pm(j, m)$ real and positive. They can be found from the normalization: $|a_\pm(j, m)|^2 = \langle j, m | \hat{J}_\mp \hat{J}_\pm |j, m\rangle = j(j+1) - m(m \pm 1)$. Finally we arrive at

$$\hat{J}_\pm |j, m\rangle = \sqrt{j(j+1) - m(m \pm 1)} |j, m \pm 1\rangle = \sqrt{(j \pm m + 1)(j \mp m)} |j, m \pm 1\rangle.$$

The orbital angular momentum of a particle $\hat{\mathbf{l}} = \mathbf{r} \times \hat{\mathbf{p}}$ (where $\hat{\mathbf{p}} = -i\nabla$ in the coordinate representation) is an example of angular momentum. In spherical coordinates

$$\hat{l}_z = -i \frac{\partial}{\partial \varphi}, \quad \hat{l}_\pm = e^{\pm i\varphi} \left(\pm \frac{\partial}{\partial \theta} + i \cot \theta \frac{\partial}{\partial \varphi} \right).$$

The eigenfunctions of \hat{l}_z are $e^{im\varphi}$; they must not change at $\varphi \rightarrow \varphi + 2\pi$; hence m must be integer. The common eigenfunctions of \hat{l}^2 and \hat{l}_z are called spherical harmonics:

$$\hat{l}^2 Y_{lm}(\theta, \varphi) = l(l+1) Y_{lm}(\theta, \varphi), \quad \hat{l}_z Y_{lm}(\theta, \varphi) = m Y_{lm}(\theta, \varphi),$$

where l is integer, and m varies from $-l$ to l by 1; $Y_{lm}(\theta, \varphi) = P_{lm}(\theta) e^{im\varphi}$. They are orthonormalized:

$$\int Y_{l'm'}^*(\theta, \varphi) Y_{lm}(\theta, \varphi) d\Omega = \delta_{l'l} \delta_{m'm}.$$

Here are the operators \hat{l}_\pm in *Mathematica*:

In[1] := `lp[f_] := Together[Exp[I * φ] * (D[f, θ] + I * Cot[θ] * D[f, φ])]`

In[2] := `lm[f_] := Together[Exp[-I * φ] * (-D[f, θ] + I * Cot[θ] * D[f, φ])]`

17.2 $Y_{ll}(\theta, \varphi)$

The angular momentum projection of this state cannot be raised by \hat{l}_+ .

In[3] := `Eq = lp[Exp[I * l * φ] * P[θ]] == 0`

Out[3] := `-e^{iφ+lφ} (l, Cot[θ]P[θ] - P'[θ]) == 0`

In[4] := s = DSolve[Eq, P[θ], θ]

Out[4] = $\left\{ \left\{ P[\theta] \rightarrow C[1] \text{Sin}[\theta]^l \right\} \right\}$

In[5] := P = P[θ]/.s[[1]]

Out[5] = $C[1] \text{Sin}[\theta]^l$

The normalization integral.

**In[6] := NI = 2 * Pi * Integrate[P^2 * Sin[θ], {θ, 0, Pi}, Assumptions -> {l ≥ 0}]/
Gamma[-l] -> -Pi/(Sin[Pi * l] * Gamma[l + 1])**

Out[6] = $\frac{2\pi^{3/2} C[1]^2 \text{Gamma}[1 + l]}{\text{Gamma}\left[\frac{3}{2} + l\right]}$

In[7] := s = Solve[NI == 1, C[1]]

Out[7] = $\left\{ \left\{ C[1] \rightarrow -\frac{\sqrt{\text{Gamma}\left[\frac{3}{2} + l\right]}}{\sqrt{2}\pi^{3/4} \sqrt{\text{Gamma}[1 + l]}} \right\}, \left\{ C[1] \rightarrow \frac{\sqrt{\text{Gamma}\left[\frac{3}{2} + l\right]}}{\sqrt{2}\pi^{3/4} \sqrt{\text{Gamma}[1 + l]}} \right\} \right\}$

In[8] := P = P/.s[[2]]

Out[8] = $\frac{\sqrt{\text{Gamma}\left[\frac{3}{2} + l\right]} \text{Sin}[\theta]^l}{\sqrt{2}\pi^{3/4} \sqrt{\text{Gamma}[1 + l]}}$

The phase can be chosen arbitrarily. According to Landau-Lifshitz [18]:

In[9] := Y[l, -l] = (-l)^l * P * Exp[l * l * φ]

Out[9] = $\frac{(-i)^l e^{i l \varphi} \sqrt{\text{Gamma}\left[\frac{3}{2} + l\right]} \text{Sin}[\theta]^l}{\sqrt{2}\pi^{3/4} \sqrt{\text{Gamma}[1 + l]}}$

In[10] := Clear[Eq, s, P]

17.3 $Y_{lm}(\theta, \varphi)$

These states can be obtained from $Y_{ll}(\theta, \varphi)$ by the lowering operator \hat{L}_- .

In[11] := S = Cos[x_]^n -> (1 - Sin[x]^2)^Quotient[n, 2] * Cos[x]^Mod[n, 2];

In[12] := Y[l, m] /; m < l := Y[l, m] =

Factor[Expand[lm[Y[l, m + 1]]/Sqrt[(l - m) * (l + m + 1)]]/.S]

In[13] := Table[Table[Y[l, m], {m, l, -l, -1}], {l, 0, 4}]

Out[13] = $\left\{ \left\{ \frac{1}{2\sqrt{\pi}} \right\}, \left\{ -\frac{1}{2} i e^{i\varphi} \sqrt{\frac{3}{2\pi}} \text{Sin}[\theta], \frac{1}{2} i \sqrt{\frac{3}{\pi}} \text{Cos}[\theta], \frac{1}{2} i e^{-i\varphi} \sqrt{\frac{3}{2\pi}} \text{Sin}[\theta] \right\}, \left\{ -\frac{1}{4} e^{2i\varphi} \sqrt{\frac{15}{2\pi}} \text{Sin}[\theta]^2, \frac{1}{2} e^{i\varphi} \sqrt{\frac{15}{2\pi}} \text{Cos}[\theta] \text{Sin}[\theta], \frac{1}{4} \sqrt{\frac{5}{\pi}} (-2 + 3 \text{Sin}[\theta]^2), -\frac{1}{2} e^{-i\varphi} \sqrt{\frac{15}{2\pi}} \text{Cos}[\theta] \text{Sin}[\theta], -\frac{1}{4} e^{-2i\varphi} \sqrt{\frac{15}{2\pi}} \text{Sin}[\theta]^2 \right\}, \right\}$

$$\left\{ \frac{1}{8}ie^{3i\varphi}\sqrt{\frac{35}{\pi}}\text{Sin}[\theta]^3, -\frac{1}{4}ie^{2i\varphi}\sqrt{\frac{105}{2\pi}}\text{Cos}[\theta]\text{Sin}[\theta]^2, \right. \\ \left. -\frac{1}{8}ie^{i\varphi}\sqrt{\frac{21}{\pi}}\text{Sin}[\theta](-4+5\text{Sin}[\theta]^2), \frac{1}{4}i\sqrt{\frac{7}{\pi}}\text{Cos}[\theta](-2+5\text{Sin}[\theta]^2), \right. \\ \left. \frac{1}{8}ie^{-i\varphi}\sqrt{\frac{21}{\pi}}\text{Sin}[\theta](-4+5\text{Sin}[\theta]^2), -\frac{1}{4}ie^{-2i\varphi}\sqrt{\frac{105}{2\pi}}\text{Cos}[\theta]\text{Sin}[\theta]^2, \right. \\ \left. -\frac{1}{8}ie^{-3i\varphi}\sqrt{\frac{35}{\pi}}\text{Sin}[\theta]^3 \right\}, \\ \left\{ \frac{3}{16}e^{4i\varphi}\sqrt{\frac{35}{2\pi}}\text{Sin}[\theta]^4, -\frac{3}{8}e^{3i\varphi}\sqrt{\frac{35}{\pi}}\text{Cos}[\theta]\text{Sin}[\theta]^3, \right. \\ \left. -\frac{3}{8}e^{2i\varphi}\sqrt{\frac{5}{2\pi}}\text{Sin}[\theta]^2(-6+7\text{Sin}[\theta]^2), \right. \\ \left. \frac{3}{8}e^{i\varphi}\sqrt{\frac{5}{\pi}}\text{Cos}[\theta]\text{Sin}[\theta](-4+7\text{Sin}[\theta]^2), \frac{3(8-40\text{Sin}[\theta]^2+35\text{Sin}[\theta]^4)}{16\sqrt{\pi}}, \right. \\ \left. -\frac{3}{8}e^{-i\varphi}\sqrt{\frac{5}{\pi}}\text{Cos}[\theta]\text{Sin}[\theta](-4+7\text{Sin}[\theta]^2), \right. \\ \left. -\frac{3}{8}e^{-2i\varphi}\sqrt{\frac{5}{2\pi}}\text{Sin}[\theta]^2(-6+7\text{Sin}[\theta]^2), \frac{3}{8}e^{-3i\varphi}\sqrt{\frac{35}{\pi}}\text{Cos}[\theta]\text{Sin}[\theta]^3, \right. \\ \left. \frac{3}{16}e^{-4i\varphi}\sqrt{\frac{35}{2\pi}}\text{Sin}[\theta]^4 \right\}$$

In[14] := Manipulate[Manipulate[Y[l, m], {m, -l, l, 1, Appearance->"Labeled"}], {l, 0, 4, 1, Appearance->"Labeled"}]

Out[14] =

$-\frac{1}{2}e^{-i\varphi}\sqrt{\frac{15}{2\pi}}\text{Cos}[\theta]\text{Sin}[\theta]$

Orthogonality of $Y_{l_1 m_1}$ and $Y_{l_2 m_2}$ with $m_1 \neq m_2$ is evident; let's check all the rest.

```

In[15] := Table[Table[Table[
  Integrate[Y[11, m] * Conjugate[Y[12, m]] * Sin[θ], {φ, 0, 2 * Pi}, {θ, 0, Pi}],
  {m, 12, -12, -1}], {12, 0, 11}], {11, 0, 4}]
```

Out[15] = {{{1}}, {{0}, {1, 1, 1}}, {{0}, {0, 0, 0}, {1, 1, 1, 1, 1}},
 {{0}, {0, 0, 0}, {0, 0, 0, 0, 0}, {1, 1, 1, 1, 1, 1, 1}},
 {{0}, {0, 0, 0}, {0, 0, 0, 0, 0}, {0, 0, 0, 0, 0, 0, 0}, {1, 1, 1, 1, 1, 1, 1, 1, 1}}}

Chapter 18

Adding Angular Momenta in Quantum Mechanics

Let \hat{J}_1 and \hat{J}_2 be two angular momentum operators commuting with each other. Then the basis $|j_1, m_1; j_2, m_2\rangle$ of common eigenstates of the operators $\hat{J}_1^2, \hat{J}_{1z}, \hat{J}_2^2, \hat{J}_{2z}$ exists. On the other hand, the total angular momentum $\hat{J} = \hat{J}_1 + \hat{J}_2$ is also an angular momentum operator. Therefore, linear combinations $|j, m\rangle$ of the states $|j_1, m_1; j_2, m_2\rangle$ at given j_1, j_2 can be constructed in such a way that they are eigenstates of \hat{J}^2 and \hat{J}_z . This problem [18] is called addition of the angular momenta j_1 and j_2 .

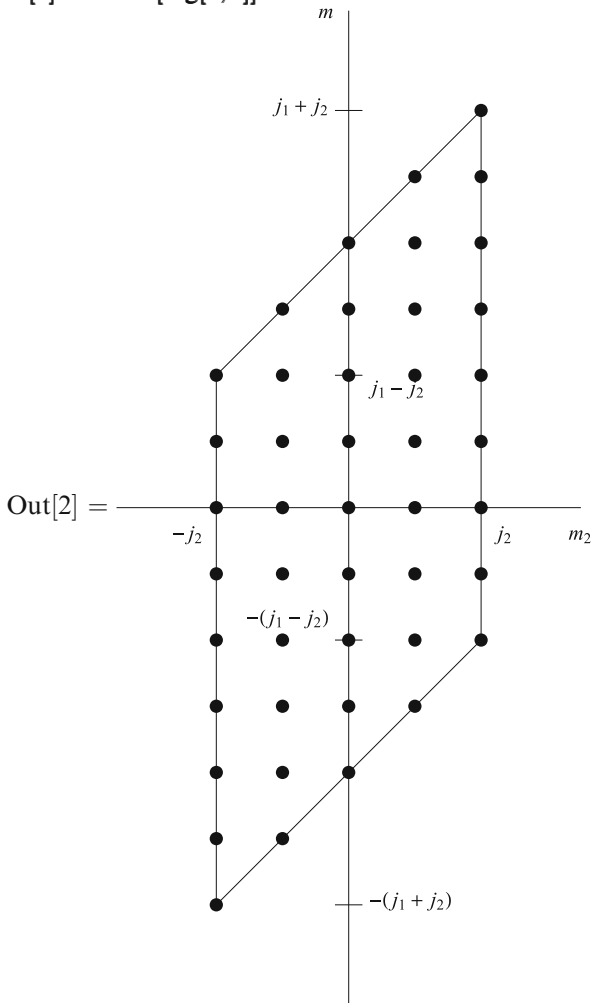
We always have $m = m_1 + m_2$ because $\hat{J}_z = \hat{J}_{1z} + \hat{J}_{2z}$. The following figure illustrates addition of j_1 and j_2 (it assumes $j_1 \geq j_2$).

```

In[1] := Fig[j1_, j2_] := If[j1 < j2, Fig[j2, j1],
  With[{d = 0.75 * j2, d2 = 0.1 * j2, d3 = 0.15 * j2, r = 0.05 * j2},
    Graphics[Join[{Line[{{j2, j1 + j2}, {j2, j2 - j1}, {-j2, -j1 - j2},
      {-j2, j1 - j2}, {j2, j1 + j2}}],
      Line[{{0, -j1 - j2 - d}, {0, j1 + j2 + d}}],
      Line[{{-j2 - d, 0}, {j2 + d, 0}}],
      Line[{{-d2, j1 + j2}, {d2, j1 + j2}}],
      Text[j1 + j2, {-d3, j1 + j2}, {1, 0}],
      Line[{{-d2, -j1 - j2}, {d2, -j1 - j2}}],
      Text[-(j1 + j2), {d3, -j1 - j2}, {-1, 0}],
      Line[{{-d2, j1 - j2}, {d2, j1 - j2}}],
      Text[j1 - j2, {d3, j1 - j2}, {-1, 1}],
      Line[{{-d2, j2 - j1}, {d2, j2 - j1}}],
      Text[-(j1 - j2), {-d3, j2 - j1}, {1, -1}],
      Text[-j2, {-j2 - d2, -d2}, {1, 1}], Text[j2, {j2 + d2, -d2}, {-1, 1}],
      Text[m, {-d2, j1 + j2 + d}, {1, 0}], Text[m2, {j2 + d, -d2}, {0, 1}]],
    Join[Table[Disk[{m2, m}, r], {m, j1 - j2 + 1, j1 + j2}, {m2, m - j1, j2}],
      Join[Table[Disk[{m2, m}, r], {m, -j1 - j2, j2 - j1 - 1}, {m2, -j2, m + j1}],
        Join[Table[Disk[{m2, m}, r], {m, j2 - j1, j1 - j2}, {m2, -j2, j2}]]]]]]]

```

In[2] := Show[Fig[4,2]]



In[3] := Clear[Fig]

There is one state with $m = j_1 + j_2$, two states with $m = j_1 + j_2 - 1$, etc. Such an increase of the number of states occurs up to $m = j_1 - j_2$; further on it is constant up to $m = -(j_1 - j_2)$ and then decreases to one at $m = -(j_1 + j_2)$. Therefore, the maximum angular momentum resulting from adding j_1 and j_2 is $j = j_1 + j_2$. One state in the two-dimensional space of states with $m = j_1 + j_2 - 1$ refers to the same angular momentum, and the other one is the state with the maximum projection for the angular momentum $j = j_1 + j_2 - 1$. Continuing such reasoning, we see that all angular momenta up to $j_1 - j_2$ appear. In general, adding angular momenta j_1 and j_2 results in the angular momenta j from $|j_1 - j_2|$ to $j_1 + j_2$ in steps of 1.

This description naturally leads to the algorithm implemented below. We start from the only state with $m = j_1 + j_2$, namely the state $|j_1, j_1; j_2, j_2\rangle$. It has $j = j_1 + j_2$, i.e., it is $|j_1 + j_2, j_1 + j_2\rangle$. Repeatedly acting by the ladder operator $\hat{J}_- = \hat{J}_{1-} + \hat{J}_{2-}$ (and dividing by the appropriate normalization factor) we construct all the states with the total angular momentum $j = j_1 + j_2$: $|j_1 + j_2, j_1 + j_2 - 1\rangle, \dots, |j_1 + j_2, -(j_1 + j_2)\rangle$. Then we turn to the projection $m = j_1 + j_2 - 1$ and choose the state orthogonal to the already constructed one $|j_1 + j_2, j_1 + j_2 - 1\rangle$. It has $j = j_1 + j_2 - 1$, i.e., it is $|j_1 + j_2 - 1, j_1 + j_2 - 1\rangle$. Using the ladder operator we construct all the states with $j = j_1 + j_2 - 1$.

Then we proceed in a similar way. At the beginning of each step, when considering a new value of the projection m , we need to construct the state orthogonal to all the states with the same m already constructed. This is achieved as follows: we start from an arbitrary state, say, $|j_1, j_1; j_2, m - j_1\rangle$, subtract its components along the already constructed states, and finally normalize the result. Then we construct all the states with the same total angular momentum from this state repeatedly acting by \hat{J}_- .

The function `AddJ` constructs the states $|j, m\rangle$ (denoted `Ket[j, m]`) as linear combinations of the states $|j_1, m_1; j_2, m_2\rangle$ (denoted `ket[m1, m2]`). It uses two local functions: `Jm` is the lowering operator \hat{J}_- and `ScaP` is the scalar product. The procedure returns its local `Ket`, so that later the user will be able to inquire about `Ket[j, m]` for specific values of j, m ; in addition to this, the procedure prints all its results.

```
In[4] := AddJ = Function[{j1, j2}, If[j2 > j1, AddJ[j2, j1],
Module[{Ket, j, J, m,
  Jm = Function[{k},
    Expand[k/.ket[m1 _, m2 _]->
      Sqrt[(j1 - m1 + 1) * (j1 + m1)] * ket[m1 - 1, m2] +
      Sqrt[(j2 - m2 + 1) * (j2 + m2)] * ket[m1, m2 - 1]]],
  ScaP = Function[{k1, k2},
    Expand[k1 * k2]/.
      {ket[m1 _, m2 _]^2->1, ket[m1 _, m2 _] * ket[M1 _, M2 _]->0}],
  Do[Ket[j, j] = ket[j - j2, j2];
    Do[Ket[j, j] = Expand[ScaP[Ket[j, j], Ket[J, j]] * Ket[J, j]],
      {J, j1 + j2, j + 1, -1}];
    Print["Ket[" , j , ", " , j , "] = " ,
      Ket[j, j] = Expand[Ket[j, j]/Sqrt[ScaP[Ket[j, j], Ket[j, j]]]];
    Do[Print["Ket[" , j , ", " , m , "] = " ,
      Ket[j, m] = Expand[Jm[Ket[j, m + 1]]/Sqrt[(j - m) * (j + m + 1)]],
      {m, j - 1, -j, -1}];
    {j, j1 + j2, j1 - j2, -1}];
  Ket]]];
In[5] := AddJ[1/2, 1/2]
Ket[1, 1] = ket  $\left[ \frac{1}{2}, \frac{1}{2} \right]$ 
Ket[1, 0] =  $\frac{\text{ket} \left[ -\frac{1}{2}, \frac{1}{2} \right]}{\sqrt{2}} + \frac{\text{ket} \left[ \frac{1}{2}, -\frac{1}{2} \right]}{\sqrt{2}}$ 
```

$$\text{Ket}[1, -1] = \text{ket} \left[-\frac{1}{2}, -\frac{1}{2} \right]$$

$$\text{Ket}[0, 0] = \frac{\text{ket} \left[-\frac{1}{2}, \frac{1}{2} \right]}{\sqrt{2}} - \frac{\text{ket} \left[\frac{1}{2}, -\frac{1}{2} \right]}{\sqrt{2}}$$

$$\text{Out}[5] = \text{Ket}\$668$$

$$\mathbf{In}[6] := \mathbf{AddJ}[1, 1/2]$$

$$\text{Ket} \left[\frac{3}{2}, \frac{3}{2} \right] = \text{ket} \left[1, \frac{1}{2} \right]$$

$$\text{Ket} \left[\frac{3}{2}, \frac{1}{2} \right] = \sqrt{\frac{2}{3}} \text{ket} \left[0, \frac{1}{2} \right] + \frac{\text{ket} \left[1, -\frac{1}{2} \right]}{\sqrt{3}}$$

$$\text{Ket} \left[\frac{3}{2}, -\frac{1}{2} \right] = \frac{\text{ket} \left[-1, \frac{1}{2} \right]}{\sqrt{3}} + \sqrt{\frac{2}{3}} \text{ket} \left[0, -\frac{1}{2} \right]$$

$$\text{Ket} \left[\frac{3}{2}, -\frac{3}{2} \right] = \text{ket} \left[-1, -\frac{1}{2} \right]$$

$$\text{Ket} \left[\frac{1}{2}, \frac{1}{2} \right] = \frac{\text{ket} \left[0, \frac{1}{2} \right]}{\sqrt{3}} - \sqrt{\frac{2}{3}} \text{ket} \left[1, -\frac{1}{2} \right]$$

$$\text{Ket} \left[\frac{1}{2}, -\frac{1}{2} \right] = \sqrt{\frac{2}{3}} \text{ket} \left[-1, \frac{1}{2} \right] - \frac{\text{ket} \left[0, -\frac{1}{2} \right]}{\sqrt{3}}$$

$$\text{Out}[6] = \text{Ket}\$669$$

$$\mathbf{In}[7] := \mathbf{AddJ}[1, 1]$$

$$\text{Ket}[2, 2] = \text{ket}[1, 1]$$

$$\text{Ket}[2, 1] = \frac{\text{ket}[0, 1]}{\sqrt{2}} + \frac{\text{ket}[1, 0]}{\sqrt{2}}$$

$$\text{Ket}[2, 0] = \frac{\text{ket}[-1, 1]}{\sqrt{6}} + \sqrt{\frac{2}{3}} \text{ket}[0, 0] + \frac{\text{ket}[1, -1]}{\sqrt{6}}$$

$$\text{Ket}[2, -1] = \frac{\text{ket}[-1, 0]}{\sqrt{2}} + \frac{\text{ket}[0, -1]}{\sqrt{2}}$$

$$\text{Ket}[2, -2] = \text{ket}[-1, -1]$$

$$\text{Ket}[1, 1] = \frac{\text{ket}[0, 1]}{\sqrt{2}} - \frac{\text{ket}[1, 0]}{\sqrt{2}}$$

$$\text{Ket}[1, 0] = \frac{\text{ket}[-1, 1]}{\sqrt{2}} - \frac{\text{ket}[1, -1]}{\sqrt{2}}$$

$$\text{Ket}[1, -1] = \frac{\text{ket}[-1, 0]}{\sqrt{2}} - \frac{\text{ket}[0, -1]}{\sqrt{2}}$$

$$\text{Ket}[0, 0] = \frac{\text{ket}[-1, 1]}{\sqrt{3}} - \frac{\text{ket}[0, 0]}{\sqrt{3}} + \frac{\text{ket}[1, -1]}{\sqrt{3}}$$

$$\text{Out}[7] = \text{Ket}\$670$$

$$\mathbf{In}[8] := \mathbf{AddJ}[2, 1]$$

$$\text{Ket}[3, 3] = \text{ket}[2, 1]$$

$$\text{Ket}[3, 2] = \sqrt{\frac{2}{3}} \text{ket}[1, 1] + \frac{\text{ket}[2, 0]}{\sqrt{3}}$$

$$\text{Ket}[3, 1] = \sqrt{\frac{2}{5}} \text{ket}[0, 1] + 2\sqrt{\frac{2}{15}} \text{ket}[1, 0] + \frac{\text{ket}[2, -1]}{\sqrt{15}}$$

$$\begin{aligned}
\text{Ket}[3,0] &= \frac{\text{ket}[-1,1]}{\sqrt{5}} + \sqrt{\frac{3}{5}}\text{ket}[0,0] + \frac{\text{ket}[1,-1]}{\sqrt{5}} \\
\text{Ket}[3,-1] &= \frac{\text{ket}[-2,1]}{\sqrt{15}} + 2\sqrt{\frac{2}{15}}\text{ket}[-1,0] + \sqrt{\frac{2}{5}}\text{ket}[0,-1] \\
\text{Ket}[3,-2] &= \frac{\text{ket}[-2,0]}{\sqrt{3}} + \sqrt{\frac{2}{3}}\text{ket}[-1,-1] \\
\text{Ket}[3,-3] &= \text{ket}[-2,-1] \\
\text{Ket}[2,2] &= \frac{\text{ket}[1,1]}{\sqrt{3}} - \sqrt{\frac{2}{3}}\text{ket}[2,0] \\
\text{Ket}[2,1] &= \frac{\text{ket}[0,1]}{\sqrt{2}} - \frac{\text{ket}[1,0]}{\sqrt{6}} - \frac{\text{ket}[2,-1]}{\sqrt{3}} \\
\text{Ket}[2,0] &= \frac{\text{ket}[-1,1]}{\sqrt{2}} - \frac{\text{ket}[1,-1]}{\sqrt{2}} \\
\text{Ket}[2,-1] &= \frac{\text{ket}[-2,1]}{\sqrt{3}} + \frac{\text{ket}[-1,0]}{\sqrt{6}} - \frac{\text{ket}[0,-1]}{\sqrt{2}} \\
\text{Ket}[2,-2] &= \sqrt{\frac{2}{3}}\text{ket}[-2,0] - \frac{\text{ket}[-1,-1]}{\sqrt{3}} \\
\text{Ket}[1,1] &= \frac{\text{ket}[0,1]}{\sqrt{10}} - \sqrt{\frac{3}{10}}\text{ket}[1,0] + \sqrt{\frac{3}{5}}\text{ket}[2,-1] \\
\text{Ket}[1,0] &= \sqrt{\frac{3}{10}}\text{ket}[-1,1] - \sqrt{\frac{2}{5}}\text{ket}[0,0] + \sqrt{\frac{3}{10}}\text{ket}[1,-1] \\
\text{Ket}[1,-1] &= \sqrt{\frac{3}{5}}\text{ket}[-2,1] - \sqrt{\frac{3}{10}}\text{ket}[-1,0] + \frac{\text{ket}[0,-1]}{\sqrt{10}} \\
\text{Out}[8] &= \text{Ket}\$671
\end{aligned}$$

Chapter 19

Classical Nonlinear Oscillator

19.1 Statement of the Problem

Let's consider one-dimensional motion of a particle with mass m near a minimum of an arbitrary smooth potential [19]

$$U(x) = \frac{kx^2}{2} + V(x), \quad V(x) = O(x^3)$$

(we have chosen the origin of x and the zero energy level to be at the minimum). If we neglect $V(x)$, then the equation of motion

$$m \frac{d^2x}{dt^2} = -\frac{dU}{dx}$$

becomes

$$\frac{d^2x}{dt^2} + \omega_0^2 x = 0, \quad \omega_0^2 = \frac{k}{m},$$

and has the solution

$$x(t) = a \cos \omega_0 t + b \sin \omega_0 t.$$

Now we consider the effect of

$$V(x) = \sum_{n=1}^{\infty} c_n x^{n+2}.$$

Choosing units of measurement in such a way that $m = 1$ and $k = 1$, we have the equation of motion

$$\frac{d^2x}{dt^2} + x = R(x) \equiv -\frac{dV}{dx}.$$

Its solution $x(t)$ is a periodic function of t . If we choose the time origin at a maximum of $x(t)$, then $x(t)$ is an even function, due to reversibility. In the zeroth approximation $x(t) = a \cos t$.

In[1] := V = Series[c[1] * x^3, {x, 0, 3}]

Out[1] = $c[1]x^3 + O[x]^4$

In[2] := R = -D[V, x]

Out[2] = $-3c[1]x^2 + O[x]^3$

In[3] := x[t] = Series[a * Cos[t], {a, 0, 1}]

Out[3] = $\text{Cos}[t]a + O[a]^2$

The equation of motion is satisfied at $O(a)$.

In[4] := D[x[t], {t, 2}] + x[t]

Out[4] = $O[a]^2$

19.2 The First Correction

Now we want to take terms of order a^2 into account. The right-hand side is

In[5] := R1 = R/.x->x[t]

Out[5] = $-3(c[1]\text{Cos}[t]^2)a^2 + O[a]^3$

Let's expand it in harmonics.

In[6] := R1 = Map[TrigReduce, R1]

Out[6] = $-\frac{3}{2}(c[1] + c[1]\text{Cos}[2t])a^2 + O[a]^3$

That is, the "driving force" contains zeroth and second harmonics. This means that we should add such harmonics to $x(t)$. We'll not add the solution of the homogeneous equation—the first harmonic: by definition of the amplitude a , it is completely given by the leading term $a \cos t$.

In[7] := x[t] = Series[a * Cos[t] +

$a^2 * \text{Sum}[b[2, j] * \text{Cos}[j * t], \{j, 0, 2, 2\}], \{a, 0, 2\}]$

Out[7] = $\text{Cos}[t]a + (b[2, 0] + b[2, 2]\text{Cos}[2t])a^2 + O[a]^3$

Now we substitute this form of the solution into the equation of motion.

In[8] := Eq = D[x[t], {t, 2}] + x[t] - (R/.x->x[t])

Out[8] = $(b[2, 0] + 3c[1]\text{Cos}[t]^2 - 3b[2, 2]\text{Cos}[2t])a^2 + O[a]^3$

In[9] := Eq = Map[TrigReduce, Eq]

Out[9] = $\frac{1}{2}(2b[2, 0] + 3c[1] - 6b[2, 2]\text{Cos}[2t] + 3c[1]\text{Cos}[2t])a^2 + O[a]^3$

In[10] := Eq2 = SeriesCoefficient[Eq, 2]

Out[10] = $\frac{1}{2}(2b[2, 0] + 3c[1] - 6b[2, 2]\text{Cos}[2t] + 3c[1]\text{Cos}[2t])$

This expression should vanish. How can we separate harmonics? Let's help *Mathematica* a little.

In[11] := Eq2 = Eq2/.Cos[j * t]->z^j

Out[11] = $\frac{1}{2}(2b[2, 0] - 6z^2b[2, 2] + 3c[1] + 3z^2c[1])$

The coefficients of z^0 and z^2 should vanish.

In[12] := Eq20 = Coefficient[Eq2, z, 0]

Out[12] = $\frac{1}{2}(2b[2, 0] + 3c[1])$

In[13] := Eq22 = Coefficient[Eq2, z, 2]

$$\text{Out[13]} = \frac{1}{2}(-6b[2, 2] + 3c[1])$$

We can find $b[2, 0]$ from the first equation and $b[2, 2]$ from the second one.

In[14] := b[2, 0] = b[2, 0] /. Solve[Eq20 == 0, b[2, 0]][[1]]

$$\text{Out[14]} = -\frac{3c[1]}{2}$$

In[15] := b[2, 2] = b[2, 2] /. Solve[Eq22 == 0, b[2, 2]][[1]]

$$\text{Out[15]} = \frac{c[1]}{2}$$

Now we know the solution.

In[16] := x[t] = x[t]

$$\text{Out[16]} = \text{Cos}[t]a + \left(-\frac{3c[1]}{2} + \frac{1}{2}c[1]\text{Cos}[2t] \right) a^2 + O[a]^3$$

Let's check energy conservation.

In[17] := Et = D[x[t], t]^2/2 + x[t]^2/2 + (V/.x->x[t]);

In[18] := Map[TrigReduce, Et]

$$\text{Out[18]} = \frac{a^2}{2} + O[a]^4$$

In[19] := Clear[b]

19.3 The Second Correction

Now we want to find two corrections.

In[20] := n = 2;

In[21] := V = Series[Sum[c[i] * x^(i + 2), {i, 1, n}], {x, 0, n + 2}]

$$\text{Out[21]} = c[1]x^3 + c[2]x^4 + O[x]^5$$

In[22] := R = -D[V, x]

$$\text{Out[22]} = -3c[1]x^2 - 4c[2]x^3 + O[x]^4$$

This is $x[t]$ up to a^2 .

In[23] := x[t] = Series[a * Cos[t] +

$$a^2 * \text{Sum}[b[2, j] * \text{Cos}[j * t], \{j, 0, 2, 2\}], \{a, 0, n\}$$

$$\text{Out[23]} = \text{Cos}[t]a + (b[2, 0] + b[2, 2]\text{Cos}[2t])a^2 + O[a]^3$$

The right-hand side of the equation of motion.

In[24] := R1 = Map[TrigReduce, ExpandAll[R /. x->x[t]]]

$$\begin{aligned} \text{Out[24]} = & -\frac{3}{2}(c[1] + c[1]\text{Cos}[2t])a^2 + \\ & (-6b[2, 0]c[1]\text{Cos}[t] - 3b[2, 2]c[1]\text{Cos}[t] - 3c[2]\text{Cos}[t] - \\ & 3b[2, 2]c[1]\text{Cos}[3t] - c[2]\text{Cos}[3t])a^3 + \\ & O[a]^4 \end{aligned}$$

There are the first and the third harmonics at the order a^3 , that is, there is a resonant term in the "driving force" which would lead to an unbounded growth of the solution. This means we have forgotten something. Namely, we have forgotten that the oscillation period depends on the amplitude (unless the potential is strictly

parabolic). And our solution should contain $\cos(j\omega t)$. If we denote $\tau = \omega t$, then the equation of motion is

$$\omega^2 \frac{d^2 x}{d\tau^2} + x = R.$$

Let's suppose that the variable t in the program really means τ and denote $\omega^2 = w$. It is a series in a^2 beginning with 1.

In[25] := w = Series[1 + Sum[u[i] * a^i, {i, 2, n + 1, 2}], {a, 0, n + 1}]

Out[25] = 1 + u[2]a^2 + O[a]^4

Now we are able to cancel the first harmonic in the a^3 term of the equation of motion. And the third one should be added to the general form of the solution.

In[26] := x[t] = Series[a * Cos[t] + a^2 * Sum[b[2, j] * Cos[j * t], {j, 0, 2, 2}] + a^3 * Sum[b[3, j] * Cos[j * t], {j, 3, 3, 2}], {a, 0, n + 1}]

Out[26] = Cos[t]a + (b[2, 0] + b[2, 2]Cos[2t])a^2 + b[3, 3]Cos[3t]a^3 + O[a]^4

The equation of motion is

**In[27] := Eq = Map[TrigReduce,
ExpandAll[w * D[x[t], {t, 2}] + x[t] - (R/.x->x[t])]]**

**Out[27] = $\frac{1}{2}(2b[2, 0] + 3c[1] - 6b[2, 2]\text{Cos}[2t] + 3c[1]\text{Cos}[2t])a^2 +$
 $(6b[2, 0]c[1]\text{Cos}[t] + 3b[2, 2]c[1]\text{Cos}[t] + 3c[2]\text{Cos}[t] -$
 $8b[3, 3]\text{Cos}[3t] + 3b[2, 2]c[1]\text{Cos}[3t] + c[2]\text{Cos}[3t] - \text{Cos}[t]u[2])a^3 +$
 $O[a]^4$**

We already know how to solve it at the order a^2 .

In[28] := Eq2 = SeriesCoefficient[Eq, 2]/.Cos[j..*t]->z^j

Out[28] = $\frac{1}{2}(2b[2, 0] - 6z^2b[2, 2] + 3c[1] + 3z^2c[1])$

**In[29] := Do[Print[b[2, j] = b[2, j]/.Solve[Coefficient[Eq2, z, j] == 0, b[2, j]][[1]],
{j, 0, 2, 2}]**

$-\frac{3c[1]}{2}$
 $\frac{c[1]}{2}$

At the order a^3 :

In[30] := Eq3 = SeriesCoefficient[Eq, 3]/.Cos[j..*t]->z^j

Out[30] = $-8z^3b[3, 3] - \frac{15}{2}zc[1]^2 + \frac{3}{2}z^3c[1]^2 + 3zc[2] + z^3c[2] - zu[2]$

This is the coefficient of the first harmonic, i.e., of z^1 :

In[31] := Eq31 = Coefficient[Eq3, z, 1]

Out[31] = $-\frac{15}{2}c[1]^2 + 3c[2] - u[2]$

It can be nullified by choosing $u[2]$.

In[32] := u[2] = u[2]/.Solve[Eq31 == 0, u[2]][[1]]

Out[32] = $-\frac{3}{2}(5c[1]^2 - 2c[2])$

And this is the coefficient of the third harmonic, i.e., of z^3 :

In[33] := Eq33 = Coefficient[Eq3, z, 3]

$$\text{Out[33]} = -8b[3, 3] + \frac{3c[1]^2}{2} + c[2]$$

It can be nullified by choosing $b[3, 3]$.

In[34] := b[3, 3] = b[3, 3]/.Solve[Eq33 == 0, b[3, 3]][[1]]

$$\text{Out[34]} = \frac{1}{16} (3c[1]^2 + 2c[2])$$

Now we know the oscillation frequency

In[35] := w = w

$$\text{Out[35]} = 1 - \frac{3}{2} (5c[1]^2 - 2c[2]) a^2 + O[a]^4$$

and $x[t]$:

In[36] := x[t] = x[t]

$$\text{Out[36]} = \text{Cos}[t]a + \left(-\frac{3c[1]}{2} + \frac{1}{2}c[1]\text{Cos}[2t] \right) a^2 + \frac{1}{16} (3c[1]^2 + 2c[2]) \text{Cos}[3t]a^3 + O[a]^4$$

Let's check energy conservation.

In[37] := Et = Map[TrigReduce,

ExpandAll[w * D[x[t], t]^2/2 + x[t]^2/2 + (V/.x->x[t])]]

$$\text{Out[37]} = \frac{a^2}{2} + \frac{1}{16} (-37c[1]^2 + 18c[2]) a^4 + O[a]^5$$

In[38] := Clear[b, u]

19.4 The n th Correction

Now we'll write a program which can find n corrections in a to the particle motion for any n . Just a single line should be changed for the calculation with a new value of n .

In[39] := n = 4;

The correction to the potential and the "driving force."

In[40] := V = Series[Sum[c[i] * x^(i + 2), {i, 1, n}], {x, 0, n + 2}]

$$\text{Out[40]} = c[1]x^3 + c[2]x^4 + c[3]x^5 + c[4]x^6 + O[x]^7$$

In[41] := R = -D[V, x]

$$\text{Out[41]} = -3c[1]x^2 - 4c[2]x^3 - 5c[3]x^4 - 6c[4]x^5 + O[x]^6$$

The frequency squared is a series in a^2 .

In[42] := w = Series[1 + Sum[u[i] * a^i, {i, 2, n + 1, 2}], {a, 0, n + 1}]

$$\text{Out[42]} = 1 + u[2]a^2 + u[4]a^4 + O[a]^6$$

The general form of the solution. The order a^i contains harmonics up to the i th one. They are all even at even values of i and odd at odd values. The first harmonic never appears—by definition, it is entirely contained in the leading term $a \cos t$.

In[43] := $x[t] = \text{Series}[a * \text{Cos}[t] + \text{Sum}[a^i * \text{Sum}[b[i, j] * \text{Cos}[j * t], \{j, \text{If}[\text{EvenQ}[i], 0, 3], i, 2\}], \{i, 2, n + 1\}], \{a, 0, n + 1\}]$

Out[43] = $\text{Cos}[t]a + (b[2, 0] + b[2, 2]\text{Cos}[2t])a^2 + b[3, 3]\text{Cos}[3t]a^3 + (b[4, 0] + b[4, 2]\text{Cos}[2t] + b[4, 4]\text{Cos}[4t])a^4 + (b[5, 3]\text{Cos}[3t] + b[5, 5]\text{Cos}[5t])a^5 + O[a]^6$

The equation of motion.

In[44] := $\text{Eq} = \text{Map}[\text{TrigReduce}, \text{ExpandAll}[w * D[x[t], \{t, 2\}] + x[t] - (R/.x \rightarrow x[t])]]$

Out[44] = $\frac{1}{2}(2b[2, 0] + 3c[1] - 6b[2, 2]\text{Cos}[2t] + 3c[1]\text{Cos}[2t])a^2 + (6b[2, 0]c[1]\text{Cos}[t] + 3b[2, 2]c[1]\text{Cos}[t] + 3c[2]\text{Cos}[t] - 8b[3, 3]\text{Cos}[3t] + 3b[2, 2]c[1]\text{Cos}[3t] + c[2]\text{Cos}[3t] - \text{Cos}[t]u[2])a^3 + \frac{1}{8}(8b[4, 0] + 24b[2, 0]^2c[1] + 12b[2, 2]^2c[1] + 48b[2, 0]c[2] + 24b[2, 2]c[2] + 15c[3] - 24b[4, 2]\text{Cos}[2t] + 48b[2, 0]b[2, 2]c[1]\text{Cos}[2t] + 24b[3, 3]c[1]\text{Cos}[2t] + 48b[2, 0]c[2]\text{Cos}[2t] + 48b[2, 2]c[2]\text{Cos}[2t] + 20c[3]\text{Cos}[2t] - 120b[4, 4]\text{Cos}[4t] + 12b[2, 2]^2c[1]\text{Cos}[4t] + 24b[3, 3]c[1]\text{Cos}[4t] + 24b[2, 2]c[2]\text{Cos}[4t] + 5c[3]\text{Cos}[4t] - 32b[2, 2]\text{Cos}[2t]u[2])a^4 +$

$(3b[2, 2]b[3, 3]c[1]\text{Cos}[t] + 6b[4, 0]c[1]\text{Cos}[t] + 3b[4, 2]c[1]\text{Cos}[t] + 12b[2, 0]^2c[2]\text{Cos}[t] + 12b[2, 0]b[2, 2]c[2]\text{Cos}[t] + 6b[2, 2]^2c[2]\text{Cos}[t] + 3b[3, 3]c[2]\text{Cos}[t] + 15b[2, 0]c[3]\text{Cos}[t] + 10b[2, 2]c[3]\text{Cos}[t] + \frac{15}{4}c[4]\text{Cos}[t] - 8b[5, 3]\text{Cos}[3t] + 6b[2, 0]b[3, 3]c[1]\text{Cos}[3t] + 3b[4, 2]c[1]\text{Cos}[3t] + 3b[4, 4]c[1]\text{Cos}[3t] + 12b[2, 0]b[2, 2]c[2]\text{Cos}[3t] + 3b[2, 2]^2c[2]\text{Cos}[3t] + 6b[3, 3]c[2]\text{Cos}[3t] + 5b[2, 0]c[3]\text{Cos}[3t] + \frac{15}{2}b[2, 2]c[3]\text{Cos}[3t] + \frac{15}{8}c[4]\text{Cos}[3t] - 24b[5, 5]\text{Cos}[5t] + 3b[2, 2]b[3, 3]c[1]\text{Cos}[5t] + 3b[4, 4]c[1]\text{Cos}[5t] + 3b[2, 2]^2c[2]\text{Cos}[5t] + 3b[3, 3]c[2]\text{Cos}[5t] + \frac{5}{2}b[2, 2]c[3]\text{Cos}[5t] + \frac{3}{8}c[4]\text{Cos}[5t] - 9b[3, 3]\text{Cos}[3t]u[2] - \text{Cos}[t]u[4])a^5 +$

$O[a]^6$

All terms of the orders a^i for i from 2 to $n + 1$ must vanish. If i is odd, the first harmonic is present; a correction to the frequency squared is found from the condition that this harmonic vanishes. All other harmonics give us coefficients in $x(t)$.

In[45] := $\text{Do}[\text{Eqi} = \text{SeriesCoefficient}[\text{Eq}, i] /. \text{Cos}[j * t] \rightarrow z^j; \text{If}[\text{OddQ}[i], u[i - 1] = u[i - 1] /. \text{Solve}[\text{Coefficient}[\text{Eqi}, z, 1] == 0, u[i - 1]][[1]]]; \text{Do}[b[i, j] = b[i, j] /. \text{Solve}[\text{Coefficient}[\text{Eqi}, z, j] == 0, b[i, j]][[1]], \{j, \text{If}[\text{EvenQ}[i], 0, 3], i, 2\}], \{i, 2, n + 1\}]$

Now we know the frequency squared.

In[46] := w = w

$$\text{Out[46]} = 1 - \frac{3}{2} (5c[1]^2 - 2c[2]) a^2 -$$

$$\frac{3}{32} (335c[1]^4 - 572c[1]^2c[2] - 4c[2]^2 + 280c[1]c[3] - 40c[4]) a^4 + O[a]^6$$

and $x(t)$

In[47] := x[t] = x[t]

$$\text{Out[47]} = \text{Cos}[t]a + \left(-\frac{3c[1]}{2} + \frac{1}{2}c[1]\text{Cos}[2t] \right) a^2 + \frac{1}{16} (3c[1]^2 + 2c[2]) \text{Cos}[3t]a^3 +$$

$$\left(-\frac{3}{8} (19c[1]^3 - 20c[1]c[2] + 5c[3]) +$$

$$\frac{1}{48} (177c[1]^3 - 186c[1]c[2] + 40c[3]) \text{Cos}[2t] +$$

$$\frac{1}{48} (3c[1]^3 + 6c[1]c[2] + 2c[3]) \text{Cos}[4t] \right) a^4 +$$

$$\left(\frac{3}{256} (237c[1]^4 - 172c[1]^2c[2] - 28c[2]^2 - 12c[1]c[3] + 20c[4]) \text{Cos}[3t] +$$

$$\frac{1}{768} (15c[1]^4 + 60c[1]^2c[2] + 12c[2]^2 + 44c[1]c[3] + 12c[4]) \text{Cos}[5t] \right) a^5 +$$

$$O[a]^6$$

Let's check energy conservation.

In[48] := Et = Map[TrigReduce,

ExpandAll[w * D[x[t], t]^2/2 + x[t]^2/2 + (V/.x->x[t])]]

$$\text{Out[48]} = \frac{a^2}{2} + \frac{1}{16} (-37c[1]^2 + 18c[2]) a^4 +$$

$$\frac{1}{1536} (-9309c[1]^4 + 17796c[1]^2c[2] + 300c[2]^2 - 10880c[1]c[3] +$$

$$1920c[4]) a^6 +$$

$$O[a]^7$$

It is easy to write a function with the parameter n which can be used as a black box. It should use only local variables.

Now we save the results for the energy Et and the frequency squared w to a file; later we'll compare them to the similar results in quantum mechanics.

In[49] := Ec = Normal[Et]/.a->Sqrt[2 * A];

Wc = Normal[Simplify[Sqrt[w]]]/.a->Sqrt[2 * A];

Save["class.m", {Ec, Wc}]

Chapter 20

Quantum Nonlinear Oscillator

20.1 Perturbation Theory

Suppose we know eigenvalues and eigenstates of a Hamiltonian \hat{H}_0 and want to find them for a Hamiltonian $\hat{H} = \hat{H}_0 + \hat{V}$ in the form of series in \hat{V} [18]. Let's concentrate on a non-degenerate eigenstate of the unperturbed Hamiltonian

$$\hat{H}_0 |\psi_0\rangle = E_0 |\psi_0\rangle.$$

After switching the perturbation on it transforms to a similar eigenstate of the full Hamiltonian

$$\hat{H} |\psi\rangle = E |\psi\rangle, \quad E = E_0 + \delta E.$$

Let's normalize $|\psi\rangle$ in such a way that $\langle \psi_0 | \psi \rangle = 1$, then $|\psi\rangle = |\psi_0\rangle + |\delta\psi\rangle$, $\langle \psi_0 | \delta\psi \rangle = 0$. We need to solve the equation

$$\hat{H}_0 |\psi\rangle + \hat{V} |\psi\rangle = E |\psi\rangle.$$

Let's separate its components parallel and orthogonal to $|\psi_0\rangle$. The parallel part is singled out by multiplying by $\langle \psi_0 |$:

$$\delta E = \langle \psi_0 | \hat{V} | \psi \rangle.$$

The orthogonal part is singled out by the projector $\hat{P} = 1 - |\psi_0\rangle\langle \psi_0|$:

$$\hat{H}_0 |\delta\psi\rangle + \hat{P}\hat{V} |\psi\rangle = E |\delta\psi\rangle,$$

or

$$|\delta\psi\rangle = \hat{D}\hat{V} |\psi\rangle, \quad \hat{D} = \frac{\hat{P}}{E - \hat{H}_0}.$$

Solving this equation by iterations, we obtain

$$|\delta\psi\rangle = \hat{D}\hat{V} |\psi_0\rangle + \hat{D}\hat{V}\hat{D}\hat{V} |\psi_0\rangle + \dots,$$

and

$$\delta E = \langle \psi_0 | \hat{V} | \psi_0 \rangle + \langle \psi_0 | \hat{V} \hat{D} \hat{V} | \psi_0 \rangle + \langle \psi_0 | \hat{V} \hat{D} \hat{V} \hat{D} \hat{V} | \psi_0 \rangle + \dots$$

Note that \hat{D} contains $E = E_0 + \delta E$ and should be expanded in δE :

$$\hat{D} = \hat{G} - \hat{G} \delta E \hat{G} + \hat{G} \delta E \hat{G} \delta E \hat{G} - \dots, \quad \hat{G} = \frac{\hat{P}}{E_0 - \hat{H}_0}.$$

20.2 Nonlinear Oscillator

We are going to apply the perturbation theory to the nonlinear oscillator

$$\hat{H}_0 = \frac{\hat{p}^2 + \hat{x}^2}{2}, \quad \hat{V} = \sum_{k=1}^{\infty} c_k \hat{x}^{k+2}.$$

The oscillation amplitude is ~ 1 if $n \sim 1$; therefore, $c_k \sim a^k$, where $a \sim 1/L \ll 1$, L is the characteristic length of the potential in the oscillator units. If $n \gg 1$, the amplitude is \sqrt{n} times larger, and the real expansion parameter is $a\sqrt{n}$.

We concentrate on the eigenstate $|n\rangle$ of \hat{H}_0 having the energy $E_0 = n + \frac{1}{2}$. In order to calculate δE up to the M th order of the perturbation theory, we need to sum all expressions of the form

$$\begin{aligned} & (\hat{V}_{k_N})_{n,n+j_{N-1}} \hat{G}_{n+j_{N-1}} (\hat{\Delta}_{k_{N-1}})_{n+j_{N-1},n+j_{N-2}} \hat{G}_{n+j_{N-2}} \dots \\ & \hat{G}_{n+j_2} (\hat{\Delta}_{k_2})_{n+j_2,n+j_1} \hat{G}_{n+j_1} (\hat{V}_{k_1})_{n+j_1,n}, \end{aligned}$$

where $\hat{\Delta}$ is \hat{V} or $-\delta E$ and the sum of the orders of smallness $k_1 + k_2 + \dots + k_N \leq M$; the sum over all nonzero j_1, j_2, \dots, j_{N-1} is assumed. The following procedure prepares the values $V[k, j]$ of $(\hat{V}_k)_{n+j,n}(x[k, j])$ means $(\hat{x}^k)_{n+j,n}$:

```
In[1] := Prepare[m_] := (M = m; x[1, 1] = Sqrt[(n + 1)/2]; x[1, -1] = Sqrt[n/2];
x[1, 0] = 0;
Do[x[k, j] = If[j < k - 1, (x[1, 1]/.n -> n + j) * x[k - 1, j + 1], 0] +
If[j > 1 - k, (x[1, -1]/.n -> n + j) * x[k - 1, j - 1], 0],
{k, 2, m + 2}, {j, -k, k}];
Do[V[k, j] = Simplify[c[k] * x[k + 2, j]], {k, 1, m}, {j, -k - 2, k + 2, 2}]]
```

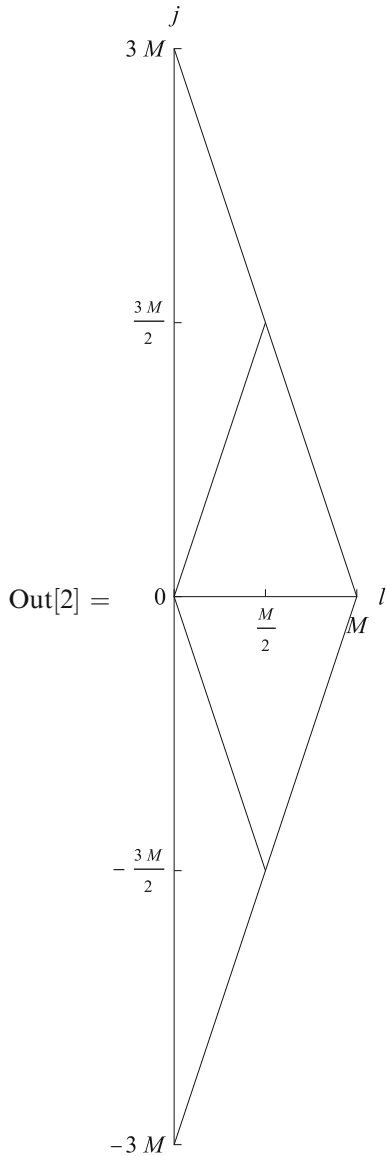
20.3 Energy Levels

The expressions we want to generate and calculate can be visualized as paths in the following graph:

```

In[2] := ParametricPlot[{{t, 3 - 3 * t}, {t, -3 + 3 * t},
  {t/2, 3/2 * t}, {t/2, -3/2 * t}}, {t, 0, 1},
  PlotRange -> {{0, 1}, {-3, 3}}, AxesLabel -> {l, j},
  Ticks -> {{0, {1/2, M/2}, {1, M}},
    {{-3, -3 * M}, {-3/2, -3/2 * M}, 0, {3/2, 3/2 * M}, {3, 3 * M}}}]

```



Here l is the number of orders of smallness we need to distribute. We start at the point $l = M$, $j = 0$. At each step we consider all possible $\hat{\Delta}_k$. If we choose $(\hat{V}_k)_{n+j+i, n+j}$, we move k steps to the left and i steps vertically. If we choose $-\delta E_k$, we move k steps to the left horizontally (this choice is not allowed as the first or the last step). Whenever we hit the $j = 0$ axis, we have a complete expression for a contribution to δE . The fastest movement along j at varying l occurs when \hat{V}_1 is used, and its velocity is 3. Hence, in order to have enough time to return to $j = 0$, we should not leave the rhombus in the figure.

Suppose we have already generated a right-hand part a of the expression up to some \hat{G}_{n+j} inclusively, and there remain l orders of smallness to distribute. The procedure considers all possible $(\hat{\Delta}_k)_{n+j+i, n+j}$ which can be inserted to the left of a . It may be $-\delta E_k$ with all possible values of k (if we are not at the very first step $l = M$), the maximum k is obtained from the rhombus. Or it may be $(\hat{V}_k)_{n+j+i, n+j}$ with all possible values of k and i . $\hat{V}_k = c_k \hat{x}^{k+2}$ has nonzero matrix elements for i from $-k-2$ to $k+2$ in steps of 2. The limits of the i loop follow from the intersection of this range with the rhombus, and the k loop terminates when this intersection disappears. If we happen to return to the initial state ($j+i=0$), this means that the generation of an expression is complete, and it should be added to the element of the list d which accumulates contributions to δE (this contribution may contain lower-order $\delta E_k = \text{de}[k]$). In all other cases, the procedure is called recursively, with l replaced by $l-k$, j by $j+i$, and a multiplied by $(\hat{\Delta}_k)_{n+j+i, n+j}$ and \hat{G}_{n+j+i} .

```
In[3] := v[l-, j-, a-] := (If[l == M, d = Table[0, {M}],
  Do[v[l - k, j, a * de[k]/j], {k, 2, l - Abs[j]/3, 2}]];
  Do[If[j + i == 0, d[[M - l + k]] += a * (V[k, i]/.n -> n + j),
    v[l - k, j + i, -a * (V[k, i]/.n -> n + j)/(j + i)],
    {k, Min[l, 1 + (3 * l - Abs[j])/2]},
    {i, Max[-k - 2, -3 * (l - k) - j], Min[k + 2, 3 * (l - k) - j], 2}]]
```

```
In[4] := Prepare[6]; v[M, 0, 1];
```

Now we substitute lower-order δE into expressions for higher-order ones, to get explicit formulas.

```
In[5] := Do[de[k] = Simplify[d[[k]]]; Print[Collect[de[k], c[_], Factor]], {k, 2, M, 2}]
```

$$\begin{aligned} & \frac{1}{8} (-11 - 30n - 30n^2) c[1]^2 + \frac{3}{4} (1 + 2n + 2n^2) c[2] \\ & - \frac{15}{32} (1 + 2n) (31 + 47n + 47n^2) c[1]^4 + \frac{9}{8} (1 + 2n) (19 + 25n + 25n^2) c[1]^2 c[2] - \\ & \frac{1}{8} (1 + 2n) (21 + 17n + 17n^2) c[2]^2 - \frac{5}{8} (1 + 2n) (13 + 14n + 14n^2) c[1] c[3] + \\ & \frac{5}{8} (1 + 2n) (3 + 2n + 2n^2) c[4] \\ & \frac{1}{128} (-39709 - 162405n - 278160n^2 - 231510n^3 - 115755n^4) c[1]^6 + \\ & \frac{3}{64} (15169 + 59385n + 98160n^2 + 77550n^3 + 38775n^4) c[1]^4 c[2] + \\ & \frac{3}{16} (111 + 347n + 472n^2 + 250n^3 + 125n^4) c[2]^3 + \end{aligned}$$

$$\begin{aligned}
& \frac{1}{16} (-4517 - 16815n - 26580n^2 - 19530n^3 - 9765n^4) c[1]^3 c[3] + \\
& \frac{1}{32} (-449 - 1400n - 2030n^2 - 1260n^3 - 630n^4) c[3]^2 - \\
& \frac{15}{8} (12 + 35n + 46n^2 + 22n^3 + 11n^4) c[2]c[4] + \\
& c[1]^2 \left(\frac{1}{32} (-11827 - 43479n - 68424n^2 - 49890n^3 - 24945n^4) c[2]^2 + \right. \\
& \quad \left. \frac{5}{16} (323 + 1125n + 1668n^2 + 1086n^3 + 543n^4) c[4] \right) + \\
& c[1] \left(\frac{3}{8} (474 + 1625n + 2430n^2 + 1610n^3 + 805n^4) c[2]c[3] - \right. \\
& \quad \left. \frac{105}{16} (5 + 16n + 22n^2 + 12n^3 + 6n^4) c[5] \right) + \\
& \frac{35}{16} (3 + 8n + 10n^2 + 4n^3 + 2n^4) c[6]
\end{aligned}$$

20.4 Correspondence Principle

At $n \gg 1$ the expansion parameter of the perturbation series is $a\sqrt{n}$ where $c_k \sim a^k$. Keeping only the highest powers of n in each order, we have

In[6] := Eq = Series[n+

Sum[(Expand[(de[2 * (j - 1)] /. n -> 1/a) * a^j] /. a -> 0) * n^j,

{j, 2, M/2 + 1}],

{n, 0, M/2 + 1}]

Out[6] = n + $\left(-\frac{15}{4} c[1]^2 + \frac{3c[2]}{2} \right) n^2 +$

$\frac{1}{16} (-705c[1]^4 + 900c[1]^2c[2] - 68c[2]^2 - 280c[1]c[3] + 40c[4])n^3 -$

$\frac{5}{128} (23151c[1]^6 - 46530c[1]^4c[2] + 19956c[1]^2c[2]^2 - 600c[2]^3 +$

$15624c[1]^3c[3] - 7728c[1]c[2]c[3] + 504c[3]^2 - 4344c[1]^2c[4] +$

$528c[2]c[4] + 1008c[1]c[5] - 112c[6])n^4 +$

$O[n]^5$

Bohr's correspondence principle must hold. From the quantum point of view, the particle at the n th energy level can radiate a photon, jumping to the $(n - 1)$ th one, or more generally to the $(n - k)$ th one. The frequency of this photon is $E_n - E_{n-k}$, or approximately $\frac{dE_n}{dn}k$. From the classical point of view, the frequencies of the emitted light are equal to the oscillation frequency ω and its harmonics. Therefore, the oscillation frequency is

$$\omega = \frac{dE_n}{dn}.$$

In[7] := Wq = D[Eq, n]

$$\begin{aligned} \text{Out[7]} &= 1 + 2 \left(-\frac{15}{4}c[1]^2 + \frac{3c[2]}{2} \right) n + \\ &\frac{3}{16} (-705c[1]^4 + 900c[1]^2c[2] - 68c[2]^2 - 280c[1]c[3] + 40c[4])n^2 - \\ &\frac{5}{32} (23151c[1]^6 - 46530c[1]^4c[2] + 19956c[1]^2c[2]^2 - 600c[2]^3 + \\ &15624c[1]^3c[3] - 7728c[1]c[2]c[3] + 504c[3]^2 - 4344c[1]^2c[4] + \\ &528c[2]c[4] + 1008c[1]c[5] - 112c[6])n^3 + \\ &O[n]^4 \end{aligned}$$

We want to compare it with the result of the calculation in classical mechanics. But the quantum expression for ω is in terms of n and the classical one in terms of the oscillation amplitude a . We need to re-express both of them via the same quantity, the energy E .

In[8] := ne = InverseSeries[Eq, e]

$$\begin{aligned} \text{Out[8]} &= e + \frac{3}{4} (5c[1]^2 - 2c[2]) e^2 + \\ &\frac{5}{16} (231c[1]^4 - 252c[1]^2c[2] + 28c[2]^2 + 56c[1]c[3] - 8c[4]) e^3 + \\ &\frac{35}{128} (7293c[1]^6 - 12870c[1]^4c[2] + 5148c[1]^2c[2]^2 - 264c[2]^3 + \\ &3432c[1]^3c[3] - 1584c[1]c[2]c[3] + 72c[3]^2 - 792c[1]^2c[4] + \\ &144c[2]c[4] + 144c[1]c[5] - 16c[6]) e^4 + \\ &O[e]^5 \end{aligned}$$

In[9] := Wqe = Simplify[Wq/.n -> ne]

$$\begin{aligned} \text{Out[9]} &= 1 + \left(-\frac{15}{2}c[1]^2 + 3c[2] \right) e - \\ &\frac{3}{16} (855c[1]^4 - 1020c[1]^2c[2] + 92c[2]^2 + 280c[1]c[3] - 40c[4]) e^2 + \\ &\frac{1}{32} (-164805c[1]^6 + 311670c[1]^4c[2] - 94920c[1]^3c[3] - \\ &180c[1]^2 (715c[2]^2 - 134c[4]) + 5040c[1] (9c[2]c[3] - c[5]) + \\ &8 (633c[2]^3 - 315c[3]^2 - 450c[2]c[4] + 70c[6])) e^3 + \\ &O[e]^4 \end{aligned}$$

Now we read the classical results for the energy E_c and the frequency W_c (written in terms of $A = a^2/2$, where the amplitude a is defined as the coefficient of the first harmonic $\cos \omega t$).

In[10] := <<class.m;

In[11] := Ec = Series[Ec, {A, 0, 3}]

$$\begin{aligned} \text{Out[11]} &= A + \left(-\frac{37}{4}c[1]^2 + \frac{9c[2]}{2} \right) A^2 + \\ &\frac{1}{192} (-9309c[1]^4 + 17796c[1]^2c[2] + 300c[2]^2 - 10880c[1]c[3] + 1920c[4]) A^3 + \\ &O[A]^4 \end{aligned}$$

In[12] := Wc = Series[Wc, {A, 0, 2}]

$$\text{Out[12]} = 1 + \left(-\frac{15}{2}c[1]^2 + 3c[2]\right)A -$$

$$\frac{3}{16} (485c[1]^4 - 692c[1]^2c[2] + 20c[2]^2 + 280c[1]c[3] - 40c[4])A^2 + O[A]^3$$

In[13] := Ae = Simplify[InverseSeries[Ec, e]]

$$\text{Out[13]} = e + \left(\frac{37c[1]^2}{4} - \frac{9c[2]}{2}\right)e^2 +$$

$$\left(\frac{14055c[1]^4}{64} - \frac{4147}{16}c[1]^2c[2] + \frac{623c[2]^2}{16} + \frac{170}{3}c[1]c[3] - 10c[4]\right)e^3 + O[e]^4$$

In[14] := Wce = Simplify[Wc/.A -> Ae]

$$\text{Out[14]} = 1 + \left(-\frac{15}{2}c[1]^2 + 3c[2]\right)e -$$

$$\frac{3}{16} (855c[1]^4 - 1020c[1]^2c[2] + 92c[2]^2 + 280c[1]c[3] - 40c[4])e^2 + O[e]^3$$

In[15] := Wqe - Wce

$$\text{Out[15]} = O[e]^3$$

20.5 States

The following procedure accumulates contributions to δE in elements of the list d and to $|\delta\psi\rangle$ in the list dp . Now we have to consider the large triangle in the figure, not just the rhombus.

**In[16] := v2[l_., j_., a_] := (dp[[M - l]] += a * ket[n + j];
 If[l < M, Do[v2[l - k, j, a * de[k]/j], {k, 2, l - 1, 2}]]];
 Do[If[j + i == 0, d[[M - l + k]] += a * (V[k, i]/.n -> n + j),
 v2[l - k, j + i, -a * (V[k, i]/.n -> n + j)/(j + i)],
 {k, l}, {i, -k - 2, k + 2, 2}]]**

**In[17] := Prepare[2]; d = Table[0, {M}]; dp = Table[0, {M}];
 Clear[de]; v2[M, 0, 1];**

**In[18] := Do[de[k] = Simplify[d[[k]]];
 Print[Collect[dp[[k]], ket[_], Simplify]], {k, M}]**

$$\frac{\sqrt{-2+n}\sqrt{-1+n}\sqrt{nc[1]}\text{ket}[-3+n]}{6\sqrt{2}} + \frac{3n^{3/2}c[1]\text{ket}[-1+n]}{2\sqrt{2}} -$$

$$\frac{3(1+n)^{3/2}c[1]\text{ket}[1+n]}{2\sqrt{2}} - \frac{\sqrt{1+n}\sqrt{2+n}\sqrt{3+nc[1]}\text{ket}[3+n]}{6\sqrt{2}}$$

$$\frac{1}{144}\sqrt{-5+n}\sqrt{-4+n}\sqrt{-3+n}\sqrt{-2+n}\sqrt{-1+n}\sqrt{nc[1]^2}\text{ket}[-6+n] +$$

$$\frac{1}{32}\sqrt{-3+n}\sqrt{-2+n}\sqrt{-1+n}\sqrt{n}((-3+4n)c[1]^2 + 2c[2])\text{ket}[-4+n] +$$

$$\begin{aligned}
& \frac{1}{16} \sqrt{-1+n} \sqrt{n} \left((1-19n+7n^2) c[1]^2 + 4(-1+2n)c[2] \right) \text{ket}[-2+n] + \\
& \frac{1}{16} \sqrt{1+n} \sqrt{2+n} \left((27+33n+7n^2) c[1]^2 - 4(3+2n)c[2] \right) \text{ket}[2+n] + \\
& \frac{1}{32} \sqrt{1+n} \sqrt{2+n} \sqrt{3+n} \sqrt{4+n} \left((7+4n)c[1]^2 - 2c[2] \right) \text{ket}[4+n] + \\
& \frac{1}{144} \sqrt{1+n} \sqrt{2+n} \sqrt{3+n} \sqrt{4+n} \sqrt{5+n} \sqrt{6+n} c[1]^2 \text{ket}[6+n]
\end{aligned}$$

As an additional problem, calculate the average values of \hat{x}^k over the states just obtained, for several k . At $n \gg 1$ compare them to the classical averages obtained from the particle's motion $x(t)$.

Chapter 21

Riemann Curvature Tensor

Catching a lion, the Einstein's method: Enter the cage and lock it from inside. Then the Universe will be subdivided into two disjoint regions in such a way that you are in one of them and the lion is in the other one. It depends on one's point of view whom to consider caught; for convenience, let's say it's the lion.

Suppose we have a coordinate system x^μ in a region of an n -dimensional Riemann (or pseudo-Riemann) manifold [20]. Components of the metric tensor $g_{\mu\nu}$ are given as functions of x^μ . We want to calculate the Riemann curvature tensor $R^\mu{}_{\nu\alpha\beta}$ and related quantities (the Ricci tensor $R_{\mu\nu}$, the scalar curvature R).

The metric tensor is symmetric; therefore, it is reasonable to ask the user to provide only the components with $\mu \geq \nu$. If the user gives an argument having a wrong shape, we print an error message and abort the calculation. We shall also need the contravariant metric tensor $g^{\mu\nu}$ defined by $g^{\mu\lambda}g_{\lambda\nu} = \delta^\mu_\nu$.

```
In[1] := Metric[g0_] := Module[{n = Length[g0], g, gu},
  Do[If[Length[g0][[μ]]] != μ, Message[Metric :: shape]; Abort[], {μ, n}];
  g = Table[If[μ ≥ ν, g0[[μ, ν]], g0[[ν, μ]]], {μ, n}, {ν, n}];
  gu = Simplify[Inverse[g]]; {n, g, gu}
```

```
In[2] := Metric :: shape = "Wrong shape of the argument";
```

Next we calculate the Christoffel symbols

$$\Gamma_{\lambda\mu\nu} = \frac{1}{2} (\partial_\mu g_{\lambda\nu} + \partial_\nu g_{\lambda\mu} - \partial_\lambda g_{\mu\nu})$$

and $\Gamma^\lambda{}_{\mu\nu} = g^{\lambda\rho}\Gamma_{\rho\mu\nu}$. They are symmetric in μ and ν ; therefore, we calculate them only at $\nu \leq \mu$ and reuse the calculated values at $\nu > \mu$. If the optional parameter `PrintNonZero` is `True`, nonzero components are printed (following the tradition, in the printed results all indices vary from 0 to $n - 1$).

```
In[3] := Christoffel[{n_, g_, gu_}, OptionsPattern[]] := Module[{Γ, Γu},
  Γ = Γu = Table[0, {λ, n}, {μ, n}, {ν, n}];
  Do[Γ[[λ, μ, ν]] = Simplify[(D[g[[λ, ν]], x[μ]] + D[g[[λ, μ]], x[ν]] -
    D[g[[μ, ν]], x[λ]])/2];
```

```

    If[ $\mu \neq \nu$ ,  $\Gamma[[\lambda, \nu, \mu]] = \Gamma[[\lambda, \mu, \nu]]$ ],
    { $\lambda, n$ }, { $\mu, n$ }, { $\nu, \mu$ }}];
Do[ $\Gamma\mathbf{u}[[\lambda, \mu, \nu]] = \text{Simplify}[\text{Sum}[\text{gu}[[\lambda, \rho]] * \Gamma[[\rho, \mu, \nu]], \{\rho, n\}]]$ ;
    If[ $\mu \neq \nu$ ,  $\Gamma\mathbf{u}[[\lambda, \nu, \mu]] = \Gamma\mathbf{u}[[\lambda, \mu, \nu]]$ ],
    { $\lambda, n$ }, { $\mu, n$ }, { $\nu, \mu$ }}];
If[OptionValue[PrintNonZero],
    Do[If[ $\Gamma[[\lambda, \mu, \nu]] \neq 0$ , Print[" $\Gamma$ ",  $\lambda - 1, \mu - 1, \nu - 1$ , " ",  $\Gamma[[\lambda, \mu, \nu]]$ ],
        { $\lambda, n$ }, { $\mu, n$ }, { $\nu, \mu$ }}];
    Do[If[ $\Gamma\mathbf{u}[[\lambda, \mu, \nu]] \neq 0$ , Print[" $\Gamma\mathbf{u}$ ",  $\lambda - 1, \mu - 1, \nu - 1$ , " ",  $\Gamma\mathbf{u}[[\lambda, \mu, \nu]]$ ],
        { $\lambda, n$ }, { $\mu, n$ }, { $\nu, \mu$ }}];
{ $\Gamma, \Gamma\mathbf{u}$ }]
In[4] := Options[Christoffel] = {PrintNonZero  $\rightarrow$  True};

```

Finally, we calculate the Riemann tensor

$$R_{\alpha\beta\mu\nu} = g_{\alpha\lambda} \left(\partial_{\mu} \Gamma^{\lambda}_{\beta\nu} - \partial_{\nu} \Gamma^{\lambda}_{\beta\mu} \right) + \Gamma_{\alpha\lambda\mu} \Gamma^{\lambda}_{\beta\nu} - \Gamma_{\alpha\lambda\nu} \Gamma^{\lambda}_{\beta\mu},$$

the Ricci tensor $R_{\mu\nu} = g^{\alpha\beta} R_{\alpha\mu\beta\nu}$, and the scalar curvature $R = g^{\mu\nu} R_{\mu\nu}$. The Riemann tensor has the properties

$$R_{\alpha\beta\mu\nu} = -R_{\beta\alpha\mu\nu} = -R_{\alpha\beta\nu\mu} = R_{\mu\nu\alpha\beta},$$

and we use them to avoid unnecessary calculations.

```

In[5] := Riemann[{n_, g_, gu_}, OptionsPattern[]] := Module[{ $\Gamma, \Gamma\mathbf{u}$ ,
    R = Table[0, { $\alpha, n$ }, { $\beta, n$ }, { $\mu, n$ }, { $\nu, n$ }], R2 = Table[0, { $\mu, n$ }, { $\nu, n$ }], R0},
    { $\Gamma, \Gamma\mathbf{u}$ } = Christoffel[{n, g, gu}];
    Do[ $R[[\alpha, \beta, \mu, \nu]] = R[[\beta, \alpha, \nu, \mu]] = \text{Simplify}[\text{Sum}[
        g[[\alpha, \lambda]] * (D[\Gamma\mathbf{u}[[\lambda, \beta, \nu]], x[\mu]] - D[\Gamma\mathbf{u}[[\lambda, \beta, \mu]], x[\nu]])
        + \Gamma[[\alpha, \lambda, \mu]] * \Gamma\mathbf{u}[[\lambda, \beta, \nu]] - \Gamma[[\alpha, \lambda, \nu]] * \Gamma\mathbf{u}[[\lambda, \beta, \mu]], \{\lambda, n\}]]$ ;
         $R[[\beta, \alpha, \mu, \nu]] = R[[\alpha, \beta, \nu, \mu]] = -R[[\alpha, \beta, \mu, \nu]]$ ;
        If[ $\mu \neq \alpha$ ,  $R[[\mu, \nu, \alpha, \beta]] = R[[\nu, \mu, \beta, \alpha]] = R[[\alpha, \beta, \mu, \nu]]$ ;
             $R[[\nu, \mu, \alpha, \beta]] = R[[\mu, \nu, \beta, \alpha]] = -R[[\alpha, \beta, \mu, \nu]]$ ],
        { $\alpha, 2, n$ }, { $\beta, \alpha - 1$ }, { $\mu, 2, \alpha$ }, { $\nu, \text{If}[\mu == \alpha, \beta, \mu - 1]$ }}];
    Do[ $R2[[\mu, \nu]] = \text{Simplify}[\text{Sum}[\text{gu}[[\alpha, \beta]] * R[[\alpha, \mu, \beta, \nu]], \{\alpha, n\}, \{\beta, n\}]]$ ;
        If[ $\mu \neq \nu$ ,  $R2[[\nu, \mu]] = R2[[\mu, \nu]]$ ],
        { $\mu, n$ }, { $\nu, \mu$ }}];
    R0 = Simplify[Sum[gu[[ $\mu, \nu$ ]] * R2[[ $\mu, \nu$ ]] * If[ $\mu \neq \nu, 2, 1$ ], { $\mu, n$ }, { $\nu, \mu$ }}];
    If[OptionValue[PrintNonZero],
        Do[If[ $R[[\alpha, \beta, \mu, \nu]] \neq 0$ ,
            Print[R,  $\alpha - 1, \beta - 1, \mu - 1, \nu - 1$ , " ",  $R[[\alpha, \beta, \mu, \nu]]$ ],
            { $\alpha, 2, n$ }, { $\beta, \alpha - 1$ }, { $\mu, 2, \alpha$ }, { $\nu, \text{If}[\mu == \alpha, \beta, \mu - 1]$ }}];
        Do[If[ $R2[[\mu, \nu]] \neq 0$ , Print["R",  $\mu - 1, \nu - 1$ , " ", R2[[ $\mu, \nu$ ]]], { $\mu, n$ }, { $\nu, \mu$ }}];
        If[R0  $\neq 0$ , Print["R ", R0]]];
    {R, R2, R0}]
In[6] := Options[Riemann] = {PrintNonZero  $\rightarrow$  True};

```

Let's consider an example: the Schwarzschild metric

$$ds^2 = \left(1 - \frac{r_0}{r}\right) dt^2 - \frac{dr^2}{1 - \frac{r_0}{r}} - r^2 (d\theta^2 + \sin^2 \theta d\phi^2).$$

First we give names to the coordinates.

In[7] := Evaluate[Table[x[μ], {μ, 4}]] = {t, r, θ, φ};

Setting the Schwarzschild radius $r_0 = 1$, we obtain

In[8] := Riemann[Metric[{{1 - 1/r}, {0, -1/(1 - 1/r)}, {0, 0, -r^2}, {0, 0, 0, -r^2 * Sin[θ]^2}}];

$$\Gamma_{010} \frac{1}{2r^2}$$

$$\Gamma_{100} - \frac{1}{2r^2}$$

$$\Gamma_{111} \frac{1}{2(-1 + r)^2}$$

$$\Gamma_{122} r$$

$$\Gamma_{133} r \text{Sin}[\theta]^2$$

$$\Gamma_{221} - r$$

$$\Gamma_{233} r^2 \text{Cos}[\theta] \text{Sin}[\theta]$$

$$\Gamma_{331} - r \text{Sin}[\theta]^2$$

$$\Gamma_{332} - r^2 \text{Cos}[\theta] \text{Sin}[\theta]$$

$$\Gamma_{u010} \frac{1}{2(-1 + r)r}$$

$$\Gamma_{u100} \frac{-1 + r}{2r^3}$$

$$\Gamma_{u111} \frac{1}{2r - 2r^2}$$

$$\Gamma_{u122} 1 - r$$

$$\Gamma_{u133} - (-1 + r) \text{Sin}[\theta]^2$$

$$\Gamma_{u221} \frac{1}{r}$$

$$\Gamma_{u233} - \text{Cos}[\theta] \text{Sin}[\theta]$$

$$\Gamma_{u331} \frac{1}{r}$$

$$\Gamma_{u332} \text{Cot}[\theta]$$

$$R_{1010} \frac{1}{r^3}$$

$$R_{2020} - \frac{-1 + r}{2r^2}$$

$$R_{2121} \frac{1}{2(-1 + r)}$$

$$R_{3030} - \frac{(-1 + r) \text{Sin}[\theta]^2}{2r^2}$$

$$R_{3131} \frac{\text{Sin}[\theta]^2}{-2 + 2r}$$

$$R_{3232} - r \text{Sin}[\theta]^2$$

The Ricci tensor (and hence the scalar curvature) vanishes. Therefore, the Schwarzschild metric satisfies the vacuum Einstein equation.

Chapter 22

Multi- ζ Functions

22.1 Definition

The Riemann ζ -function is defined by

$$\zeta_s = \sum_{n>0} \frac{1}{n^s}.$$

Mathematica knows this function; it can be expressed via powers of π for even integer values of s .

In[1] := Table[Zeta[s], {s, 2, 6}]

Out[1] = { $\frac{\pi^2}{6}$, Zeta[3], $\frac{\pi^4}{90}$, Zeta[5], $\frac{\pi^6}{945}$ }

Let's define

$$\zeta_{s_1 s_2} = \sum_{n_1 > n_2 > 0} \frac{1}{n_1^{s_1} n_2^{s_2}}, \quad \zeta_{s_1 s_2 s_3} = \sum_{n_1 > n_2 > n_3 > 0} \frac{1}{n_1^{s_1} n_2^{s_2} n_3^{s_3}},$$

and so on. These series converge at $s_1 > 1$. *Mathematica* does not know these multi- ζ functions. The sum $s_1 + s_2 + \dots + s_k$ is called the *weight*. All relations we shall discuss contain terms of the same weight (the weight of a product is the sum of the weights of its factors).

22.2 Stuffing Relations

Suppose we want to multiply $\zeta_s \zeta_{s_1 s_2}$:

$$\zeta_s \zeta_{s_1 s_2} = \sum_{\substack{n>0 \\ n_1 > n_2 > 0}} \frac{1}{n^s n_1^{s_1} n_2^{s_2}}.$$

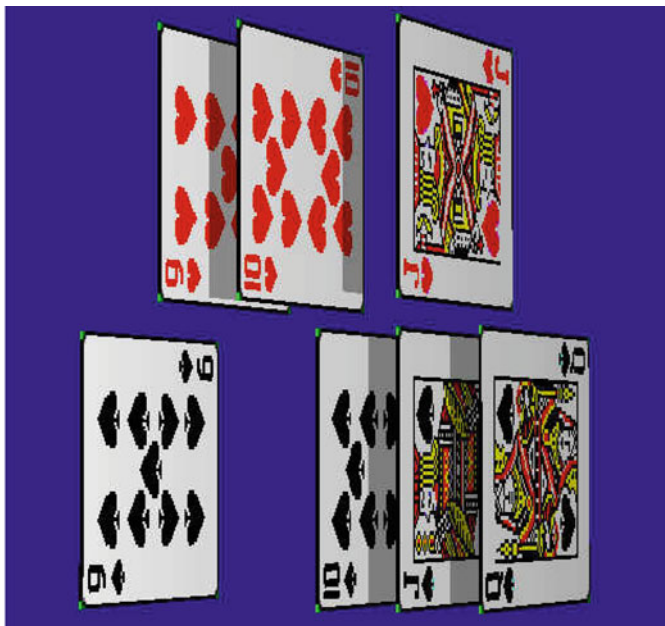
Here n can be anywhere with respect to n_1, n_2 . There are five contributions:

$$\begin{aligned} \sum_{n > n_1 > n_2 > 0} \frac{1}{n^s n_1^{s_1} n_2^{s_2}} &= \zeta_{s s_1 s_2}, \\ \sum_{n = n_1 > n_2 > 0} \frac{1}{n^s n_1^{s_1} n_2^{s_2}} &= \zeta_{s + s_1, s_2}, \\ \sum_{n_1 > n > n_2 > 0} \frac{1}{n^s n_1^{s_1} n_2^{s_2}} &= \zeta_{s_1 s s_2}, \\ \sum_{n_1 > n = n_2 > 0} \frac{1}{n^s n_1^{s_1} n_2^{s_2}} &= \zeta_{s_1, s + s_2}, \\ \sum_{n_1 > n_2 > n > 0} \frac{1}{n^s n_1^{s_1} n_2^{s_2}} &= \zeta_{s_1 s_2 s}. \end{aligned}$$

This process reminds shuffling cards. The order of cards in the upper deck, as well as in the lower one, is kept fixed. We sum over all possible shufflings. Unlike real playing cards, however, two cards may be exactly on top of each other. In this case they are stuffed together: a single card (which is their sum) appears in the resulting deck. A mathematical jargon term for such shuffling with (possible) stuffing is *stuffing*.

In[2] := Show[Import["c1.jpg"]]

Out[2] =



Let's implement this in *Mathematica*. The multi- ζ function will be called ζ ; it can have any number of arguments. The function `Stuffing` first of all transforms products of ζ functions (including squares) to a local function z with three list parameters: the first two contain the arguments of the initial ζ functions, and the third one is empty. These are our two decks for shuffling and the resulting deck, initially empty. All the work is done by the following repeated substitution. Let the two unprocessed decks be nonempty: the first one contains some front "card" a and the remainder A ; the second one—the front "card" b and the remainder B . Then there are three possibilities: either we move the front "card" from the first deck (a) to the resulting deck, or we move the front "card" from the second deck (b) to the resulting deck, or we take the front "cards" from both decks and put their sum to the resulting deck (stuffing). We need to use a delayed substitution `:>` here to ensure that the command `Expand` in its right-hand side is executed when the substitution is applied. In addition to this, we should take care of the situations when one of the source decks becomes empty. In this case we can just append the other source deck to the resulting one and yield the result.

This process can also be described as the following. The final result is a sum of many ζ functions with various argument lists. During intermediate steps, the function `z[deck1,deck2,res]` represents the sum of a subset of terms of the result whose argument lists begin with `res`. At each step we subdivide this sum into three parts, according to three possible values of the next argument.

```
In[3] := Stuffing[x_] := Module[{y,z}, y = x/.
  {z[A_...]^2->z[{A}, {A}, {}], z[A_...]*z[B_...]->z[{A}, {B}, {}]};
  y//.{z[{}], {B_...}, {C_...]}->z[C,B], z[{A_...}, {}, {C_...]}->z[C,A],
  z[{a_-,A_...}, {b_-,B_...}, {C_...}]:>
  Expand[z[{A}, {b,B}, {C,a}] + z[{a,A}, {B}, {C,b}] +
  z[{A}, {B}, {C,a+b}]]]
In[4] := Map[Stuffing, {z[x]^2, z[x]*z[y], z[x]*z[y,z]}]
Out[4] = {z[2x] + 2z[x,x], z[x+y] + z[x,y] + z[y,x],
  z[y,x+z] + z[x+y,z] + z[x,y,z] + z[y,x,z] + z[y,z,x]}
```

22.3 Integral Representation

It is easy to check the integral representation of the ζ -function of an integer argument

$$\zeta_s = \int_{1 > x_1 > \dots > x_s > 0} \frac{dx_1}{x_1} \dots \frac{dx_{s-1}}{x_{s-1}} \frac{dx_s}{1-x_s}.$$

Let's denote

$$\omega_0 = \frac{dx}{x}, \quad \omega_1 = \frac{dx}{1-x}.$$

All integrals will always have the integration region $1 > x_1 > \dots > x_s > 0$. Then

$$\zeta_s = \int \omega_0^{s-1} \omega_1.$$

This representation can be generalized to multi- ζ functions:

$$\zeta_{s_1 s_2} = \int \omega_0^{s_1-1} \omega_1 \omega_0^{s_2-1} \omega_1, \quad \zeta_{s_1 s_2 s_3} = \int \omega_0^{s_1-1} \omega_1 \omega_0^{s_2-1} \omega_1 \omega_0^{s_3-1} \omega_1,$$

and so on. Let's write functions for transforming ζ with integer arguments to such integral representation and back. The integral representation ζ_i takes an arbitrary number of arguments equal to 0 or 1 corresponding to ω_0, ω_1 .

```
In[5] := s2i[x_] := Module[{y,z}, y = x/.  $\zeta[A\_]$  -> z[{A}, {1}];
y//. {z[{1}, {B\_}] ->  $\zeta_i[B]$ ,
z[{a_, A\_}, {B\_}] :>
z[{A}, Append[Join[{B}, Table[0, {a-1}], 1]]]}
In[6] := l = Map[s2i, { $\zeta[2]$ ,  $\zeta[2,3]$ }]
Out[6] = { $\zeta_i[0,1]$ ,  $\zeta_i[0,1,0,0,1]$ }
In[7] := i2s[x_] := Module[{y,z}, y = x/.  $\zeta_i[A\_]$  -> z[{A}, {1}];
y//. {z[{1}, {B\_}] ->  $\zeta[B]$ ,
z[{a_, A\_}, {B_, b_}] :>
If[a == 0, z[{A}, {B, b+1}], z[{A}, {B, b, 1}]]]}
In[8] := Map[i2s, l]
Out[8] = { $\zeta[2]$ ,  $\zeta[2,3]$ }
In[9] := Clear[l]
```

22.4 Shuffling Relations

Suppose we want to multiply $\zeta_2 \cdot \zeta_2$:

$$\zeta_2^2 = \int_{1 > x_1 > x_2 > 0} \omega_0 \omega_1 \cdot \int_{1 > x'_1 > x'_2 > 0} \omega_0 \omega_1.$$

The order of primed and non-primed integration variables is not fixed. There are six contributions:

$$1 > x_1 > x_2 > x'_1 > x'_2 > 0 : \int \omega_0 \omega_1 \omega_0 \omega_1 = \zeta_{22};$$

$$1 > x_1 > x'_1 > x_2 > x'_2 > 0 : \int \omega_0 \omega_0 \omega_1 \omega_1 = \zeta_{31};$$

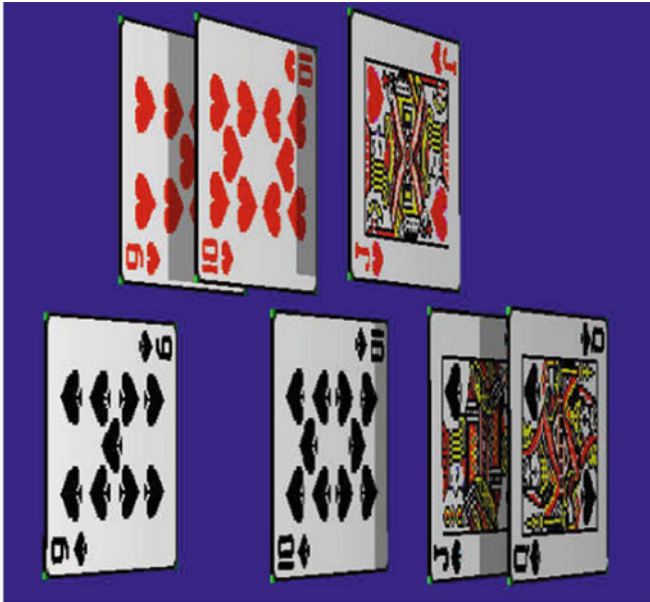
$$1 > x_1 > x'_1 > x'_2 > x_2 > 0 : \int \omega_0 \omega_0 \omega_1 \omega_1 = \zeta_{31};$$

$$\begin{aligned}
 1 > x'_1 > x_1 > x_2 > x'_2 > 0 &: \int \omega_0 \omega_0 \omega_1 \omega_1 = \zeta_{31}; \\
 1 > x'_1 > x_1 > x'_2 > x_2 > 0 &: \int \omega_0 \omega_0 \omega_1 \omega_1 = \zeta_{31}; \\
 1 > x'_1 > x'_2 > x_1 > x_2 > 0 &: \int \omega_0 \omega_1 \omega_0 \omega_1 = \zeta_{22}.
 \end{aligned}$$

Now we are multiplying integrals, not sums. Therefore our “cards” are now infinitely thin and cannot be exactly on top of each other. There are just two kinds of “cards”: ω_0 and ω_1 , and we sum over all possible shufflings of two decks.

```
In[10] := Show[Import["c2.jpg"]]
```

Out[10] =



```

In[11] := shuffling[x_] := Module[{y,z}, y = x/.
  {ζi[A_...]^2 -> z[{A}, {A}, {}],
   ζi[A_...] * ζi[B_...] -> z[{A}, {B}, {}]};
  y //. {z[{}, {B_...}, {C_...}] -> ζi[C, B], z[{A_...}, {}, {C_...}] -> ζi[C, A],
  z[{a_-, A_...}, {b_-, B_...}, {C_...}] :>
  Expand[z[{A}, {b, B}, {C, a}] + z[{a, A}, {B}, {C, b}]]]
In[12] := Shuffling[x_] := i2s[Expand[shuffling[s2i[x]]]]
In[13] := Map[Shuffling, {ζ[2]^2, ζ[2] * ζ[3], ζ[2] * ζ[2, 1]}]
Out[13] = {2ζ[2, 2] + 4ζ[3, 1], ζ[2, 3] + 3ζ[3, 2] + 6ζ[4, 1],
  ζ[2, 1, 2] + 3ζ[2, 2, 1] + 6ζ[3, 1, 1]}

```


22.5 Duality Relations

The integral representation allows us to derive another set of useful relations, even simpler than shuffling—duality relations. Let's make the substitution $x_i \rightarrow 1 - x_i$. Then $\omega_0 \longleftrightarrow \omega_1$; to preserve the order $1 > x_1 > \dots > x_s > 0$, we have to arrange all the ω factors in the opposite order. In other words, after writing down an integral representation for a multi- ζ value, we may read it in the Arabic fashion, right to left, simultaneously replacing $\omega_0 \longleftrightarrow \omega_1$. Duality relations are the only known relations which say that two multi- ζ values with distinct arguments are just equal to each other.

**In[14] := duality[x_] := Module[{y, z}, y = x/. ζ i[A_---] -> z[{A}, {}];
y//. {z[{}], {B_---}] -> ζ i[B], z[{a_-, A_---}, {B_---}] -> z[{A}, {1 - a, B}]]]**

In[15] := Duality[x_] := i2s[duality[s2i[x]]]

In[16] := Map[Duality, { ζ [3], ζ [4], ζ [5], ζ [4, 1], ζ [3, 2], ζ [2, 3]}]

Out[16] = { ζ [2, 1], ζ [2, 1, 1], ζ [2, 1, 1, 1], ζ [3, 1, 1], ζ [2, 2, 1], ζ [2, 1, 2]}

22.6 Weight 4

There are four converging multi- ζ series of weight 4: ζ_4 , ζ_{31} , ζ_{22} , and ζ_{211} . Due to duality, two of them are equal to each other.

In[17] := Duality[ζ [4]]

Out[17] = ζ [2, 1, 1]

We can express ζ_4 via ζ_2^2 using their explicit values:

In[18] := S = ζ [4] -> Zeta[4]/Zeta[2]^2 * ζ [2]^2

Out[18] = ζ [4] -> $\frac{2\zeta[2]^2}{5}$

Two equations for ζ_2^2 follow from stuffing

In[19] := eq1 = ζ [2]^2 == Stuffling[ζ [2]^2]

Out[19] = ζ [2]^2 == ζ [4] + 2 ζ [2, 2]

and shuffling

In[20] := eq2 = ζ [2]^2 == Shuffling[ζ [2]^2]

Out[20] = ζ [2]^2 == 2 ζ [2, 2] + 4 ζ [3, 1]

They can be solved for ζ_{22} and ζ_{31} (taking the expression for ζ_4 into account).

In[21] := s = Solve[{eq1/. S, eq2}, { ζ [2, 2], ζ [3, 1]}][[1]]

Out[21] = { ζ [2, 2] -> $\frac{3\zeta[2]^2}{10}$, ζ [3, 1] -> $\frac{\zeta[2]^2}{10}$ }

Thus we have demonstrated that all multi- ζ values of weight 4 can be expressed via ζ_2^2 .

In[22] := Clear[S]

22.7 Weight 5

There are four distinct multi- ζ values of weight 5,

$$\mathbf{In[23]} := \mathbf{Map}[\mathbf{Duality}, \{\zeta[5], \zeta[4, 1], \zeta[3, 2], \zeta[2, 3]\}]$$

$$\mathbf{Out[23]} = \{\zeta[2, 1, 1, 1], \zeta[3, 1, 1], \zeta[2, 2, 1], \zeta[2, 1, 2]\}$$

due to duality. The shuffling relation for $\zeta_2 \zeta_3$:

$$\mathbf{In[24]} := \mathbf{eq1} = \zeta[2] * \zeta[3] == \mathbf{Stuffing}[\zeta[2] * \zeta[3]]$$

$$\mathbf{Out[24]} = \zeta[2]\zeta[3] == \zeta[5] + \zeta[2, 3] + \zeta[3, 2]$$

A similar equation where ζ_3 is written in the dual form.

$$\mathbf{In[25]} := \mathbf{eq2} = \mathbf{Stuffing}[\zeta[2] * \mathbf{Duality}[\zeta[3]]]$$

$$\mathbf{Out[25]} = \zeta[2, 3] + \zeta[4, 1] + \zeta[2, 1, 2] + 2\zeta[2, 2, 1]$$

$$\mathbf{In[26]} := \mathbf{eq2} = \mathbf{eq2} /. \{\zeta[2, 1, 2] :> \mathbf{Duality}[\zeta[2, 1, 2]], \\ \zeta[2, 2, 1] :> \mathbf{Duality}[\zeta[2, 2, 1]]\}$$

$$\mathbf{Out[26]} = 2\zeta[2, 3] + 2\zeta[3, 2] + \zeta[4, 1]$$

$$\mathbf{In[27]} := \mathbf{eq2} = \zeta[2] * \zeta[3] == \mathbf{eq2}$$

$$\mathbf{Out[27]} = \zeta[2]\zeta[3] == 2\zeta[2, 3] + 2\zeta[3, 2] + \zeta[4, 1]$$

The shuffling relation for $\zeta_2 \zeta_3$:

$$\mathbf{In[28]} := \mathbf{eq3} = \zeta[2] * \zeta[3] == \mathbf{Shuffling}[\zeta[2] * \zeta[3]]$$

$$\mathbf{Out[28]} = \zeta[2]\zeta[3] == \zeta[2, 3] + 3\zeta[3, 2] + 6\zeta[4, 1]$$

This system can be solved for ζ_{41} , ζ_{32} , and ζ_{23} .

$$\mathbf{In[29]} := s = \mathbf{Solve}\{\{\mathbf{eq1}, \mathbf{eq2}, \mathbf{eq3}\}, \{\zeta[4, 1], \zeta[3, 2], \zeta[2, 3]\}\}\{\mathbf{1}\}$$

$$\mathbf{Out[29]} = \left\{ \zeta[4, 1] \rightarrow -\zeta[2]\zeta[3] + 2\zeta[5], \zeta[3, 2] \rightarrow 3\zeta[2]\zeta[3] - \frac{11\zeta[5]}{2}, \right. \\ \left. \zeta[2, 3] \rightarrow -2\zeta[2]\zeta[3] + \frac{9\zeta[5]}{2} \right\}$$

Thus we have demonstrated that all multi- ζ values of weight 5 can be expressed via $\zeta_2 \zeta_3$ and ζ_5 .

Chapter 23

Rainbow

23.1 Statement of the Problem

We consider scattering of light by a spherical water drop in geometrical optics. For small drops, diffraction becomes significant; our analysis is only valid for drops which are not too small. Let the drop radius be 1. The ray with the impact parameter ρ splits into the reflected ray and the refracted one. Their directions are given by the Snell law

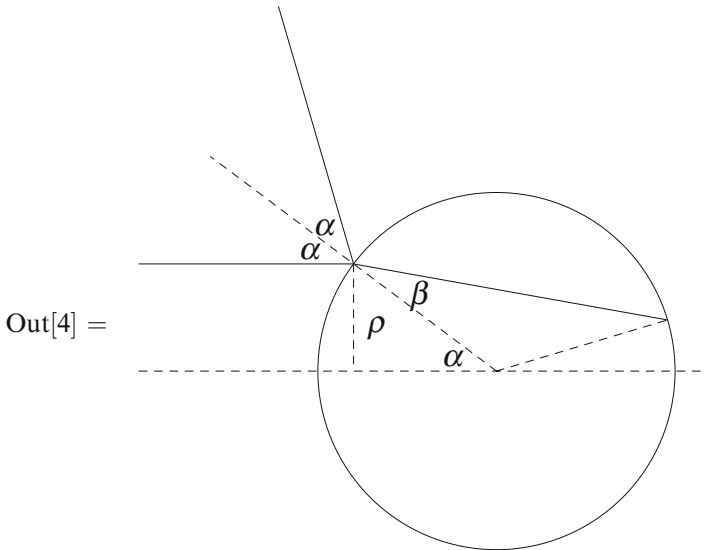
```
In[1] := Snell = { $\alpha \rightarrow \text{ArcSin}[\rho], \beta \rightarrow \text{ArcSin}[\rho/n]$ };
```

where the refraction index of water is

```
In[2] := Water = n -> 1.333;
```

```
In[3] := col = {RGBColor[1, 0, 0], RGBColor[0, 0, 1], RGBColor[0, 1, 0],
  RGBColor[1, 0, 1], RGBColor[0, 1, 1], RGBColor[1, 1, 0]};
```

```
In[4] := With[{y = 0.6},
  With[{ $\alpha = \alpha /. \text{Snell} /. \rho \rightarrow y, \beta = \beta /. \text{Snell} /. \rho \rightarrow y /. \text{Water}$ ,
    x = -Sqrt[1 - y^2]},
  With[{ $\varphi = \pi - \alpha, \vartheta = \pi - 2 * \alpha$ }, With[{ $\psi = \varphi + 2 * \beta - \pi$ },
  Graphics[{Black, Circle[], Line[{{-2, 0}, {1.2, 0}}],
    Line[{{0, 0}, {2 * Cos[ $\varphi$ ], 2 * Sin[ $\varphi$ ]}]},
    Line[{{0, 0}, {Cos[ $\psi$ ], Sin[ $\psi$ ]}]},
    Line[{{x, y}, {x, 0}}, col[[1]], Line[{{-2, y}, {x, y}},
    col[[2]], Line[{{x, y}, {x + 1.5 * Cos[ $\vartheta$ ], y + 1.5 * Sin[ $\vartheta$ ]}]},
    col[[3]], Line[{{x, y}, {Cos[ $\psi$ ], Sin[ $\psi$ ]}]},
    Black, Inset[Style[" $\alpha$ ", 24], {-0.25, 0.1}],
    Inset[Style[" $\alpha$ ", 24], {x - 0.25, y + 0.1}],
    Inset[Style[" $\alpha$ ", 24], {x - 0.15, y + 0.25}],
    Inset[Style[" $\beta$ ", 24], {x + 0.35, y - 0.15}],
    Inset[Style[" $\rho$ ", 24], {x + 0.1, 0.5 * y}]]]]]]]
```



The incident ray hits the drop at the point $\{\text{Cos}[\varphi], \text{Sin}[\varphi]\}$, where $\varphi = \pi - \alpha$. The reflected ray has the direction $\vartheta = \pi - 2\alpha$. The refracted ray hits the drop surface again at $\{\text{Cos}[\psi], \text{Sin}[\psi]\}$, where $\psi = \varphi + 2\beta - \pi$.

The incident light has two polarizations, with the electric field orthogonal to the scattering plane or lying in this plane. The reflection coefficients for these polarizations are given by the Fresnel formulas [21] (they don't depend on the direction, i.e., are the same for a ray entering water and a ray leaving it).

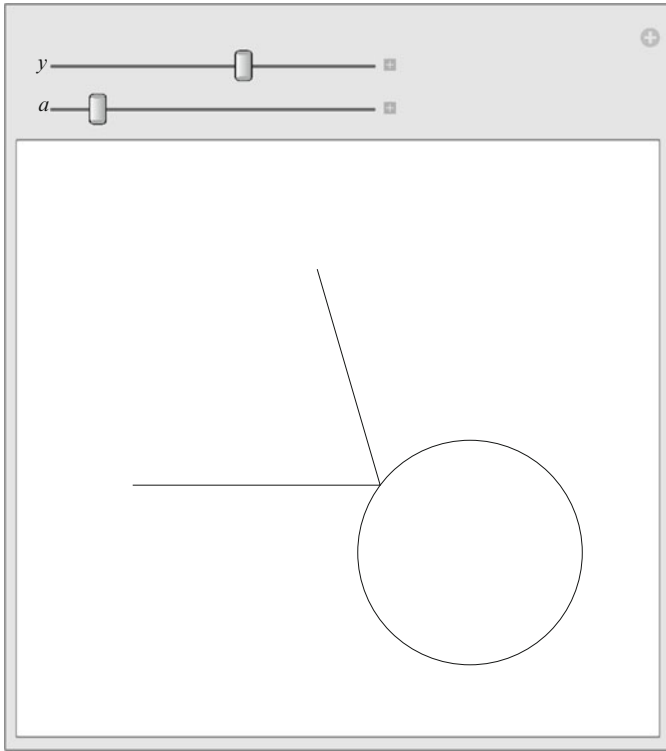
In[5] := $\mathbf{Rs} = (\text{Sin}[\alpha - \beta] / \text{Sin}[\alpha + \beta])^2$; $\mathbf{Rp} = (\text{Tan}[\alpha - \beta] / \text{Tan}[\alpha + \beta])^2$;

23.2 0 Ray Segments Inside the Drop

First let's consider rays reflected by the drop immediately after they hit its surface.

```
In[6] := Ray0[y_., a_] := With[{alpha = alpha /. Snell /. rho -> y,
  beta = beta /. Snell /. rho -> y /. Water, x = -Sqrt[1 - y^2]},
  With[{vartheta = pi - 2 * alpha}, {col[[1]], Line[{{-1 - a, y}, {x, y}}],
    col[[2]], Line[{{x, y}, {x + a * Cos[vartheta], y + a * Sin[vartheta]}]}]]]
In[7] := Manipulate[Graphics[Join[{Black, Circle[], Ray0[y, a]},
  PlotRange -> {{-1 - a, 1.1}, {-1.1, a + 0.7}}],
  {y, 0.6}, 0, 1], {a, 2}, 1, 10]
```

Out[7] =

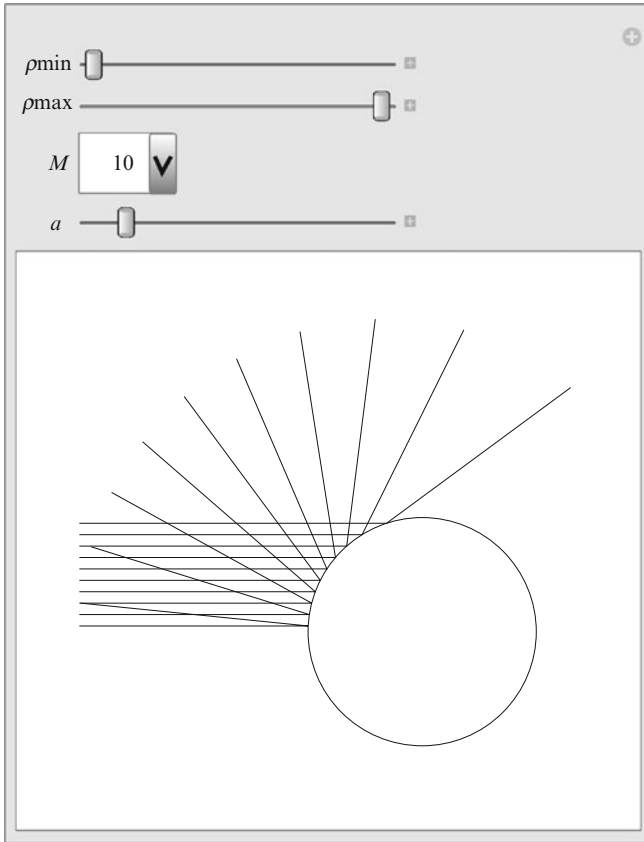


```

In[8] := Manipulate[
  Graphics[
    Join[{Black, Circle[]},
      With[{ $\delta = (\rho_{\max} - \rho_{\min})/M$ },
        If[ $\delta > 0$ , Apply[Join, Table[Ray0[y, a], {y,  $\rho_{\min} + \delta/2$ ,  $\rho_{\max}$ ,  $\delta$ }], {}]]],
      PlotRange -> {{-1 - a, 1.1}, {-1.1, a + 0.7}},
      {{ $\rho_{\min}$ , 0}, 0, 1}, {{ $\rho_{\max}$ , 1}, 0, 1},
      {{M, 10}, Table[i, {i, 30}]}, {{a, 2}, 1, 10}
    ]
  ]

```

Out[8] =

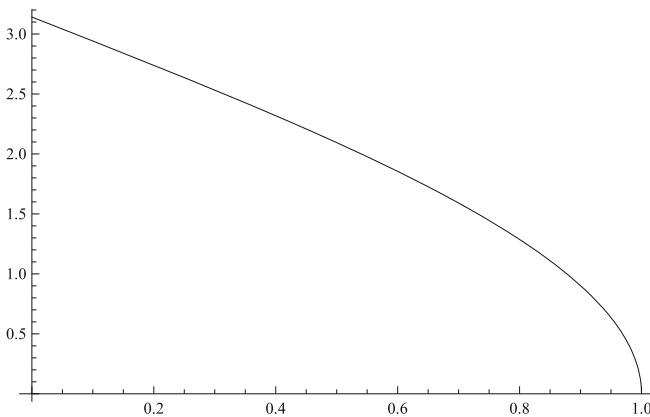


The scattering angle is

In[9] := $\vartheta_0 = \pi - 2 * \alpha$;

In[10] := Plot[Evaluate[ϑ_0 /. Snell], { ρ , 0, 1}]

Out[10] =



In order to calculate the differential cross section, we need to express ρ via the scattering angle ϑ :

In[11] := sol0 = Simplify[Solve[(ϑ_0 /.Snell) == ϑ , ρ], $0 < \vartheta < \pi$]

Out[11] = $\left\{ \left\{ \rho \rightarrow \cos \left[\frac{\vartheta}{2} \right] \right\} \right\}$

In[12] := $\rho_0 = \rho$ /.sol0[[1]]

Out[12] = $\cos \left[\frac{\vartheta}{2} \right]$

In[13] := Clear[sol0]

The angles α and β are

In[14] := Snell0 = { $\alpha \rightarrow (\pi - \vartheta)/2$, $\beta \rightarrow \text{ArcSin}[\text{Sin}[\alpha]/n]$ };

The area of the ring in the transverse plane corresponding to the scattering angles between ϑ and $\vartheta + d\vartheta$, divided by $d\Omega = 2\pi \text{Sin}[\vartheta] d\vartheta$, is

In[15] := $\sigma_0 = \text{Simplify}[-D[\rho_0^2, \vartheta]/(2 * \text{Sin}[\vartheta])]$

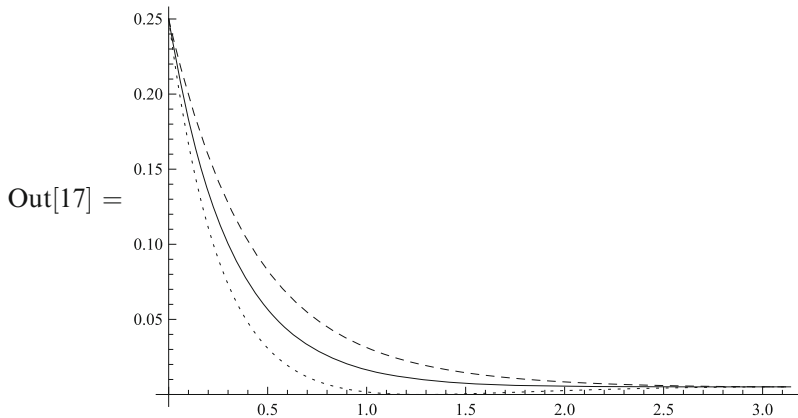
Out[15] = $\frac{1}{4}$

Therefore, the cross sections for the two polarizations are

In[16] := $\sigma_{0s}[\vartheta_] = \text{Simplify}[\sigma_0 * \text{Rs}/\text{.Snell0}/\text{.Water}$];

$\sigma_{0p}[\vartheta_] = \text{Simplify}[\sigma_0 * \text{Rp}/\text{.Snell0}/\text{.Water}$];

**In[17] := Plot[{ $\sigma_{0s}[\vartheta]$, $\sigma_{0p}[\vartheta]$, ($\sigma_{0s}[\vartheta] + \sigma_{0p}[\vartheta])/2$ }, { ϑ , 0, π },
PlotRange -> All, PlotStyle -> col]**



Note that there is a scattering angle ϑ at which the scattered light is completely polarized: its electric field is orthogonal to the scattering plane. This happens when α is equal to the Brewster angle α_B

In[18] := $\alpha_B = \text{ArcTan}[n]$; α_B /.Water

Out[18] = 0.927175

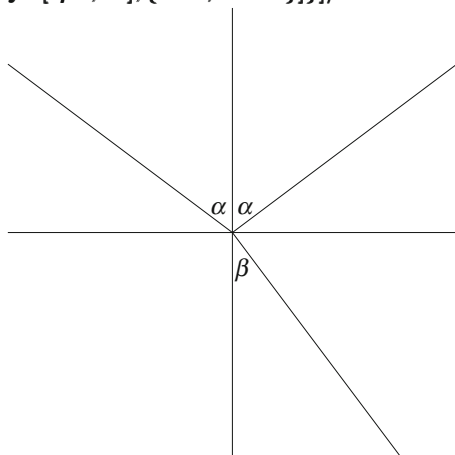
In this case

In[19] := $\beta_B = \text{ArcTan}[1/n]$;

so that $\alpha_B + \beta_B = \pi/2$ —the refracted ray is perpendicular to the reflected one:

```
In[20] := Graphics[{Line[{{-1, 0}, {1, 0}}], Line[{{0, -1}, {0, 1}}],
  col[[1]], Line[{{0, 0}, {-1, 1/n}}],
  col[[2]], Line[{{0, 0}, {1, 1/n}}],
  col[[3]], Line[{{0, 0}, {1/n, -1}}],
  Black, Inset[Style[" $\alpha$ ", 24], {-0.06, 0.13}], Inset[Style[" $\alpha$ ", 24], {0.06, 0.13}],
  Inset[Style[" $\beta$ ", 24], {0.06, -0.13}]]/. Water
```

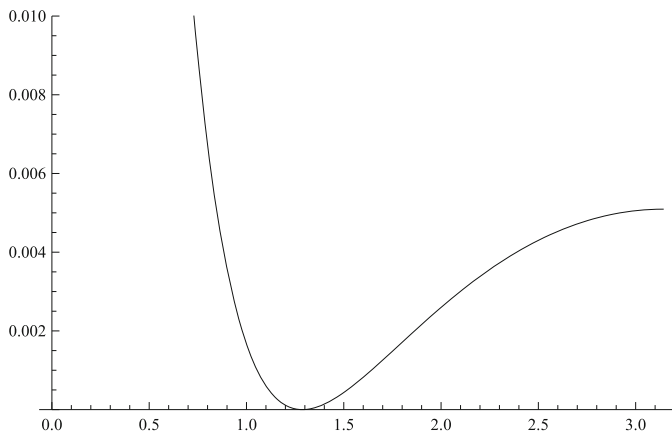
Out[20] =



When the incoming light with the electric field in the scattering plane reaches water, electric dipoles in it oscillate along the direction perpendicular to the refracted ray; they don't radiate in the direction along this axis, i.e., don't produce the reflected ray: $R_p = 0$.

```
In[21] := Plot[ $\sigma_0 p[\vartheta]$ , { $\vartheta$ , 0,  $\pi$ }, PlotRange -> {0, 0.01}]
```

Out[21] =



```
In[22] :=  $\vartheta_0$  /.  $\alpha$  ->  $\alpha B$  /. Water
```

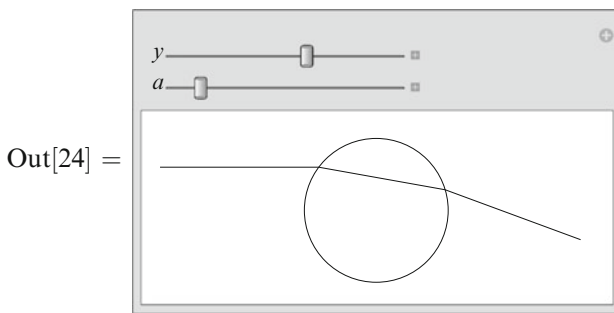
Out[22] = 1.28724

23.3 1 Ray Segment Inside the Drop

```

In[23] := Ray1[y_., a_] := With[{ $\alpha = \alpha /. \text{Snell} /. \rho \rightarrow y$ ,
   $\beta = \beta /. \text{Snell} /. \rho \rightarrow y /. \text{Water}$ ,  $x = -\text{Sqrt}[1 - y^2]$ },
  With[{ $\varphi = \pi - \alpha$ ,  $\vartheta = 2 * (\beta - \alpha)$ }, With[{ $\psi = \varphi + 2 * \beta - \pi$ },
    With[{ $x1 = \text{Cos}[\psi]$ ,  $y1 = \text{Sin}[\psi]$ }, {col[[1]], Line[{{-1 - a, y}, {x, y}}]},
    col[[3]], Line[{{x, y}, {x1, y1}}]},
    col[[2]], Line[{{x1, y1}, {x1 + a * Cos[\vartheta], y1 + a * Sin[\vartheta]}]}]}]]]]
In[24] := Manipulate[Graphics[Join[{Black, Circle[]}, Ray1[y, a]],
  PlotRange -> {{-1 - a, 1 + a}, {-1.1, 1.1}}, {{y, 0.6}, 0, 1},
  {{a, 2}, 1, 10}]

```

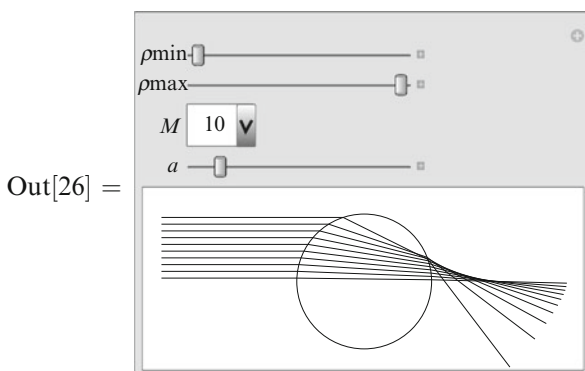


When entering water, the ray is deflected by $\alpha - \beta$ clockwise; when leaving water, it is deflected by the same angle again. The direction of the outgoing ray is

```

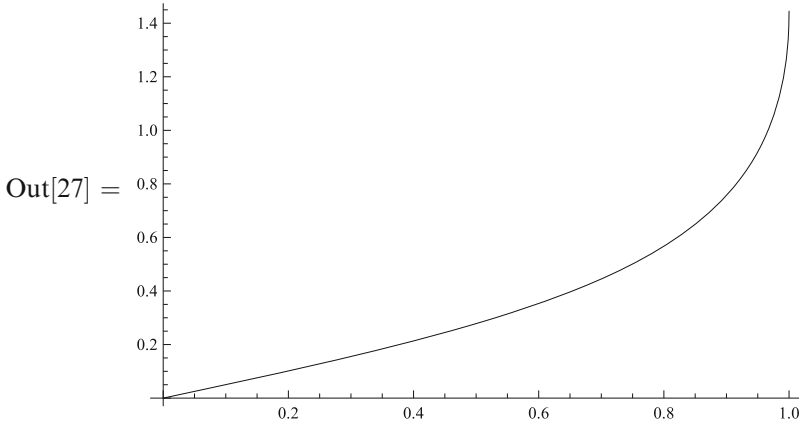
In[25] :=  $\vartheta1 = 2 * (\beta - \alpha)$ ;
In[26] := Manipulate[
  Graphics[
    Join[{Black, Circle[]},
      With[{ $\delta = (\rho_{\text{max}} - \rho_{\text{min}}) / M$ },
        If[ $\delta > 0$ , Apply[Join, Table[Ray1[y, a], {y,  $\rho_{\text{min}} + \delta / 2$ ,  $\rho_{\text{max}}$ ,  $\delta$ }], {}]]],
      PlotRange -> {{-1 - a, 1 + a}, {-1.1, 1.1}},
      {{ $\rho_{\text{min}}$ , 0}, 0, 1}, {{ $\rho_{\text{max}}$ , 1}, 0, 1},
      {{M, 10}, Table[i, {i, 30}]}, {{a, 2}, 1, 10}]

```



The scattering angle ϑ is obtained by reducing $\vartheta 1$ to the interval $[-\pi, \pi]$ and then taking Abs; in the present case, it is just $-\vartheta 1$.

In[27] := Plot[Evaluate[- $\vartheta 1$ /.Snell/.Water], { ρ , 0, 1}]



It varies from 0 to

In[28] := $\vartheta 1m = \text{Simplify}[-\vartheta 1/.Snell/.\rho - > 1]$

Out[28] = $\pi - 2 \text{ArcSin}\left[\frac{1}{n}\right]$

In[29] := $\vartheta 1m/.Water$

Out[29] = 1.4449

Now we have to solve the equation $\alpha - \beta = \vartheta/2$ for ρ .

In[30] := eq1 = (TrigExpand[Sin[$\alpha - \beta$]]/.{Sin[α]-> ρ , Cos[α]->Sqrt[1 - ρ^2], Sin[β]-> ρ/n , Cos[β]->Sqrt[1 - (ρ/n)^2]}) == Sin[$\vartheta/2$]

Out[30] = $-\frac{\rho\sqrt{1-\rho^2}}{n} + \rho\sqrt{1-\frac{\rho^2}{n^2}} == \text{Sin}\left[\frac{\vartheta}{2}\right]$

In[31] := sol1 = Solve[eq1, ρ]

Out[31] = $\left\{ \left\{ \rho \rightarrow -\frac{\sqrt{n^2 - n^2 \text{Cos}[\vartheta]}}{\sqrt{2}\sqrt{1 + n^2 - 2n \text{Cos}\left[\frac{\vartheta}{2}\right]}} \right\}, \left\{ \rho \rightarrow \frac{\sqrt{n^2 - n^2 \text{Cos}[\vartheta]}}{\sqrt{2}\sqrt{1 + n^2 - 2n \text{Cos}\left[\frac{\vartheta}{2}\right]}} \right\}, \left\{ \rho \rightarrow -\frac{\sqrt{n^2 - n^2 \text{Cos}[\vartheta]}}{\sqrt{2}\sqrt{1 + n^2 + 2n \text{Cos}\left[\frac{\vartheta}{2}\right]}} \right\}, \left\{ \rho \rightarrow \frac{\sqrt{n^2 - n^2 \text{Cos}[\vartheta]}}{\sqrt{2}\sqrt{1 + n^2 + 2n \text{Cos}\left[\frac{\vartheta}{2}\right]}} \right\} \right\}$

We discard the negative solutions. The positive ones evaluated at $\vartheta 1m$ are

In[32] := Simplify[$\rho/.sol1[[{2, 4}]]/. $\vartheta -> \vartheta 1m, n > 1$]$

Out[32] = $\left\{ 1, \sqrt{\frac{-1 + n^2}{3 + n^2}} \right\}$

So, the right solution is number 2:

```
In[33] := ρ1 = Simplify[Simplify[ρ /. sol1[[2]], {n > 1, 0 < ϑ < π}]/
Cos[ϑ] -> 1 - 2 * Sin[ϑ/2]^2, {n > 1, ϑ > 0, ϑ < π}]
```

```
Out[33] = 
$$\frac{n \sin\left[\frac{\vartheta}{2}\right]}{\sqrt{1 + n^2 - 2n \cos\left[\frac{\vartheta}{2}\right]}}$$

```

```
In[34] := Clear[eq1, sol1]
```

The geometrical cross section for ϑ between ϑ and $\vartheta + d\vartheta$, divided by $d\Omega$, is

```
In[35] := σ1 = Simplify[D[ρ1^2, ϑ]/(4 * c2 * s2)/
{Cos[ϑ/2] -> c2, Sin[ϑ/2] -> s2}]
```

```
Out[35] = 
$$\frac{n^2 (c2 - 2c2^2n + c2n^2 - ns2^2)}{4c2 (1 - 2c2n + n^2)^2}$$

```

where $c2 = \cos[\vartheta/2]$, $s2 = \sin[\vartheta/2]$. The angles α and β are

```
In[36] := Snell1 = {α -> ArcSin[ρ1], β -> ArcSin[ρ1/n]};
```

The differential cross sections for the two polarizations are

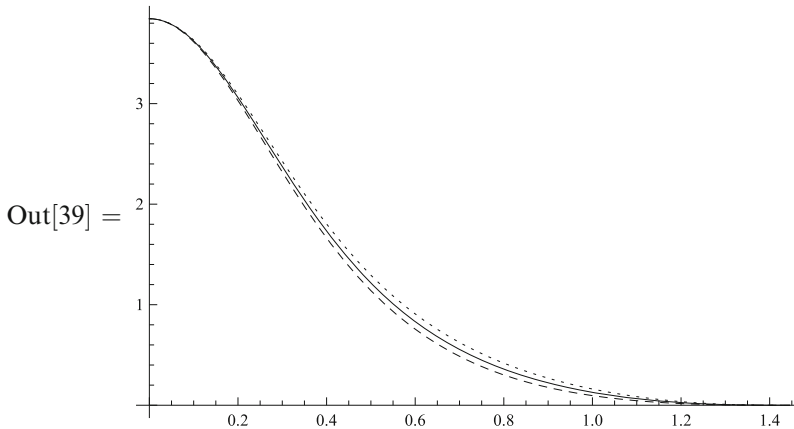
```
In[37] := cs2 = {c2 -> Cos[ϑ/2], s2 -> Sin[ϑ/2]};
```

```
In[38] := σ1s[ϑ_] = Simplify[σ1 * (1 - Rs)^2 /. Snell1 /. cs2 /. Water];
```

```
σ1p[ϑ_] = Simplify[σ1 * (1 - Rp)^2 /. Snell1 /. cs2 /. Water];
```

(the transmission coefficient is $T = 1 - R$, and transmission happens twice).

```
In[39] := Plot[{σ1s[ϑ], σ1p[ϑ], (σ1s[ϑ] + σ1p[ϑ])/2}, {ϑ, 0, ϑ1m /. Water},
PlotRange -> All, PlotStyle -> col]
```



23.4 2 Ray Segments Inside the Drop

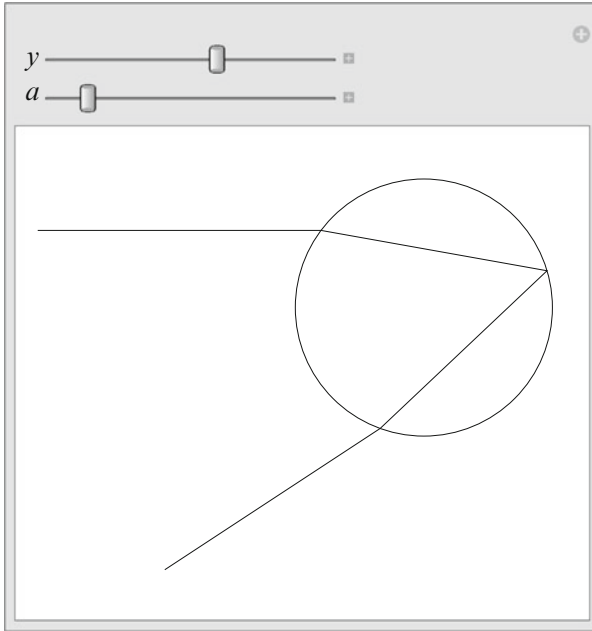
```
In[40] := Ray2[y_., a_] := With[{α = α /. Snell /. ρ -> y,
β = β /. Snell /. ρ -> y /. Water, x = -Sqrt[1 - y^2]},
With[{ϑ = 4 * β - 2 * α - π},
Module[{φ = π - α, R = {col[[1]], Line[{{-1 - a, y}, {x, y}}]},
x1 = x, y1 = y, x2, y2, ψ},
```

```

 $\psi = \varphi + 2 * \beta - \pi$ ; x2 = Cos[ $\psi$ ]; y2 = Sin[ $\psi$ ];
R = Join[R, {col[[3]], Line[{{x1, y1}, {x2, y2}}]}}];
 $\varphi = \psi$ ; x1 = x2; y1 = y2;  $\psi = \varphi + 2 * \beta - \pi$ ; x2 = Cos[ $\psi$ ]; y2 = Sin[ $\psi$ ];
R = Join[R, {col[[4]], Line[{{x1, y1}, {x2, y2}}]}}];
Join[R, {col[[2]], Line[{{x2, y2}, {x2 + a * Cos[ $\vartheta$ ], y2 + a * Sin[ $\vartheta$ ]}]}]}]]]]]]
In[41] := Manipulate[Graphics[Join[{Black, Circle[]}, Ray2[y, a]],
  PlotRange->{{-1 - a, 1.1}, {-1 - 0.7 * a, 1.1}},
  {{y, 0.6}, 0, 1}, {{a, 2}, 1, 10}]

```

Out[41] =



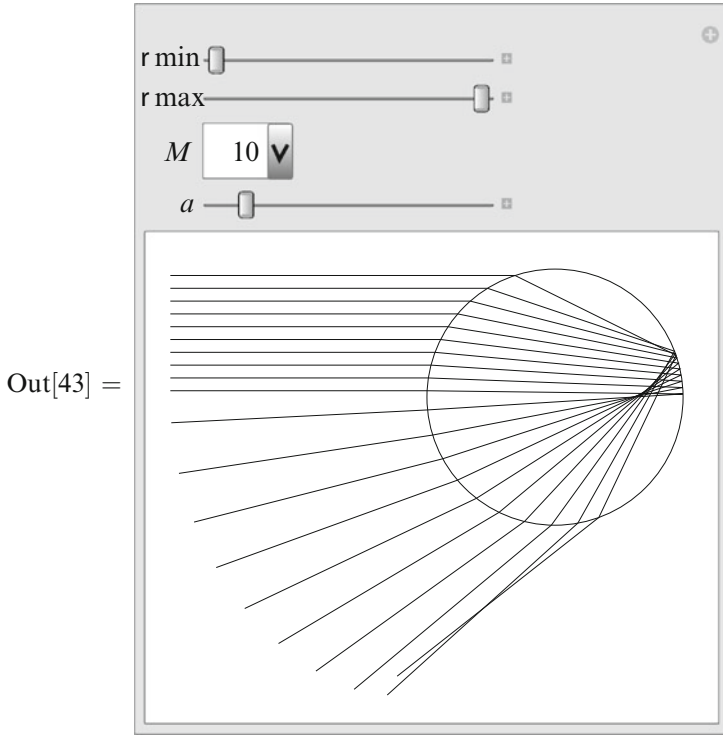
The second segment of the ray inside the drop is obtained from the first one by rotating by the angle $\pi - 2\beta$ clockwise; hence the outgoing ray is obtained from the one in the previous section by the same rotation:

```
In[42] :=  $\vartheta 2 = 4 * \beta - 2 * \alpha - \pi$ ;
```

```

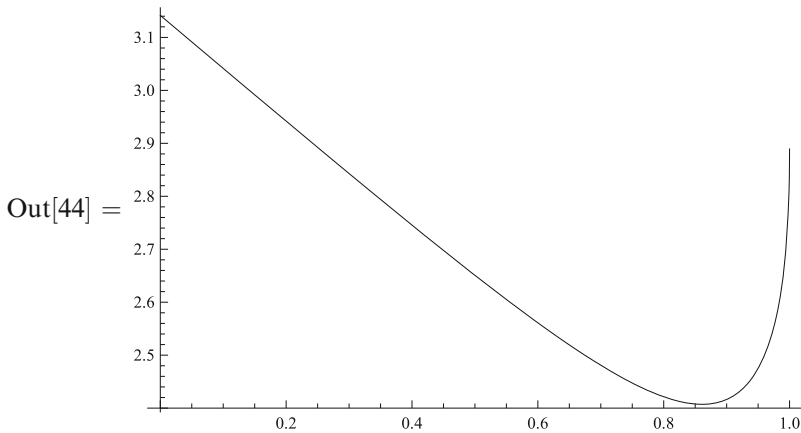
In[43] := Manipulate[
  Graphics[
    Join[{Black, Circle[]},
      With[ $\{\delta = (\rho_{\max} - \rho_{\min}) / M\}$ ,
        If[ $\delta > 0$ , Apply[Join, Table[Ray2[y, a], {y,  $\rho_{\min} + \delta / 2$ ,  $\rho_{\max}$ ,  $\delta$ }], {}]]],
      PlotRange->{{-1 - a, 1.1}, {-1 - 0.7 * a, 1.1}},
      {{ $\rho_{\min}$ , 0}, 0, 1}, {{ $\rho_{\max}$ , 1}, 0, 1},
      {{M, 10}, Table[i, {i, 30}], {{a, 2}, 1, 10}]

```



The real scattering angle is $\vartheta = -\vartheta_2$:

```
In[44] := Plot[-ϑ2/.Snell/.Water, {ρ, 0, 1}]
```



It has a minimum at

```
In[45] := s2r = Solve[D[ϑ2/.Snell, ρ] == 0, ρ]
```

$$\text{Out[45]} = \left\{ \left\{ \rho \rightarrow -\frac{\sqrt{4-n^2}}{\sqrt{3}} \right\}, \left\{ \rho \rightarrow \frac{\sqrt{4-n^2}}{\sqrt{3}} \right\} \right\}$$

In[46] := $\rho 2r = \rho / .s2r[2]$

$$\text{Out[46]} = \frac{\sqrt{4-n^2}}{\sqrt{3}}$$

equal to

In[47] := $\vartheta 2r = -\vartheta 2 / .Snell / .\rho \rightarrow \rho 2r$

$$\text{Out[47]} = \pi + 2 \text{ArcSin} \left[\frac{\sqrt{4-n^2}}{\sqrt{3}} \right] - 4 \text{ArcSin} \left[\frac{\sqrt{4-n^2}}{\sqrt{3}n} \right]$$

In[48] := $\{\rho 2r, \vartheta 2r\} / .Water$

Out[48] = $\{0.860835, 2.40719\}$

Rays from a relatively wide ring around $\rho 2r$ have practically the same scattering angle $\vartheta 2r$. This contribution to the cross section tends to ∞ at this angle. When an observer sees a cloud of water drops illuminated by the sun, especially bright light rays arrive along the cone with angle $\pi - \vartheta 2r$. Usually, only a part of the circle is seen (the full circle can be sometimes observed from an airplane).

In[49] := With[$\{R = -\text{Tan}[\vartheta 2r / .Water]\}$, Graphics3D[

Join[$\{\text{Opacity}[0.1], \text{Yellow}, \text{Cone}[\{\{1, 0, 0\}, \{0, 0, 0\}, R]\}$,

Apply[Join, Table[

Module[$\{x = 0.5 * (\text{Random}[] + 1), \varphi = 2 * \pi * \text{Random}[], y, z\}$,

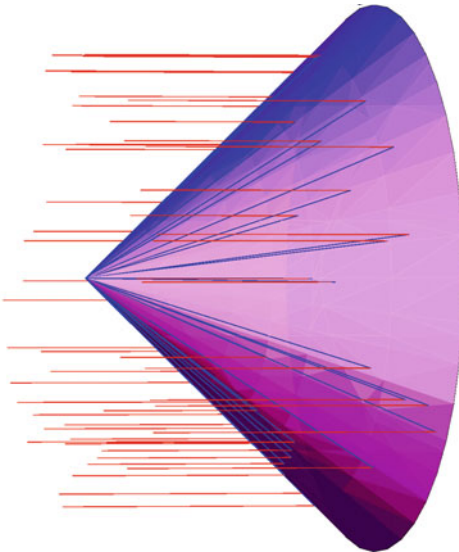
$y = R * x * \text{Cos}[\varphi]; z = R * x * \text{Sin}[\varphi];$

$\{\text{Red}, \text{Line}[\{\{0, y, z\}, \{x, y, z\}\}], \text{Blue}, \text{Line}[\{\{x, y, z\}, \{0, 0, 0\}\}]\}$,

$\{n, 50\}]\}$,

Boxed \rightarrow False, ViewPoint \rightarrow $\{-10, -30, 0\}$]

Out[49] =



In[50] := Show[Import["rainbow.jpg"]]



Out[50] =

The scattering angle at $\rho = 1$ is

In[51] := $\vartheta_{2m} = -\vartheta_{2r} \cdot \text{Snell} / . \rho \rightarrow 1$

Out[51] = $2\pi - 4 \text{ArcSin}\left[\frac{1}{n}\right]$

In[52] := $\{\vartheta_{2rw}, \vartheta_{2mw}\} = \{\vartheta_{2r}, \vartheta_{2m}\} / . \text{Water};$

Now we have to solve the equation $\alpha - 2\beta = (\vartheta - \pi)/2$ for ρ .

In[53] := eq2 = (TrigExpand[Sin[$\alpha - 2 * \beta$]]) / .

$\{\text{Sin}[\alpha] \rightarrow \rho, \text{Cos}[\alpha] \rightarrow \text{Sqrt}[1 - \rho^2],$

$\text{Sin}[\beta] \rightarrow \rho/n, \text{Cos}[\beta] \rightarrow \text{Sqrt}[1 - (\rho/n)^2]\} == -\text{Cos}[\vartheta/2]$

Out[53] = $-\frac{\rho^3}{n^2} - \frac{2\rho\sqrt{1-\rho^2}\sqrt{1-\frac{\rho^2}{n^2}}}{n} + \rho\left(1 - \frac{\rho^2}{n^2}\right) == -\text{Cos}\left[\frac{\vartheta}{2}\right]$

In[54] := sol2 = Solve[eq2, ρ];

The solution number 3 is the smaller one; number 4 is the larger one (1 and 2 are negative).

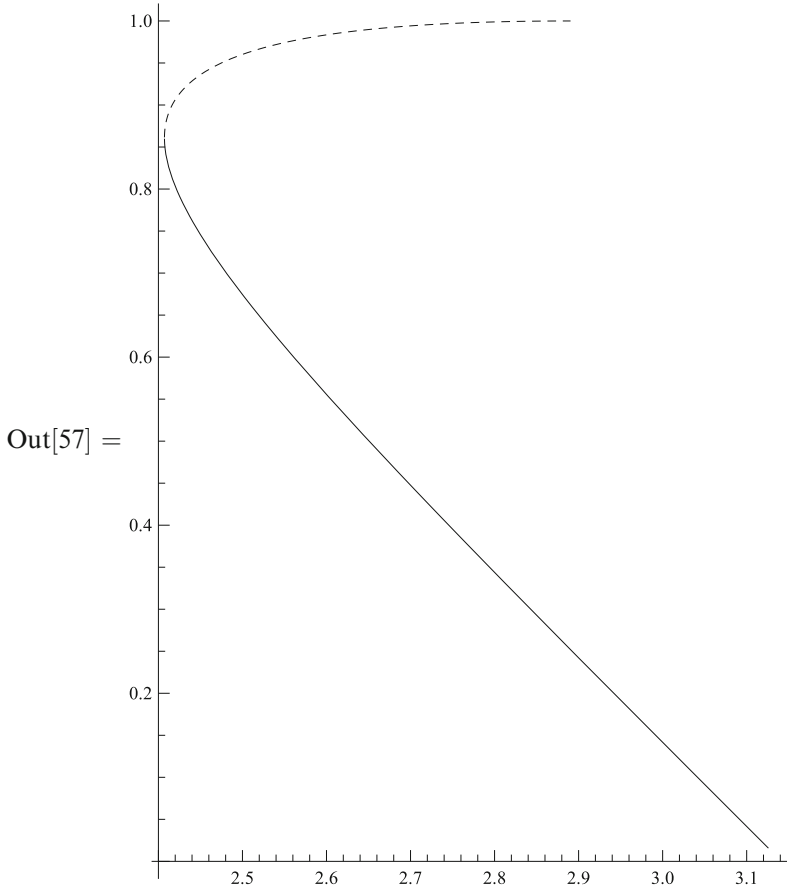
In[55] := $\rho_{2a} = \rho / . \text{sol2}[[3]]$; $\rho_{2b} = \rho / . \text{sol2}[[4]]$;

In[56] := Clear[eq2, sol2]

In[57] := $p_{2a} = \text{ParametricPlot}[\{\vartheta, \rho_{2a} / . \text{Water}\}, \{\vartheta, \vartheta_{2r} / . \text{Water}, \pi\},$
PlotStyle \rightarrow Blue];

$p_{2b} = \text{ParametricPlot}[\{\vartheta, \rho_{2b} / . \text{Water}\}, \{\vartheta, \vartheta_{2r} / . \text{Water}, \vartheta_{2m} / . \text{Water}\},$
PlotStyle \rightarrow Red];

Show[p_{2a}, p_{2b}, PlotRange \rightarrow {{ $\vartheta_{2r} / . \text{Water}, \pi$ }, {0, 1}}]



In order to scatter between ϑ and $\vartheta + d\vartheta$, the incident ray has to hit one of the two rings, σ_{2a} or (if $\vartheta < \vartheta_{2m}$) σ_{2b} .

```
In[58] :=  $\sigma_{2a} = -(D[\rho_{2a}^2, \vartheta] / (4 * c^2 * s^2)) / \{ \text{Cos}[\vartheta/2] \rightarrow c^2, \text{Sin}[\vartheta/2] \rightarrow s^2, \text{Cos}[\vartheta] \rightarrow c^2 - s^2, \text{Sin}[\vartheta] \rightarrow 2 * c^2 * s^2 \};$ 
 $\sigma_{2b} = D[\rho_{2b}^2, \vartheta] / (4 * c^2 * s^2) / \{ \text{Cos}[\vartheta/2] \rightarrow c^2, \text{Sin}[\vartheta/2] \rightarrow s^2, \text{Cos}[\vartheta] \rightarrow c^2 - s^2, \text{Sin}[\vartheta] \rightarrow 2 * c^2 * s^2 \};$ 
```

The angles α and β in these two cases are

```
In[59] :=  $\text{Snell2a} = \{ \alpha \rightarrow \text{ArcSin}[\rho_{2a}], \beta \rightarrow \text{ArcSin}[\rho_{2a}/n] \};$ 
 $\text{Snell2b} = \{ \alpha \rightarrow \text{ArcSin}[\rho_{2b}], \beta \rightarrow \text{ArcSin}[\rho_{2b}/n] \};$ 
```

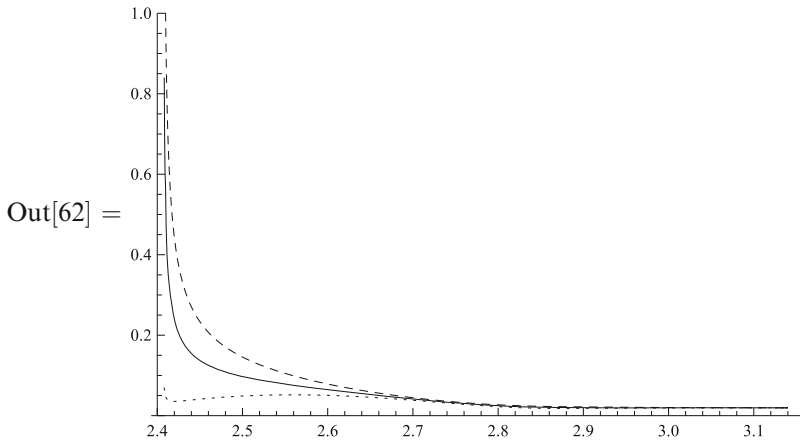
The differential cross sections for the two polarizations are

```
In[60] :=  $\sigma_{2as}[\vartheta] = \sigma_{2a} * (1 - R_s)^2 * R_s / \text{Snell2a} / \text{cs} / \text{Water};$ 
 $\sigma_{2ap}[\vartheta] = \sigma_{2a} * (1 - R_p)^2 * R_p / \text{Snell2a} / \text{cs} / \text{Water};$ 
 $\sigma_{2bs}[\vartheta] = \sigma_{2b} * (1 - R_s)^2 * R_s / \text{Snell2b} / \text{cs} / \text{Water};$ 
 $\sigma_{2bp}[\vartheta] = \sigma_{2b} * (1 - R_p)^2 * R_p / \text{Snell2b} / \text{cs} / \text{Water};$ 
```

```
In[61] :=  $\sigma_{2s}[\vartheta] := \text{If}[\vartheta > \vartheta_{2rw}, \sigma_{2as}[\vartheta] + \text{If}[\vartheta < \vartheta_{2mw}, \sigma_{2bs}[\vartheta], 0], 0];$ 
 $\sigma_{2p}[\vartheta] := \text{If}[\vartheta > \vartheta_{2rw}, \sigma_{2ap}[\vartheta] + \text{If}[\vartheta < \vartheta_{2mw}, \sigma_{2bp}[\vartheta], 0], 0];$ 
```

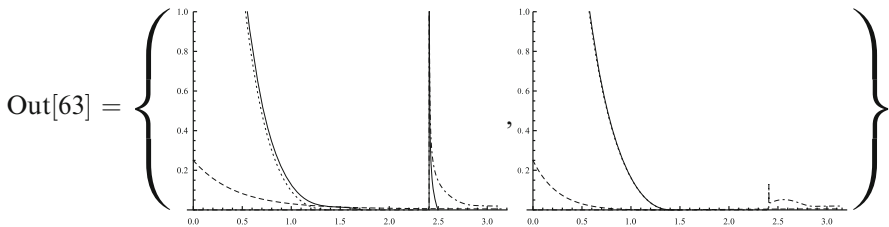


```
In[62] := Plot[{σ2s[ϑ], σ2p[ϑ], (σ2s[ϑ] + σ2p[ϑ])/2}, {ϑ, ϑ2rw + 0.001, π},
PlotRange->{0, 1}, PlotStyle->col]
```



The considered contributions (0, 1, 2 ray segments inside the drop) to the cross sections with the s, p polarizations, as well as their sums, are

```
In[63] := {Plot[{σ0s[ϑ], σ1s[ϑ], σ2s[ϑ], σ0s[ϑ] + σ1s[ϑ] + σ2s[ϑ]}, {ϑ, 0, π},
PlotRange->{0, 1}, PlotStyle->col},
Plot[{σ0p[ϑ], σ1p[ϑ], σ2p[ϑ], σ0p[ϑ] + σ1p[ϑ] + σ2p[ϑ]}, {ϑ, 0, π},
PlotRange->{0, 1}, PlotStyle->col]}
```



Of course, there are also higher contributions, not included here. The cross section is small for scattering angles below the rainbow peak; i.e., the sky just outside the rainbow is darker.

Near the rainbow peak, the last contribution (2 ray segments) is dominant; the ratio of the s and p polarizations is

```
In[64] := Rs2 = Simplify[TrigExpand[Rs/.Snell/.ρ->ρ2r], n > 1];
Rp2 = Simplify[TrigExpand[Rp/.Snell/.ρ->ρ2r], n > 1];
P2 = Simplify[(1 - Rs2)^2 * Rs2 / ((1 - Rp2)^2 * Rp2), n > 1]
```

$$\text{Out[64]} = \frac{(2 + n^2)^6}{729n^4(-2 + n^2)^2}$$

```
In[65] := P2/.Water
```

Out[65] = 25.3347

The rainbow light is highly linearly polarized, with the electric field orthogonal to the scattering plane, i.e., along the rainbow. The reason is that the incidence angle β of the ray which is about to reflect from the inner surface of the drop

In[66] := $\beta / .\text{Snell} / .\rho \rightarrow \rho 2r / .\text{Water}$

Out[66] = 0.702055

is close to the Brewster angle

In[67] := $\beta B / .\text{Water}$

Out[67] = 0.643621

23.5 L Ray Segments Inside the Drop

Repeating the arguments from the previous sections, we obtain the direction of the outgoing ray $\vartheta_L = 2(\beta - \alpha) - (L - 1)(\pi - 2\beta)$:

In[68] := $\vartheta_L = 2 * L * \beta - 2 * \alpha - (L - 1) * \pi$;

**In[69] := $\text{Ray}[L, y, a] := \text{With}[\{\alpha = \alpha / .\text{Snell} / .\rho \rightarrow y,$
 $\beta = \beta / .\text{Snell} / .\rho \rightarrow y / .\text{Water}\},$**

Module[\{ $R = \{\text{col}[[1]], \text{Line}[\{\{-1 - a, y\}, \{-\text{Sqrt}[1 - y^2], y\}\}],$
 $\varphi = \pi - \alpha, \varphi_1, \vartheta = 2 * L * \beta - 2 * \alpha - (L - 1) * \pi\},$

Do[\(\varphi_1 = \varphi + 2 * \beta - \pi;

$R = \text{Join}[R, \{\text{col}[[m + 2]], \text{Line}[\{\{\text{Cos}[\varphi], \text{Sin}[\varphi]\}, \{\text{Cos}[\varphi_1], \text{Sin}[\varphi_1]\}\}]]$;

$\varphi = \varphi_1, \{m, L\}$];

Join[\(R, \{\text{col}[[2]],

$\text{Line}[\{\{\text{Cos}[\varphi], \text{Sin}[\varphi]\}, \{\text{Cos}[\varphi] + a * \text{Cos}[\vartheta], \text{Sin}[\varphi] + a * \text{Sin}[\vartheta]\}\}]]]$

In[70] := $\vartheta[L, \rho] = \vartheta_L / .\text{Snell} / .\text{Water}$;

In[71] := Manipulate [

Manipulate[\

Graphics[\(\text{Join}[\{\text{Black}, \text{Circle}\}], \text{Ray}[L, y, a],

$\text{PlotRange} \rightarrow \{\{-1 - a, 1 + a\}, \{-1 - a, 1 + a\}\}$,

\(\{y, 0.6\}, 0, 1\}, \{a, 2\}, 1, 10\}),

Manipulate[\

Graphics[\(\text{Join}[\{\text{Black}, \text{Circle}\}],

$\text{With}[\{\delta = (\rho_{\text{max}} - \rho_{\text{min}}) / M\}$,

$\text{If}[\delta > 0, \text{Apply}[\text{Join}, \text{Table}[\text{Ray}[L, y, a], \{y, \rho_{\text{min}} + \delta / 2, \rho_{\text{max}}, \delta\}], \{\}]]$,

$\text{PlotRange} \rightarrow \{\{-1 - a, 1 + a\}, \{-1 - a, 1 + a\}\}$,

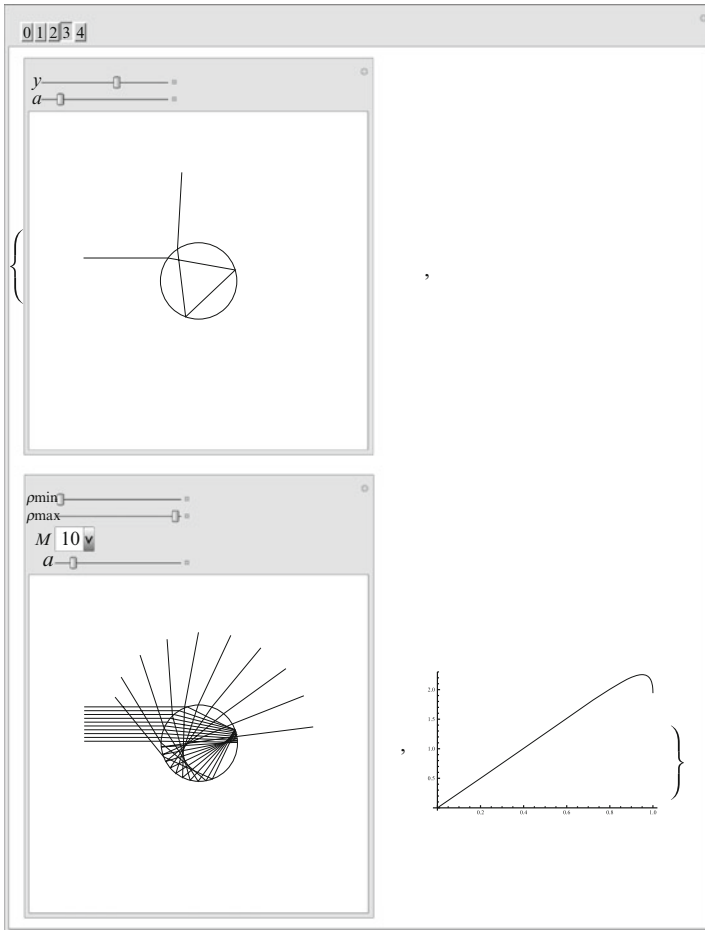
\(\{\rho_{\text{min}}, 0\}, 0, 1\}, \{\rho_{\text{max}}, 1\}, 0, 1\},

\(\{M, 10\}, \text{Table}[i, \{i, 30\}]\}, \{a, 2\}, 1, 10\}),

$\text{Plot}[\text{Abs}[\text{Mod}[\vartheta[L, \rho], 2 * \pi, -\pi]], \{\rho, 0, 1\}]$,

\(\{L, 3\}, \text{Table}[i, \{i, 0, 4\}])]

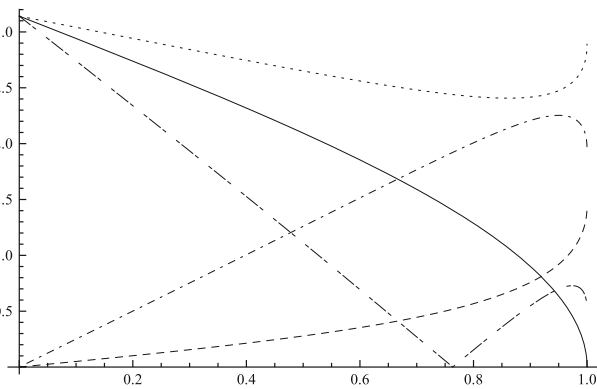
Out[71] =



Scattering angles for $L = 0, 1, 2, 3, 4$ as functions of ρ are

```
In[72] := Plot[Evaluate[Table[Abs[Mod[ϑ[L, ρ], 2 * π, -π]], {L, 0, 4}], {ρ, 0, 1},
PlotStyle -> col]
```

Out[72] =



Naturally, those with even L start from π at $\rho = 0$; for odd L they start from 0. For $L \geq 2$ each one has a single extremum:

In[73] := sol = Solve[D[$\vartheta L / . \text{Snell}, \rho] == 0, \rho]$

$$\text{Out[73]} = \left\{ \left\{ \rho \rightarrow -\frac{\sqrt{L^2 - n^2}}{\sqrt{-1 + L^2}} \right\}, \left\{ \rho \rightarrow \frac{\sqrt{L^2 - n^2}}{\sqrt{-1 + L^2}} \right\} \right\}$$

In[74] := $\rho L = \rho / . \text{sol}[[2]]$

$$\text{Out[74]} = \frac{\sqrt{L^2 - n^2}}{\sqrt{-1 + L^2}}$$

In[75] := $\vartheta r = \vartheta L / . \text{Snell} / . \rho \rightarrow \rho L$

$$\text{Out[75]} = -(-1 + L)\pi - 2 \text{ArcSin} \left[\frac{\sqrt{L^2 - n^2}}{\sqrt{-1 + L^2}} \right] + 2L \text{ArcSin} \left[\frac{\sqrt{L^2 - n^2}}{\sqrt{-1 + L^2}n} \right]$$

These extrema produce rainbows at the angles (in degrees)

In[76] := Table[($\pi - \text{Abs}[\text{Mod}[\vartheta r / . \text{Water}, 2 * \pi, -\pi]]$)/Degree, {L, 2, 4}]

$$\text{Out[76]} = \{42.0781, 50.8908, 138.263\}$$

The angles at which an observer sees the first and the second rainbow are

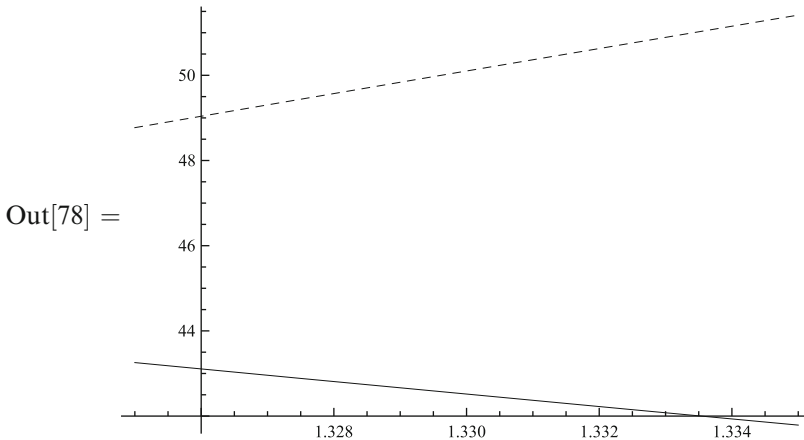
In[77] := { $\vartheta r_2, \vartheta r_3$ } = { $\pi + (\vartheta r / . L \rightarrow 2), -\pi - (\vartheta r / . L \rightarrow 3)$ }

$$\text{Out[77]} = \left\{ -2 \text{ArcSin} \left[\frac{\sqrt{4 - n^2}}{\sqrt{3}} \right] + 4 \text{ArcSin} \left[\frac{\sqrt{4 - n^2}}{\sqrt{3}n} \right], \right. \\ \left. \pi + 2 \text{ArcSin} \left[\frac{\sqrt{9 - n^2}}{2\sqrt{2}} \right] - 6 \text{ArcSin} \left[\frac{\sqrt{9 - n^2}}{2\sqrt{2}n} \right] \right\}$$

The sky is somewhat darker between the first rainbow and the second one, because neither rays with $L = 2$ nor those with $L = 3$ come from these directions.

Until now, we discussed monochromatic light. Then each rainbow is just a bright arc of the same color. In fact, the refraction index of water n depends on the color (wavelength) of light (dispersion). It is larger for violet light than for red one. Therefore the positions of the maxima of intensity of the scattered light also depend on the color.

In[78] := Plot[{ $\vartheta r_2/\text{Degree}, \vartheta r_3/\text{Degree}$ }, {n, 1.325, 1.335}, PlotStyle -> col]



We see that the order of colors in the second rainbow is opposite to that in the first one; and the second rainbow is wider.

The ratios of s and p polarizations at the rainbow peaks are

In[79] := Rs0 = Simplify[TrigExpand[Rs/.Snell/.rho->rhoL], {n > 1, L > 1}]

$$\text{Out[79]} = \frac{(-1+L)^2}{(1+L)^2}$$

In[80] := Rp0 = Simplify[TrigExpand[Rp/.Snell/.rho->rhoL], {n > 1, L > 1}]

$$\text{Out[80]} = \frac{(L-n^2)^2}{(L+n^2)^2}$$

In[81] := P = Simplify[(1-Rs0)^2 * Rs0^(L-1)/((1-Rp0)^2 * Rp0^(L-1)), {n > 1, L > 1}]

$$\text{Out[81]} = \frac{\left(\frac{(-1+L)^2(L+n^2)^2}{(1+L)^2(L-n^2)^2}\right)^L (L^2-n^4)^2}{(-1+L^2)^2 n^4}$$

In[82] := Table[P/.Water, {L, 2, 4}]

$$\text{Out[82]} = \{25.3347, 9.36736, 8.10737\}$$

Higher rainbows are not so strongly polarized as the first one, because the incidence angle for the reflections inside the drop is not so close to the Brewster angle.

Chapter 24

Cyclohexane

24.1 Statement of the Problem

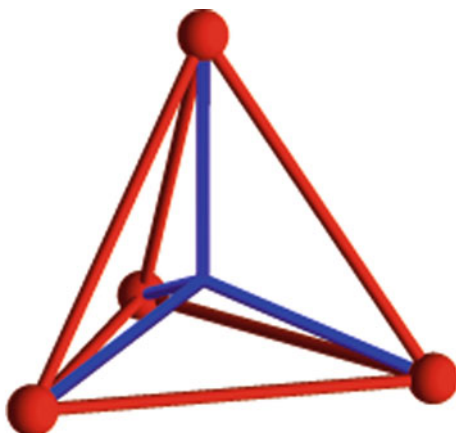
Cyclohexane molecule contains 6 C atoms connected cyclically by single chemical bonds; 2 H atoms are attached to each carbon one. Single bonds of a C atom have a fixed length and form fixed angles: if the C atom is put to the center of a tetrahedron, then its single bonds point to its vertices. The problem is: how many geometrical configurations (conformations, as chemists call them) of the cyclohexane molecule exist—one, several, or infinitely many (and if so, what is the dimensionality of this set).

Tetrahedron

What is the angle between single bonds? The unit vectors $a[1], \dots, a[4]$ (blue) are directed from the center of the tetrahedron (red) to its vertices.

```
In[1] := a[0] = {0, 0, 0}; a[1] = {0, 0, 1}; a[2] = {2 * Sqrt[2], 0, -1}/3;
a[3] = {-Sqrt[2], Sqrt[6], -1}/3; a[4] = {-Sqrt[2], -Sqrt[6], -1}/3;
In[2] := rc = 0.025; rs = 0.1;
In[3] := Graphics3D[{Blue, Cylinder[{a[0], a[1]}, rc], Cylinder[{a[0], a[2]}, rc],
Cylinder[{a[0], a[3]}, rc], Cylinder[{a[0], a[4]}, rc],
Red, Cylinder[{a[1], a[2]}, rc], Cylinder[{a[1], a[3]}, rc],
Cylinder[{a[1], a[4]}, rc], Cylinder[{a[2], a[3]}, rc],
Cylinder[{a[2], a[4]}, rc], Cylinder[{a[3], a[4]}, rc],
Sphere[a[1], rs], Sphere[a[2], rs], Sphere[a[3], rs], Sphere[a[4], rs]},
Boxed->False, ViewPoint->{10, 10, 4}]
```

Out[3] =



Let's check: the vectors are indeed unit and form equal angles with each other; hence all edges have equal lengths, so this is indeed a tetrahedron.

```
In[4] := MatrixForm[Table[a[i].a[j], {i, 1, 4}, {j, 1, 4}]]
```

```
Out[4] // MatrixForm =
```

$$\begin{pmatrix} 1 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & 1 & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 1 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 1 \end{pmatrix}$$

Cosine of the angle between these vectors is $-1/3$.

```
In[5] := Clear[a, rc, rs]
```

24.2 First Steps

Thus we have 6 vectors $a[1], \dots, a[6]$ drawn from each C atom to the next one. They form a closed hexagon: $a[1] + \dots + a[6] = 0$. Let the length of a single C–C bond be 1, then all the vectors are unit. Scalar products of neighboring vectors are $1/3$ (the sign has changed because now one of the vectors points in, not out). In order not to deal with the overall orientation of the molecule, let's consider invariant quantities—cosines of the angles $c[i, j] = a[i].a[j]$.

```
In[6] := c[i_-, j_-] /; i > j := c[j, i]
```

```
In[7] := Do[c[i, i] = 1, {i, 1, 6}]
```

```
In[8] := Do[c[i, i + 1] = 1/3, {i, 1, 5}]; c[1, 6] = 1/3;
```

In[9] := MatrixForm[M = Array[c, {6, 6}]]

Out[9]//MatrixForm =

$$\begin{pmatrix} 1 & \frac{1}{3} & c[1,3] & c[1,4] & c[1,5] & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & c[2,4] & c[2,5] & c[2,6] \\ c[1,3] & \frac{1}{3} & 1 & \frac{1}{3} & c[3,5] & c[3,6] \\ c[1,4] & c[2,4] & \frac{1}{3} & 1 & \frac{1}{3} & c[4,6] \\ c[1,5] & c[2,5] & c[3,5] & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & c[2,6] & c[3,6] & c[4,6] & \frac{1}{3} & 1 \end{pmatrix}$$

Linear Equations

Multiplying the vector equality $a[1] + \dots + a[6] = 0$ by each vector $a[i]$, we get 6 linear equations for $c[i, j]$.

In[10] := Eq = Table[Sum[c[i, j], {j, 1, 6}] == 0, {i, 1, 6}]

$$\text{Out[10]} = \left\{ \begin{aligned} \frac{5}{3} + c[1,3] + c[1,4] + c[1,5] &== 0, \frac{5}{3} + c[2,4] + c[2,5] + c[2,6] == 0, \\ \frac{5}{3} + c[1,3] + c[3,5] + c[3,6] &== 0, \frac{5}{3} + c[1,4] + c[2,4] + c[4,6] == 0, \\ \frac{5}{3} + c[1,5] + c[2,5] + c[3,5] &== 0, \frac{5}{3} + c[2,6] + c[3,6] + c[4,6] == 0 \end{aligned} \right\}$$

Let's take $x = c[1, 3]$, $y = c[3, 5]$, $z = c[5, 1]$ as independent variables and express the remaining ones via them.

In[11] := c[1, 3] = x; c[3, 5] = y; c[1, 5] = z;

In[12] := s = Solve[Eq, {c[1, 4], c[2, 4], c[2, 5], c[2, 6], c[3, 6], c[4, 6]}][[1]]

$$\text{Out[12]} = \left\{ \begin{aligned} c[1, 4] \rightarrow -\frac{5}{3} - x - z, c[2, 4] \rightarrow z, c[2, 5] \rightarrow -\frac{5}{3} - y - z, c[2, 6] \rightarrow y, \\ c[3, 6] \rightarrow -\frac{5}{3} - x - y, c[4, 6] \rightarrow x \end{aligned} \right\}$$

In[13] := c[1, 4] = c[1, 4]/.s; c[2, 4] = c[2, 4]/.s; c[2, 5] = c[2, 5]/.s;

c[2, 6] = c[2, 6]/.s; c[3, 6] = c[3, 6]/.s; c[4, 6] = c[4, 6]/.s;

In[14] := Clear[s]

In[15] := MatrixForm[M = M]

Out[15]//MatrixForm =

$$\begin{pmatrix} 1 & \frac{1}{3} & x & -\frac{5}{3} - x - z & z & \frac{1}{3} \\ \frac{1}{3} & 1 & \frac{1}{3} & z & -\frac{5}{3} - y - z & y \\ x & \frac{1}{3} & 1 & \frac{1}{3} & y & -\frac{5}{3} - x - y \\ -\frac{5}{3} - x - z & z & \frac{1}{3} & 1 & \frac{1}{3} & x \\ z & -\frac{5}{3} - y - z & y & \frac{1}{3} & 1 & \frac{1}{3} \\ \frac{1}{3} & y & -\frac{5}{3} - x - y & x & \frac{1}{3} & 1 \end{pmatrix}$$

24.3 Equations

Generating Combinations

Now let's recall that our vectors $a[1], \dots, a[6]$ live in 3-dimensional space. Any 4 of them are linearly dependent. First we have to find a way to generate all 4-element lists made of the numbers from 1 to 6 in the increasing order. We shall accumulate them in the list L . The main work is done by the recursive function Gen . Its parameters: l —part of the list which has been already constructed; n —how many elements are to be added; a and b —boundaries of the interval from which numbers can be taken. If everything has been done ($n = 0$), the constructed list l is appended to the list of results L . Otherwise, we add each number i from the allowed interval to l and call Gen recursively to add $n - 1$ numbers. The upper limit of the loop is determined by the requirement to have at least $n - 1$ numbers in the interval from $i + 1$ to b .

In[16] := $L = \{\}$;

In[17] := $\text{Gen}[l _, n _, a _, b _] := \text{If}[n \leq 0, L = \text{Append}[L, l],$

Do[$\text{Gen}[\text{Append}[l, i], n - 1, i + 1, b], \{i, a, b - n + 1\}]$

In[18] := $\text{Gen}[\{\}, 4, 1, 6]; L$

Out[18] = $\{\{1, 2, 3, 4\}, \{1, 2, 3, 5\}, \{1, 2, 3, 6\}, \{1, 2, 4, 5\}, \{1, 2, 4, 6\}, \{1, 2, 5, 6\},$
 $\{1, 3, 4, 5\}, \{1, 3, 4, 6\}, \{1, 3, 5, 6\}, \{1, 4, 5, 6\}, \{2, 3, 4, 5\}, \{2, 3, 4, 6\},$
 $\{2, 3, 5, 6\}, \{2, 4, 5, 6\}, \{3, 4, 5, 6\}\}$

It works. There are 15 4-combinations of 6 numbers.

Nonlinear Equations

For each set of 4 vectors, the determinant of the corresponding 4×4 submatrix of the matrix M (it is the square of the 4-dimensional volume spanned by these vectors) should be equal to 0.

In[19] := $\text{Eq} = \text{Table}[\text{Det}[M[[l, l]]], \{l, L\}]$

Out[19] = $\left\{ \begin{array}{l} \frac{5}{3} - \frac{34x}{9} - \frac{23x^2}{9} - \frac{34z}{9} - \frac{20xz}{9} + \frac{2x^2z}{3} - \frac{23z^2}{9} + \frac{2xz^2}{3} + x^2z^2, \\ -2 + \frac{2x}{9} + \frac{16x^2}{9} - \frac{40y}{9} + \frac{10xy}{9} + \frac{10x^2y}{3} - \frac{23y^2}{9} + \frac{2xy^2}{3} + x^2y^2 - \frac{40z}{9} + \\ \frac{10xz}{9} + \frac{10x^2z}{3} - \frac{32yz}{9} + \frac{10xyz}{3} + 2x^2yz - \frac{23z^2}{9} + \frac{2xz^2}{3} + x^2z^2, \\ -\frac{5}{3} - \frac{34x}{9} - \frac{23x^2}{9} - \frac{34y}{9} - \frac{20xy}{9} + \frac{2x^2y}{3} - \frac{23y^2}{9} + \frac{2xy^2}{3} + x^2y^2, \\ \frac{7}{3} + \frac{50x}{9} + \frac{16x^2}{9} + \frac{50y}{9} + \frac{98xy}{9} + \frac{10x^2y}{3} + \frac{16y^2}{9} + \frac{10xy^2}{3} + x^2y^2 + \frac{20z}{3} + \\ \frac{118xz}{9} + \frac{10x^2z}{3} + \frac{118yz}{9} + \frac{40xyz}{3} + 2x^2yz + \frac{10y^2z}{3} + 2xy^2z + 4z^2 + \end{array} \right.$

$$\begin{aligned}
& \frac{20xz^2}{3} + x^2z^2 + \frac{20yz^2}{3} + 2xyz^2 + y^2z^2, \\
-2 & - \frac{40x}{9} - \frac{23x^2}{9} + \frac{2y}{9} + \frac{10xy}{9} + \frac{2x^2y}{3} + \frac{16y^2}{9} + \frac{10xy^2}{3} + x^2y^2 - \frac{40z}{9} - \\
& \frac{32xz}{9} + \frac{10yz}{9} + \frac{10xyz}{3} + \frac{10y^2z}{3} + 2xy^2z - \frac{23z^2}{9} + \frac{2yz^2}{3} + y^2z^2, \\
- \frac{5}{3} & - \frac{34x}{9} - \frac{23y^2}{9} - \frac{34z}{9} - \frac{20yz}{9} + \frac{2y^2z}{3} - \frac{23z^2}{9} + \frac{2yz^2}{3} + y^2z^2, \\
-2 & - \frac{40x}{9} - \frac{23x^2}{9} + \frac{2y}{9} + \frac{10xy}{9} + \frac{2x^2y}{3} + \frac{16y^2}{9} + \frac{10xy^2}{3} + x^2y^2 - \frac{40z}{9} - \\
& \frac{32xz}{9} + \frac{10yz}{9} + \frac{10xyz}{3} + \frac{10y^2z}{3} + 2xy^2z - \frac{23z^2}{9} + \frac{2yz^2}{3} + y^2z^2, \\
\frac{7}{3} & + \frac{20x}{3} + 4x^2 + \frac{50y}{9} + \frac{118xy}{9} + \frac{20x^2y}{3} + \frac{16y^2}{9} + \frac{10xy^2}{3} + x^2y^2 + \frac{50z}{9} + \\
& \frac{118xz}{9} + \frac{20x^2z}{3} + \frac{98yz}{9} + \frac{40xyz}{3} + 2x^2yz + \frac{10y^2z}{3} + 2xy^2z + \frac{16z^2}{9} + \\
& \frac{10xz^2}{3} + x^2z^2 + \frac{10yz^2}{3} + 2xyz^2 + y^2z^2, \\
-2 & - \frac{40x}{9} - \frac{23x^2}{9} - \frac{40y}{9} - \frac{32xy}{9} - \frac{23y^2}{9} + \frac{2z}{9} + \frac{10xz}{9} + \frac{2x^2z}{3} + \frac{10yz}{9} + \\
& \frac{10xyz}{3} + \frac{2y^2z}{3} + \frac{16z^2}{9} + \frac{10xz^2}{3} + x^2z^2 + \frac{10yz^2}{3} + 2xyz^2 + y^2z^2, \\
- \frac{5}{3} & - \frac{34x}{9} - \frac{23x^2}{9} - \frac{34z}{9} - \frac{20xz}{9} + \frac{2x^2z}{3} - \frac{23z^2}{9} + \frac{2xz^2}{3} + x^2z^2, \\
- \frac{5}{3} & - \frac{34y}{9} - \frac{23y^2}{9} - \frac{34z}{9} - \frac{20yz}{9} + \frac{2y^2z}{3} - \frac{23z^2}{9} + \frac{2yz^2}{3} + y^2z^2, \\
-2 & - \frac{40x}{9} - \frac{23x^2}{9} - \frac{40y}{9} - \frac{32xy}{9} - \frac{23y^2}{9} + \frac{2z}{9} + \frac{10xz}{9} + \frac{2x^2z}{3} + \frac{10yz}{9} + \\
& \frac{10xyz}{3} + \frac{2y^2z}{3} + \frac{16z^2}{9} + \frac{10xz^2}{3} + x^2z^2 + \frac{10yz^2}{3} + 2xyz^2 + y^2z^2, \\
\frac{7}{3} & + \frac{50x}{9} + \frac{16x^2}{9} + \frac{20y}{3} + \frac{118xy}{9} + \frac{10x^2y}{3} + 4y^2 + \frac{20xy^2}{3} + x^2y^2 + \frac{50z}{9} + \\
& \frac{98xz}{9} + \frac{10x^2z}{3} + \frac{118yz}{9} + \frac{40xyz}{3} + 2x^2yz + \frac{20y^2z}{3} + 2xy^2z + \frac{16z^2}{9} + \\
& \frac{10xz^2}{3} + x^2z^2 + \frac{10yz^2}{3} + 2xyz^2 + y^2z^2, \\
-2 & + \frac{2x}{9} + \frac{16x^2}{9} - \frac{40y}{9} + \frac{10xy}{9} + \frac{10x^2y}{3} - \frac{23y^2}{9} + \frac{2xy^2}{3} + x^2y^2 - \frac{40z}{9} + \\
& \frac{10xz}{9} + \frac{10x^2z}{3} - \frac{32yz}{9} + \frac{10xyz}{3} + 2x^2yz - \frac{23z^2}{9} + \frac{2xz^2}{3} + x^2z^2, \\
- \frac{5}{3} & - \frac{34x}{9} - \frac{23x^2}{9} - \frac{34y}{9} - \frac{20xy}{9} + \frac{2x^2y}{3} - \frac{23y^2}{9} + \frac{2xy^2}{3} + x^2y^2 \}
\end{aligned}$$

These 15 polynomials of 3 variables must be equal to 0.

In[20] := Clear[L]

Gröbner Basis

Let's find a simpler set of polynomials having the same solution set—the Gröbner basis.

In[21] := GB = GroebnerBasis[Eq, {z, y, x}]

Out[21] = $\{-15 - 34x - 23x^2 - 34y - 20xy + 6x^2y - 23y^2 + 6xy^2 + 9x^2y^2,$
 $-102 - 400x - 284x^2 + 18x^4 - 69y - 212xy + 18x^2y + 108x^3y + 27x^4y -$
 $69z - 212xz + 18x^2z + 108x^3z + 27x^4z,$
 $18 + 54x + 18x^2 - 6x^3 + 21y + 41xy - 9x^2y - 9x^3y + 21z + 41xz -$
 $9x^2z - 9x^3z + 20yz,$
 $-15 - 34x - 23x^2 - 34z - 20xz + 6x^2z - 23z^2 + 6xz^2 + 9x^2z^2\}$

Do they factorize?

In[22] := GB = Map[Factor, GB]

Out[22] = $\{-15 - 34x - 23x^2 - 34y - 20xy + 6x^2y - 23y^2 + 6xy^2 + 9x^2y^2,$
 $(3 + x)(1 + 3x)(-34 - 20x + 6x^2 - 23y + 6xy + 9x^2y - 23z + 6xz + 9x^2z),$
 $18 + 54x + 18x^2 - 6x^3 + 21y + 41xy - 9x^2y - 9x^3y + 21z + 41xz - 9x^2z -$
 $9x^3z + 20yz,$
 $-15 - 34x - 23x^2 - 34z - 20xz + 6x^2z - 23z^2 + 6xz^2 + 9x^2z^2\}$

In[23] := p1 = GB[[1]]; p2 = GB[[2]]/(3 + x)/(1 + 3 * x);

p3 = GB[[3]]; p4 = GB[[4]];

24.4 Projection onto the x, y Plane

Allowed Region

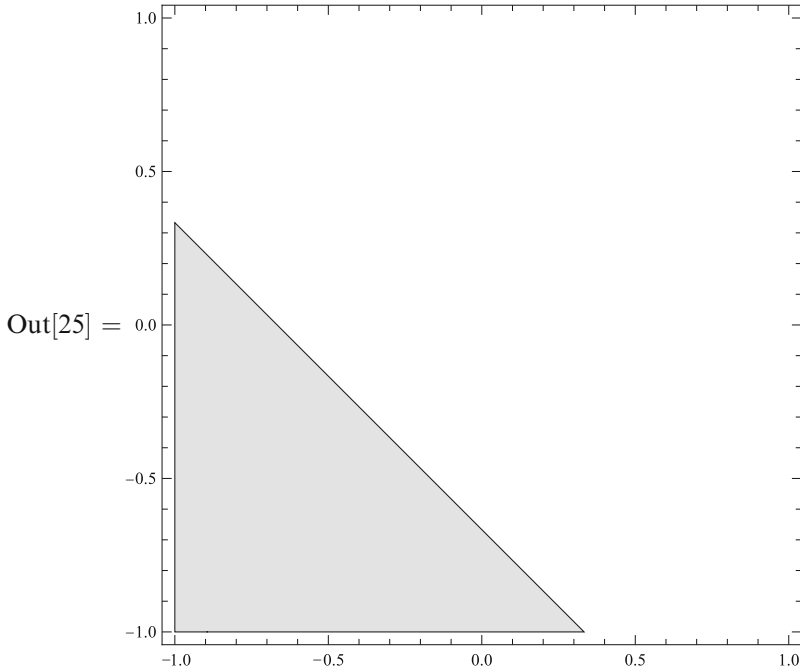
First let's find the projection of the solution set onto the x, y plane. What part of this plane are we interested in? First, x and y should lie between -1 and 1; second,

In[24] := c[3, 6]

Out[24] = $-\frac{5}{3} - x - y$

should also lie between -1 and 1.

In[25] := RegionPlot[-1 ≤ x ≤ 1 && -1 ≤ y ≤ 1 && -1 ≤ c[3, 6] ≤ 1,
 $\{x, -1, 1\}, \{y, -1, 1\}$



That is, the allowed region is the triangle with the vertices $(-1, -1)$, $(-1, 1/3)$, and $(1/3, -1)$.

Solutions with $x = -1/3$

The second equation is satisfied at $x = -1/3$. What about the other ones?

In[26] := Eq = {p1, p3, p4} /. x -> -1/3

$$\text{Out[26]} = \left\{ -\frac{56}{9} - \frac{80y}{3} - 24y^2, \frac{20}{9} + \frac{20y}{3} + \frac{20z}{3} + 20yz, -\frac{56}{9} - \frac{80z}{3} - 24z^2 \right\}$$

In[27] := GB = GroebnerBasis[Eq, {z, y}]

$$\text{Out[27]} = \{7 + 30y + 27y^2, 1 + 3y + 3z + 9yz, 7 + 30z + 27z^2\}$$

In[28] := GB = Map[Factor, GB]

$$\text{Out[28]} = \{(1 + 3y)(7 + 9y), (1 + 3y)(1 + 3z), (1 + 3z)(7 + 9z)\}$$

So, we have found the solutions $x = y = z = -1/3$; $x = y = -1/3, z = -7/9$; and $x = z = -1/3, y = -7/9$.

In[29] := Eq /. {y -> -1/3, z -> -1/3}

$$\text{Out[29]} = \{0, 0, 0\}$$

In[30] := Eq /. {y -> -1/3, z -> -7/9}

$$\text{Out[30]} = \{0, 0, 0\}$$

In[31] := Eq/.{y->-7/9,z->-1/3}

Out[31] = {0,0,0}

It is clear from the symmetry argument that $y = z = -1/3$, $x = -7/9$ is also a solution.

In[32] := {p1,p2,p3,p4}/.{x->-7/9,y->-1/3,z->-1/3}

Out[32] = {0,0,0,0}

Other Solutions

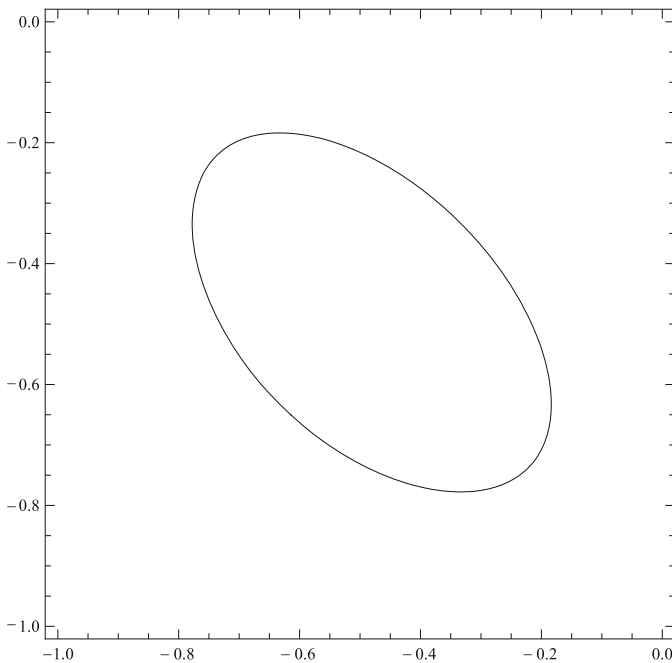
The first equation contains only x and y .

In[33] := p1

Out[33] = $-15 - 34x - 23x^2 - 34y - 20xy + 6x^2y - 23y^2 + 6xy^2 + 9x^2y^2$

In[34] := P1 = ContourPlot[p1 == 0, {x, -1, 0}, {y, -1, 0}]

Out[34] =



This equation is quadratic in y .

In[35] := Do[c[i] = Coefficient[p1,y,i],{i,0,2}]

Does $c[2]$ vanish somewhere?

In[36] := s = Solve[c[2] == 0,x]

Out[36] = $\left\{ \left\{ x \rightarrow \frac{1}{3} (-1 - 2\sqrt{6}) \right\}, \left\{ x \rightarrow \frac{1}{3} (-1 + 2\sqrt{6}) \right\} \right\}$

In[37] := N[x/.s]

Out[37] = $\{-1.96633, 1.29966\}$

In our region $c[2] > 0$. The discriminant:

In[38] := d = Factor[c[1]^2 - 4 * c[0] * c[2]]

Out[38] = $32(-1+x)(7+9x)(1+6x+3x^2)$

In[39] := s = Solve[d == 0, x]

Out[39] = $\left\{ \left\{ x \rightarrow -\frac{7}{9} \right\}, \{x \rightarrow 1\}, \left\{ x \rightarrow \frac{1}{3}(-3 - \sqrt{6}) \right\}, \left\{ x \rightarrow \frac{1}{3}(-3 + \sqrt{6}) \right\} \right\}$

In[40] := N[x/.s]

Out[40] = $\{-0.777778, 1, -1.8165, -0.183503\}$

The discriminant is positive between $x = -7/9$ and

In[41] := xmax = x/.s[[4]]

Out[41] = $\frac{1}{3}(-3 + \sqrt{6})$

In[42] := Clear[s]

Two values of y (with $pm = \pm 1$) correspond to each x from this interval.

In[43] := y1 = (-c[1] + pm * Sqrt[d]) / (2 * c[2])

Out[43] = $\left(34 + 20x - 6x^2 + 4\sqrt{2}pm\sqrt{(-1+x)(7+9x)(1+6x+3x^2)} \right) / (2(-23+6x+9x^2))$

In[44] := Clear[d]

Do the points we found earlier lie on this curve?

In[45] := {p1/.{x->-1/3,y->-1/3}, p1/.{x->-1/3,y->-7/9}, p1/.{x->-7/9,y->-1/3}}

Out[45] = $\{0, 0, 0\}$

Yes, they do. Let's denote these points A, B, C.

In[46] := pA = {-1/3, -1/3}; pB = {-1/3, -7/9}; pC = {-7/9, -1/3};

So, all solutions we are interested in project onto this curve in the x, y plane.

We shall need a few extra points on this curve. Let's find the second intersection with the diagonal $x = y$ (the first one is the point A).

In[47] := s = Solve[(p1/.y->x) == 0, x]

Out[47] = $\left\{ \{x \rightarrow -3\}, \left\{ x \rightarrow -\frac{1}{3} \right\}, \left\{ x \rightarrow \frac{1}{3}(3 - 2\sqrt{6}) \right\}, \left\{ x \rightarrow \frac{1}{3}(3 + 2\sqrt{6}) \right\} \right\}$

In[48] := N[x/.s]

Out[48] = $\{-3., -0.333333, -0.632993, 2.63299\}$

In[49] := x0 = x/.s[[3]]

Out[49] = $\frac{1}{3}(3 - 2\sqrt{6})$

Let's call this point D.

In[50] := pD = {x0, x0};

In[51] := Clear[s, x0]

Finally, let's introduce some additional point between A and B (rather arbitrarily; e.g., let it have $x = -1/4$) and call it E. Let its mirror image be the point F.

```
In[52] := x0 = -1/4;
```

```
In[53] := y0 = Simplify[y1/. {x->x0, pm->1}]
```

```
Out[53] =  $\frac{1}{383} (-229 - 10\sqrt{38})$ 
```

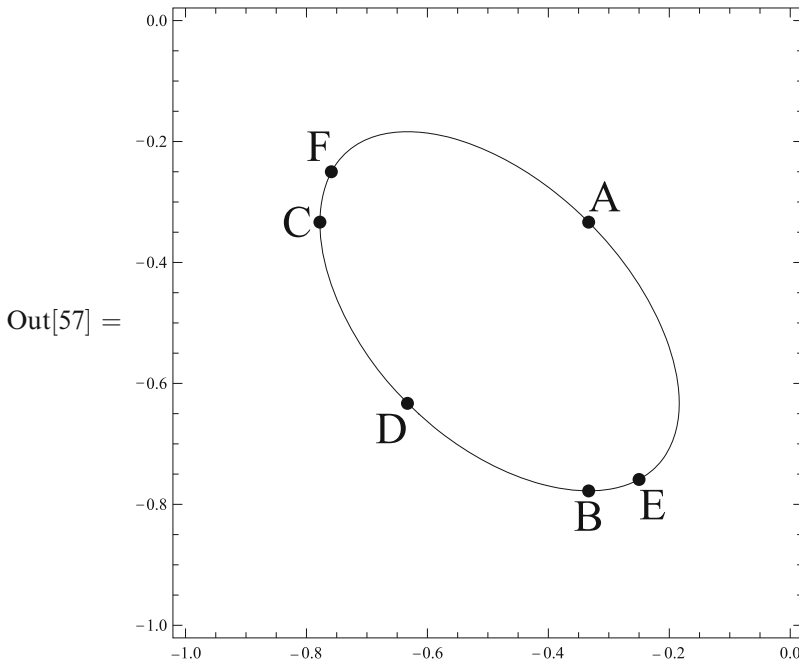
```
In[54] := pE = {x0, y0}; pF = {y0, x0};
```

```
In[55] := Clear[x0, y0]
```

This is shown in the plot.

```
In[56] := P2 = Graphics[{PointSize[Large],
  Red, Point[pA], Text[Style[A, Large], pA, {-1, -1}],
  Point[pB], Text[Style[B, Large], pB, {0, 1}],
  Point[pC], Text[Style[C, Large], pC, {1, 0}],
  Darker[Green], Point[pD], Text[Style[D, Large], pD, {1, 1}],
  Point[pE], Text[Style[E, Large], pE, {-1, 1}],
  Point[pF], Text[Style[F, Large], pF, {1, -1}]}];
```

```
In[57] := Show[P1, P2]
```

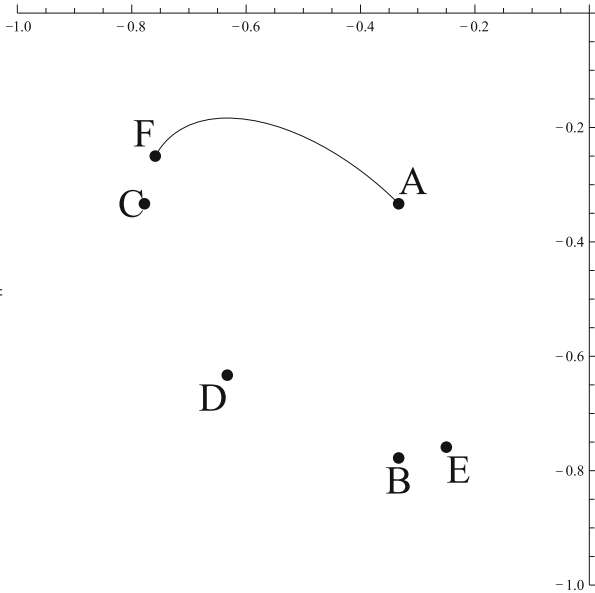


What's the reason for introducing the points D, E, F? Now it is easy to write down our curve parametrically. For $t \in [0, 1]$ let the point move from F to A, the motion along x being uniform.

```
In[58] := pFA[t_] := With[{xt = (1 - t) * pF[[1]] + t * pA[[1]]},
  {xt, y1/. {x->xt, pm->1}}]
```

```
In[59] := P1 = ParametricPlot[pFA[t], {t, 0, 1}, PlotRange->{{-1, 0}, {-1, 0}}];
```

In[60] := Show[P1,P2]



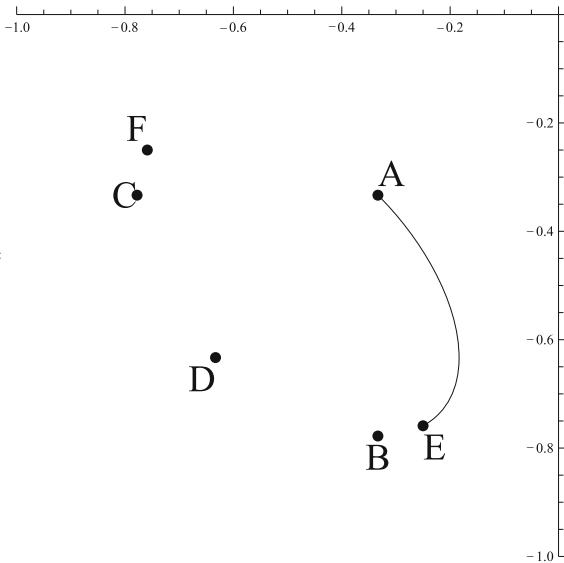
Out[60] =

For $t \in [1,2]$ let the point move from A to E; this segment is mirror-symmetric to the previous one.

**In[61] := pAE[t_] := With[{y0 = (2 - t) * pA[[2]] + (t - 1) * pE[[2]]},
 {y1/. {x->y0, pm->-1}, y0}]**

In[62] := P1 = ParametricPlot[pAE[t], {t, 1, 2}, PlotRange->{{-1, 0}, {-1, 0}}];

In[63] := Show[P1,P2]



Out[63] =

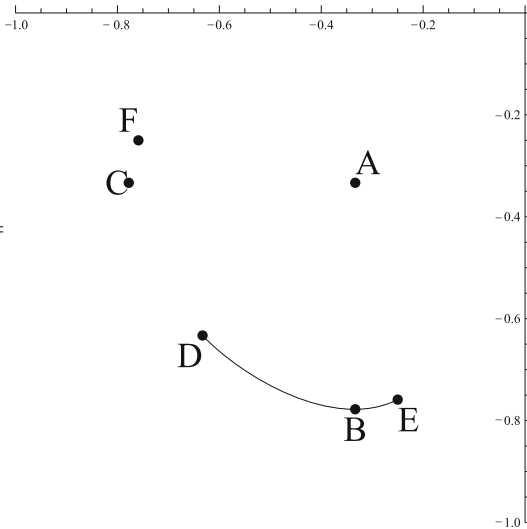
For $t \in [2, 3]$ the point moves from E to D, the motion along x being uniform.

```
In[64] := pED[t_] := With[{x0 = (3 - t) * pE[[1]] + (t - 2) * pD[[1]]},
  {x0, y1 /. {x -> x0, pm -> 1}}]
```

```
In[65] := P1 = ParametricPlot[pED[t], {t, 2, 3}, PlotRange -> {{-1, 0}, {-1, 0}}];
```

```
In[66] := Show[P1, P2]
```

Out[66] =



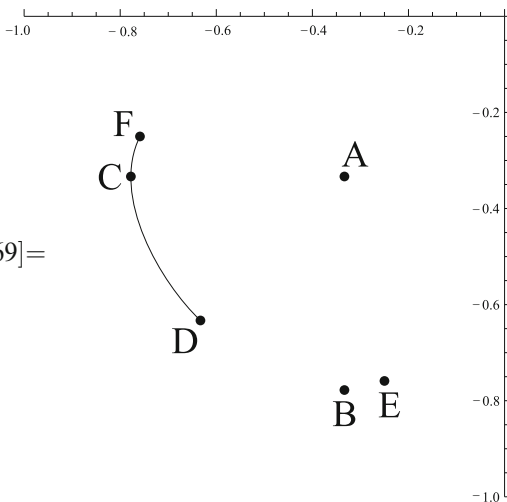
Finally, for $t \in [3, 4]$ the point moves from D to F; this segment is mirror-symmetric to the previous one.

```
In[67] := pDF[t_] := With[{y0 = (4 - t) * pD[[2]] + (t - 3) * pF[[2]]},
  {y1 /. {x -> y0, pm -> 1}, y0}]
```

```
In[68] := P1 = ParametricPlot[pDF[t], {t, 3, 4}, PlotRange -> {{-1, 0}, {-1, 0}}];
```

```
In[69] := Show[P1, P2]
```

Out[69] =



Later we shall join these segments and construct a parametric curve in the 3-dimensional space x, y, z .

24.5 Complete Analysis of the Solutions

How to find the value (or values) of z corresponding to some point x, y on our curve? It is easiest to use the second equation—it is linear in z .

In[70] := p2

Out[70] = $-34 - 20x + 6x^2 - 23y + 6xy + 9x^2y - 23z + 6xz + 9x^2z$

In[71] := Do[c[i] = Coefficient[p2, z, i], {i, 0, 1}]

Does $c[1]$ vanish somewhere in our region?

In[72] := s = Solve[c[1] == 0, x]

Out[72] = $\left\{ \left\{ x \rightarrow \frac{1}{3} (-1 - 2\sqrt{6}) \right\}, \left\{ x \rightarrow \frac{1}{3} (-1 + 2\sqrt{6}) \right\} \right\}$

In[73] := N[x/.s]

Out[73] = $\{-1.96633, 1.29966\}$

No, it does not.

In[74] := Clear[s]

So there is a single solution:

In[75] := z1 = -c[0]/c[1]

Out[75] = $\frac{34 + 20x - 6x^2 + 23y - 6xy - 9x^2y}{-23 + 6x + 9x^2}$

And what about the third and fourth equations?

In[76] := p3 = Numerator[Together[p3/.z->z1]]

Out[76] = $-20(-15 - 34x - 23x^2 - 34y - 20xy + 6x^2y - 23y^2 + 6xy^2 + 9x^2y^2)$

In[77] := p4 = Numerator[Together[p4/.z->z1]]

Out[77] = $-15 - 34x - 23x^2 - 34y - 20xy + 6x^2y - 23y^2 + 6xy^2 + 9x^2y^2$

In[78] := Cancel[p3/p1]

Out[78] = -20

In[79] := Cancel[p4/p1]

Out[79] = 1

They are satisfied automatically. What z corresponds to $x = y = -1/3$?

In[80] := z1 /. {x->-1/3, y->-1/3}

Out[80] = $-\frac{7}{9}$

So, one of the solutions found earlier, namely $x = y = z = -1/3$, does not belong to our one-dimensional family of solutions. To summarize: we have found one isolated solution plus a one-dimensional family of solutions. In the parametric form:

**In[81] := xyz[t_] := With[{xy = Which[t < 1, pFA[t], t < 2, pAE[t],
t < 3, pED[t], True, pDF[t]]},**

{xy[[1]], xy[[2]], z1 /. {x->xy[[1]], y->xy[[2]]}}]

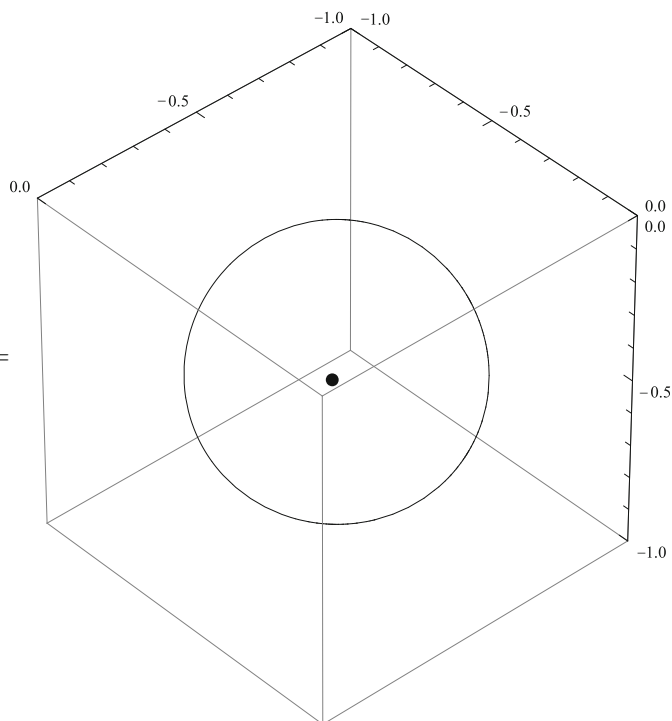
In[82] := P1 = ParametricPlot3D[xyz[t], {t, 0, 4},

PlotRange->{{-1, 0}, {-1, 0}, {-1, 0}}, ViewPoint->{10, 11, 12}];

In[83] := p0 = {-1/3, -1/3, -1/3};

```
In[84] := P2 = Graphics3D[{Darker[Green], PointSize[Large], Point[p0]};
In[85] := Show[P1, P2]
```

Out[85] =



You can rotate this plot with your mouse to understand it better.

24.6 Shape of the Molecule

What does the cyclohexane molecule look like? Let's direct the x -axis along $a[1]$:

```
In[86] := a[1] = {1, 0, 0}
```

```
Out[86] = {1, 0, 0}
```

Let $a[2]$ lie in the x, y plane:

```
In[87] := a[2] = {1/3, 2* Sqrt[2]/3, 0}
```

```
Out[87] = { 1/3, 2* Sqrt[2]/3, 0 }
```

That is, the unit vector along y is a combination of $a[1]$ and $a[2]$:

```
In[88] := (3* a[2] - a[1])/(2* Sqrt[2])
```

```
Out[88] = {0, 1, 0}
```

The projections of $a[3]$ onto x and y are $c[1,3] = x$ and

```
In[89] := (3* c[2,3] - c[1,3])/(2* Sqrt[2])
```

```
Out[89] = (1 - x) / (2* Sqrt[2])
```

The projection of $a[3]$ onto the z axis can be found from normalization:

```
In[90] := a[3] = {x, (1 - x)/(2 * Sqrt[2]),
  pm * Sqrt[(1 - x) * (7 + 9 * x)]/(2 * Sqrt[2])}
```

```
Out[90] = {x, 1 - x / (2 * Sqrt[2]), pm * Sqrt[(1 - x) * (7 + 9 * x)] / (2 * Sqrt[2])}
```

where $pm = \pm 1$. Two molecule shapes correspond to a single set of values of x, y, z ; they differ by the mirror reflection of the z coordinates. We shall discuss this matter in a moment.

```
In[91] := Table[Expand[a[i].a[3]]/.pm^2->1, {i, 1, 3}]
```

```
Out[91] = {x, 1/3, 1}
```

That is, the unit vector along the z axis is a combination of $a[1], a[2], a[3]$:

```
In[92] := Simplify[2 * Sqrt[2]/Sqrt[(1 - x) * (7 + 9 * x)] *
  (a[3] - 3/8 * (1 - x) * a[2] + (1 - 9 * x)/8 * a[1])]
```

```
Out[92] = {0, 0, pm}
```

The rest is easy.

```
In[93] := Do[Print[a[i] = Simplify[{c[1, i], (3 * c[2, i] - c[1, i])/(2 * Sqrt[2]),
  2 * Sqrt[2] * pm/Sqrt[(1 - x) * (7 + 9 * x)] *
  (c[3, i] - 3/8 * (1 - x) * c[2, i] + (1 - 9 * x)/8 * c[1, i])}],
```

```
{i, 4, 6}]
{ -5/3 - x - z, 5/3 + x + 4z, pm(1 + 9x^2 - 4z + 2x(7 + 6z)) / (2 * Sqrt[2] * Sqrt[7 + 2x - 9x^2]) }
{ z, -5 + 3y + 4z, -pm(-5 - 11y - 4z + x(5 + 3y + 12z)) / (2 * Sqrt[2] * Sqrt[7 + 2x - 9x^2]) }
{ 1/3, -1 + 9y, pm(-13 - 11y + x(-11 + 3y)) / (6 * Sqrt[2] * Sqrt[7 + 2x - 9x^2]) }
```

Let's write a function which constructs the molecule for a given values of x, y, z and of the sign pm .

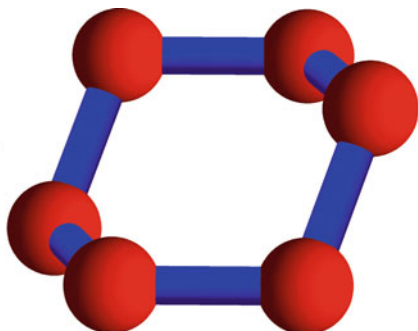
```
In[94] := rc = 0.1; rs = 0.25;
```

```
In[95] := Molecule[xyz_., s_] := Module[{r = {0, 0, 0}, r2, l = {Blue},
  S = {x->xyz[[1]], y->xyz[[2]], z->xyz[[3]], pm->s}},
  Do[r2 = r + (a[i]/.S); l = Append[l, Cylinder[{r, r2}, rc]]; r = r2, {i, 1, 6}];
  r = {0, 0, 0}; l = Append[l, Red];
  Do[r2 = r + (a[i]/.S); l = Append[l, Sphere[r, rs]]; r = r2, {i, 1, 6}];
  Graphics3D[l]]
```

This is the isolated conformation of the cyclohexane molecule with $x = y = z = -1/3$. Use your mouse to understand it better.

```
In[96] := Show[Molecule[p0, 1], ViewPoint->{15, -5, 5}, Boxed->False]
```

Out[96] =



And this is the one-parameter family of conformations. To ensure smooth dependence on t , it is necessary to flip the sign pm when passing through the point C (where the expression under the radical sign vanishes). This happens at

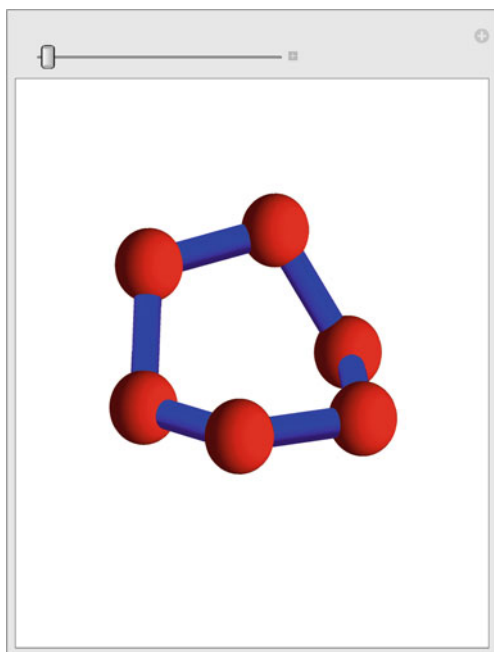
In[97] := $t_0 = (621 - 8 * \text{Sqrt}[6])/159$

Out[97] = $\frac{1}{159} (621 - 8\sqrt{6})$

So, the molecule returns to its initial shape after we traverse the loop in the x , y , z space twice. You can see this conformation family especially clearly if you start animation.

**In[98] := Manipulate[Show[
 Molecule[If[$t > 4$, xyz[$t - 4$], xyz[t]], If[$t > t_0 \&\& t < t_0 + 4$, -1, +1]],
 PlotRange -> {{-0.7, 1.7}, {-0.5, 1.9}, {-1.7, 1.7}},
 ViewPoint -> {10, -10, 4}, Boxed -> False],
 { t , 0, 8}]**

Out[98] =



Chapter 25

Problems for Students

1. Write a procedure which returns the hydrogen wave function (in spherical coordinates, i. e., an expression containing r, θ, ϕ) for given quantum numbers n, l, m . Write a procedure to calculate the rate of the electric dipole transition [22] from the state n, l, m to the state n', l', m' .
2. Calculate Poisson brackets of the Hamiltonian, the angular momentum components, and the Runge–Lenz vector components [19] for a particle in the Coulomb field $U = -a/r$. Calculate commutators of the same quantities in quantum mechanics [18].
3. The hypergeometric function [23, 24, 27] is defined as the sum of the series

$$F(a, b, c, x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n}{(c)_n} \frac{x^n}{n!},$$

where $(x)_n = x(x+1)\cdots(x+n-1)$ is the Pochhammer symbol. In many cases it can be expressed via simpler functions. Write a list of substitutions for simplifying hypergeometric functions. It is sufficient to consider only simplifications valid for an arbitrary x (not for specific values) where results are expressed via elementary functions. More general substitutions should be near the beginning of the list, then their particular cases can be eliminated.

4. Consider indefinite integrals of the form

$$\int A(x) \log B(x) dx,$$

where $A(x)$ and $B(x)$ are rational functions of x . *Mathematica* can calculate such integrals, but often produces results in which some terms have imaginary parts in the region of x we are interested in. It is not easy to trace their cancellations. We'll suppose that $A(x)$ and $B(x)$ contain no parameters (except x), only numbers. We'll also suppose that *Mathematica* is able to find all roots of the denominator of $A(x)$, as well as of the numerator and the denominator of $B(x)$, and all these roots are real.

We are interested in a neighborhood of some point x_0 ; we want to get a result all terms of which are real near this point (if this is possible, of course). Implement the following obvious approach:

- Expand $A(x)$ into partial fractions with respect to x .
- Replace $\log B(x)$ by a combination of terms $\log(x - a_i)$ and $\log(a_i - x)$ (plus a constant) in such a way that they are all real near x_0 .
- Multiply.
- Take integrals of $x^n \log(x - a)$ ($n \geq 0$), $\log(x - a)/(x - b)^n$ ($n \geq 2$) by parts to eliminate the logarithm. Don't use the *Mathematica* integrator—it can produce $\log(x - a)$ where $\log(a - x)$ is needed.
- We are left with the most difficult terms of the forms $\log(x - a)/(x - b)$ and $\log(a - x)/(x - b)$. By linear substitutions they reduce to 3 cases:

$$\int \frac{\log(y + 1)}{y} dy = -\text{Li}_2(-y),$$

$$\int \frac{\log(y - 1)}{y} dy = \log(y) \log(y - 1) + \text{Li}_2(1 - y),$$

$$\int \frac{\log(1 - y)}{y} dy = -\text{Li}_2(y),$$

where y is positive near $x = x_0$ (the third formula is the definition of $\text{Li}_2(y)$; the first one follows from it using the substitution $y \rightarrow -y$; the second one—using integration by parts).

The result must be real ($\log(x)$ is real at $x > 0$; $\text{Li}_2(x)$ —at $x < 1$). If this is impossible, print an error message.

5. Implement the algebra of Boolean expressions. They consist of the constants true and false, variables, the function not (one argument), and the functions and, or (an arbitrary number of arguments). The last two functions are commutative and associative. Take into account simplifications when one of the arguments is true or false; when two arguments coincide or equal to a and not[a]. Expressions should be reduced to the disjunctive normal form: “or” at the top level; its arguments can be “and”; their arguments can be “not” or variables.

6. Implement the algebra of quaternions.

7. Implement Dirac γ -matrix expressions, including trace calculations (in 4 dimensions [22] or in the general case of dimensional regularization, see, e.g., [24]). Pay no attention to efficiency.

8. Implement calculation of color factors of Feynman diagrams for the color group $SU(N_c)$ using the Cvitanović algorithm [27] (see also [24]).

9. Write a procedure to calculate two-loop massless propagator diagrams using integration by parts (see, e.g., [24]). Results should be linear combinations of the two basis integrals.

10. Hypergeometric functions whose argument is 1 and whose parameters contain a small parameters ε and tend to integers at $\varepsilon \rightarrow 0$ can be expanded in series in ε . The algorithm is described, e.g., in [24]; implement it.

11. Any polynomial over the field of complex numbers can be factorized into linear factors:

$$p(x) = \prod (x - a_i)^{d_i},$$

where a_i are its roots and d_i are their multiplicities (to simplify formulas, we have assumed that the leading coefficient is 1). Let's group factors with equal d_i :

$$p(x) = \prod p_i^{d_i},$$

where all d_i are distinct and the polynomials $p_i(x)$ have only simple zeros (are square-free). This square-free factorization can be obtained by a simple algorithm which uses only gcd (this is much simpler than the full factorization). Namely,

$$\gcd(p, p') = \prod p_i^{d_i-1}.$$

Indeed, the polynomial $p(x)$ has zero of the order d_i at $x \rightarrow a_i$, and its derivative $p'(x)$ has zero of the order $d_i - 1$. Write a function to calculate square-free factorization using only gcd.

References

1. Buchberger, B., Collins, G.E., Loos, R. (ed.): *Computer Algebra: Symbolic and Algebraic Computation*, 2nd edn. Springer, Vienna (1983)
2. Davenport, J.H., Siret, Y., Tournier, E.: *Computer Algebra: Systems and Algorithms for Algebraic Computation*, 2nd edn. Academic Press, London (1993)
3. Geddes, K.O., Czapor, S.R., Labahn, G.: *Algorithms for Computer Algebra*. Kluwer Academic Publishers, Boston (1992)
4. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*, 2nd edn. Cambridge University Press, Cambridge (2003)
5. Grozin, A.G.: *Using REDUCE in High Energy Physics*. Cambridge University Press, New York (1997); paperback edition (2005)
6. Wolfram, S.: *The Mathematica Book*, 5th edn. Wolfram Media, Champaign (2003)
7. Trott, M.: *The Mathematica GuideBook for Symbolics*. Springer Science+Business Media, Inc., New York (2006)
8. Trott, M.: *The Mathematica GuideBook for Numerics*. Springer Science+Business Media, Inc., New York (2006)
9. Trott, M.: *The Mathematica GuideBook for Graphics*. Springer, New York (2004)
10. Trott, M.: *The Mathematica GuideBook for Programming*. Springer, New York (2004)
11. Mangano, S.: *Mathematica Cookbook*, O'Reilly Media, Inc., Sebastopol, CA (2010)
12. Cox, D., Little, J., O'Shea, D.: *Ideals, Varieties, and Algorithms*, 3rd edn. Springer, New York (2007)
13. Arzhantsev, I.V.: *Gröbner Bases and Systems of Algebraic Equations (in Russian)*, 3rd edn. MCCMO, Moscow (2003). <http://www.mccme.ru/free-books/dubna/arjantsev.pdf>
14. Kredel, H., Weispfenning, V.: J. Symbolic Computing dimension and independent sets for polynomial ideals **6**, 231 (1988)
15. Fateman, R.J.: A review of *Mathematica*. J. Symbolic Comput. **13**, 545 (1992). (<http://www.cs.berkeley.edu/~fateman/papers/mma.pdf>); <http://www.cs.berkeley.edu/~fateman/papers/mma6rev.pdf>
16. Bronstein, M.: *Symbolic Integration I*. Springer, Berlin (1997)
17. Davenport, J.H.: On the integration of algebraic functions. In: *Lecture notes in computer science*, vol. 102. Springer, New York (1981)
18. Landau, L.D., Lifshitz, E.M.: *Quantum Mechanics: Non-relativistic Theory*, 3rd edn. Butterworth-Heinemann, Oxford (1981)
19. Landau, L.D., Lifshitz, E.M.: *Mechanics*, 3rd edn. Butterworth-Heinemann, Oxford (1982)
20. Landau, L.D., Lifshitz, E.M.: *The Classical Theory of Fields*, 4th edn. Butterworth-Heinemann, Oxford (1980)
21. Landau, L.D., Lifshitz, E.M., Pitaevskii, L.P.: *Electrodynamics of Continuous Media*, 2nd edn. Butterworth-Heinemann, Oxford (1995)

22. Berestetskii, V.B., Lifshitz, E.M., Pitaevskii, L.P.: Quantum Electrodynamics, 2nd edn. Butterworth-Heinemann, Oxford (1982)
23. Prudnikov, A.P., Brychkov, Yu.A., Marichev, O.I.: Integrals and Series, vol. 3, Chapter 7. Gordon and Breach, New York (1990)
24. NIST Handbook of Mathematical Functions, ed. by F.W.J. Olver, D.W. Lozier, R.F. Boisvert, C.W. Clark, Cambridge University Press, Cambridge (2010). <http://dlmf.nist.gov/>
25. <http://functions.wolfram.com/>
26. Grozin, A.G.: Lectures on QED and QCD: Practical calculation and renormalization of one- and multi-loop Feynman diagrams. World Scientific (2007)
27. Cvitanović, P.: Group Theory. Princeton University Press, Princeton (2008). <http://www.nbi.dk/GroupTheory/>

Index

Symbols

"", 22, 43–46, 77, 99–104, 106–107, 130, 131, 136, 141, 151, 161–162, 166, 169, 173–174, 177–178, 185
!=, 45, 47
., 93–97, 194
/., 17–19, 27–28, 38, 40
//., 28–33, 167–170, 177
/;, 41–42
/;;, 32–33, 39, 103, 135, 194
::, 9, 107–108, 161
:=, 35–41, 43, 44, 46, 48, 51–52, 68, 74, 81–89, 100–101, 107, 120, 125, 126, 128, 129, 134, 135, 139, 154, 156, 159, 161, 162, 167–170, 174, 178, 181, 186, 188, 194, 196, 202–205, 207
: >, 35–36, 167–169, 171
;;, 9, 43–44
:::, 23–24, 76
?, 37–38, 41, 108
[[]], 14, (23, 18–24, 57, 58, 60, 61, 74, 76, 78, 80, 82, 83, 85–89, 92, 95, 196
[], 9, 12–13, 22
%, 12, 57, 58, 67, 68
&&, 118, 123, 198, 208
→, 27–28, 45–46, 103
..?, 32
.., 30–32, 146, 148, 150
--, 33, 103
---, 33, 40, 103, 167–170
`, 24–25, 37–38, 41, 73–75, 105–108
~~, 103–104
{}, 14
*~, 47
++, 47, 92
+~, 92, 156, 159
-~, 141

→, 17–19, 27–28, 35–36, 65
<<, 99–101, 106, 108, 158
<>, 102
=!=, 45, 161–162
=, 9, 22, 35–39, 51–52, 92–93, 95–96
==, 18, 33, 44–45, 60, 70–71, 76, 78, 80, 82–89, 116, 123, 129, 134–135, 147–150, 156, 159, 168, 170–171, 177, 180, 183, 185, 190, 195, 200, 201, 205
===, 45, 162
>>, 100
^:=, 41
^=, 42
||, 33
|, 97, 103
\$Assumptions, 69, 97–98
\$Context, 105–107
\$ContextPath, 105–107
\$KernelID, 50–51
\$Path, 100, 106
\$RecursionLimit, 9

A

Abort, 161
Abs, 126, 156, 180, 188–190
Accuracy, 74
AccuracyGoal, 77
All, 92, 120, 177, 181
Antisymmetric, 97
Apart, 12
Appearance, 11, 130, 131, 136
Append, 100, 168, 196, 207
Apply, 48, 175, 179, 182, 184, 188
ArcSin, 69, 173, 177, 180, 181, 184–186, 190
ArcTan, 66, 177
Array, 91–92, 195
Arrays, 97–98

Assumptions, 69, 135

AtomQ, 24

Attributes, 40–41, 52–53

Automatic, 75

AxesLabel, 155

B

Begin, 106–108

BeginPackage, 107–108

BesselI, 69

Binomial, 70

Blank, 32–33, 103

Block, 49

Boxed, 184, 207, 208

C

Cancel, 12, 88, 131, 205

CForm, 102

Circle, 173–175, 179, 182, 188

Clear, 9, 12, 14, 18, 19, 22, 24–26, 28, 31–33, 35–44, 46–53, 57, 59–61, 64–67, 74, 76, 78, 80, 82–84, 86–89, 93–95, 97, 100–103, 105, 107, 118, 121, 122, 129, 135, 140, 147, 149, 159, 168, 170, 177, 181, 185, 194, 195, 197, 201, 202, 205

ClearAll, 41, 52

Close, 101

Coefficient, 11, 80, 82–89, 146–150, 200, 205

Collect, 11, 156, 159

Complex, 24

ComplexExpand, 60

ComplexInfinity, 76

ComposeSeries, 65–66

CompoundExpression, 43

Condition, 32–33

ConditionalExpression, 69

Cone, 184

Conjugate, 137

Context, 105–107

ContourPlot, 116, 200

ContourPlot3D, 123

Contours, 116

Cos, 12–13, (64, (67, 15–67, 71, 78, 112, 113, 121, 122, 125–126, 135–136, 146–151, 163, 173–174, 176–177, 179–182, 184–186, 188

Cot, 63, 134, 163

Cross, 97

Csc, 69, 71

Cycles, 97–98

Cylinder, 193, 207

D

D, 13–14, 38, 65, 67–68, 80–81, 83–89, 128, 134, 146–151, 158, 161, 162, 177, 181, 183, 186, 190

Darker, 202, 206

Dashed, 113

DeclarePackage, 106

Degree, 190

DegreeLexicographic, 59, 61–62

Denominator, 25, 126

Derivative, 67–68

Det, 94–95, 196

Dimensions, 93

DirectedInfinity, 64

Disk, 139

DistributeDefinitions, 50–51

Do, 46, 63, 74, 95, 126, 141, 148, 150, 154, 156, 159, (162, (162, 161–162, 188, 194, 196, 200, 205, 207

DSolve, 70–71, 78, 129, 134–135

E

E, 10, 12, 68, 70–71

Eigensystem, 95–97

Eigenvalues, 95

Eigenvectors, 95

Element, 95, 97–98

EllipticF, 69

End, 106–108

EndPackage, 107–108

Erfi, 69

EulerSum, 106

Evaluate, 52–53, 66, 130, 163, 176, 180, 189

EvenQ, 126, 150

Exp, 12–13, 15, 63, 65–69, (134, 76–135

Expand, 10–11, 35–36, 38, 41, 57, 58, 60, 67, 68, 84, 87, 107, 135, 141, 157, 167, 169, 207

ExpandAll, 82, 88, 89, 147–151

Exponent, 11

Extension, 11, 12

F

Factor, 11–12, 61, 135, 156, 198–199, 201

Factorial, 10

False, 24–25, 39, 45, 93, 104, 184, 193, 207, 208

FilePrint, 99–102, 107

FindIntegerNullVector, 77

First, 18, 47

Fit, 117

Flat, 40

For, 47

FortranForm, 102

- FractionalPart, 125
 - FreeQ, 45
 - FullForm, 24–26, 28–33, 35, 43, 63–64, 67–68, 73–74, 102–103
 - Function, 47–49, 51, 91–92, 141
- G**
- Gamma, 69, 135
 - Get, 100
 - Global, 37, 38, 105–107
 - Graphics, 139–140, 173–176, 178–179, 182, 188, 202
 - Graphics3D, 184, 193, 205–207
 - Greater, 33
 - GroebnerBasis, 59–62, 198–199
- H**
- Head, 22–24, 27, 48
 - Hold, 9, 25–26, 29, 43, 53
 - HoldAll, 52–53
 - HoldFirst, 52
 - Hypergeometric2F1Regularized, 69
- I**
- I, 11–12, 21, 24, 131, 134–135
 - If, 44, 126, 139, 141, 150, 154, 156, 159, 161–162, 168, 175, 179, 182, 186, 188, 196, 208
 - Im, 25
 - Import, 166, 169, 185
 - Indeterminate, 76
 - Infinity, 14, 64, 67, 70, 77, 129, 131
 - Inset, 173–174, 178
 - Integer, 24, 27, 32–33, 125
 - IntegerQ, 25
 - Integrate, 14, 51, 65, 68–69, 129, 131, 135, 137
 - InterpolatingFunction, 78
 - Inverse, 94, 96, 161
 - InverseSeries, 65–66, 158–159
- J**
- Join, 100, 139, 168, 174–176, 179, 181–182, 184, 188
 - JordanDecomposition, 96
- K**
- KroneckerDelta, 70
- L**
- Length, 22, 161
 - Line, 139, 173–174, 178–179, 181–182, 184, 188
 - LinearSolve, 94
 - List, 25, 28, 32, 50, 63–64
 - Listable, 41, 50
 - ListPlot, (117, 75–117
 - Log, 12–14, 64, 68–69, 77, 80, 83–84, 86–88
 - LogIntegral, 68
 - LogLogPlot, 114–115
 - LogPlot, 114
- M**
- MachineNumberQ, 73
 - MachinePrecision, 73
 - Manipulate, (15, (16, (121, 11–122, 130–131, 136, 174–176, 179, 182, 188, 208
 - Map, 48–49, 75, 76, 146–151, 167–171, 198, 199
 - MatchQ, 45
 - Matrices, 97
 - MatrixForm, 91–96, 194–195
 - MatrixPower, 94
 - MatrixQ, 93
 - MatrixRank, 95
 - Max, 156
 - Message, 161
 - Method, 77
 - Min, 156
 - Mod, 135, 188–190
 - Module, 49, 74, 126, 141, 161, 162, 167–170, 181, 184, 188, 207
- N**
- N, 10, 73–74, 117, 126, 201, 205
 - ND, 106
 - NDSolve, 78
 - NIntegrate, 77
 - NLimit, 106
 - Normal, 66, 151
 - Not, 39, 45
 - NResidue, 106
 - NSeries, 106
 - NSolve, 76
 - NSum, 77
 - NSumTerms, 77
 - Null, 43
 - NullSpace, 95
 - NumberQ, 32, 45
 - Numerator, 25, 205
 - NumericalCalculus, 106
 - NumericQ, 45
- O**
- OddQ, 150
 - Opacity, 184
 - OpenWrite, 101
 - Options, 40, 162
 - OptionsPattern, 40, 161

OptionValue, 40, 162
 OrderedQ, 39
 Orderless, 40–41
 OutputStream, 101

P

Parallelize, 50–51, 131
 ParametricPlot, (115, 15–115, 154–156,
 185–186, 202–204
 ParametricPlot3D, (121, 16–122, 205–206
 Part, 22–23
 Pattern, 32–33, 103
 Pi, 10, 12, 16–17, 45, 51, 53, 66, 69, (125,
 76–126, 131, 135, 137
 Plot, (52, 15–53, 66, (78, (111, 76–114, 117,
 130–131, 176–178, 180–181, 183,
 186–190
 Plot3D, 15–16, 101, 120
 PlotMarkers, 75
 PlotPoints, 123
 PlotRange, 15, 16, 75, 114, 120, 122, 130, 131,
 155, 174–179, 181, 182, 185, 187–188,
 202–205, 208
 PlotStyle, 112–113, 177, 181, 185, 187, 189,
 190
 Plus, 25–27, 31–33, 40, 41, 46, 48, 67
 Point, 202, 205–206
 PointSize, 202, 205–206
 PolyLog, 68
 PolynomialReduce, 57–59
 Power, 25–26, 28–31, 64, 67, 102
 Precision, 73–76
 PrecisionGoal, 77
 Prepend, 66, 100
 Print, 43, 44, 46–47, 50, 63, 95, 101, 141, 148,
 156, 159, 162, 207
 Put, 100

Q

Quaternion, 106
 Quaternions, 106
 Quotient, 135

R

Random, 184
 Rational, 24–25, 28, 63–64, 125
 Re, 25
 Real, 24
 Reals, 76, 95, 97–98
 RegionPlot, 118, 198
 RegionPlot3D, 123
 ReleaseHold, 53
 Remove, 105
 ReplaceAll, 29

ReplaceRepeated, 29
 Rest, 47
 Rule, 29, 32–33

S

Save, 101, 151
 Select, 48
 Series, 14, 63–66, 145–150, 157–159
 SeriesCoefficient, 63, 66–67, 146, 148, 150
 SeriesData, 63–64
 Set, 25–26, 52
 SetDelayed, 35, 52
 Show, 117–118, 140, 166, 169, 185–186,
 202–204, 206–208
 Simplify, 13, 95, 126, 151, 154, 156, 158–159,
 161, 162, 177, 180–181, 187, 191, 202,
 207
 Sin, 12–13, (64, 15–67, 69, 71, (102, 76–102,
 112–113, 115, 117, 120–122, 125,
 134–137, 163, 173–174, 177, (179, (180,
 179–182, 184–186, 188
 Solve, 18–19, 45, 60, 80, 129, 135, 147–150,
 170–171, 176–177, 180, 183–185, 190,
 195, 200–201, 205
 Span, 23
 Sphere, 193, 207
 Sqrt, 11–13, 28, 64, 120, 128–129, 131, 135,
 141, 151, 154, 173, 174, 179–181, 185,
 188, 193, 201, 206–208
 StreamPlot, 119
 StringExpression, 103
 StringFreeQ, 104
 StringLength, 103
 StringMatchQ, 103–104
 StringReplace, 103
 StringSplit, 104
 Style, 173–174, 178, 202
 Sum, 14, 67, 70, 81, 83–89, 146–150, 157, 162,
 195
 Switch, 45–46
 Symmetric, 97–98
 System, 105, 107

T

Table, (50, 50–51, 53, 66, 74, 80, 82–89,
 91–92, 117, 125, 129–131, 135–137,
 139, 156, 159, 161–163, 165, 168, 175,
 179, 182, 184, 188–191, 194–196, 207
 Tan, 64–65, 113–114, 174, 184
 TensorContract, 97–98
 TensorDimensions, 98
 TensorProduct, 97–98
 TensorRank, 98
 TensorReduce, 97–98

TensorSymmetry, 98
TeXForm, 102
Text, 139, 202
Ticks, 155
Times, 25, 28–33, 40–41, 46, 48, 50, 64, 67
Together, 12, 14, 26, 80, 94, 95, 128, 134, 205
Tr, 94
Transpose, 94
TreeForm, (26, 23–26
TrigExpand, 13, 180, 185, 187, 191
TrigReduce, 13, 146–151
True, 24–25, 39, 44–45, 67, 69, 73, 82–86, 88,
93, 98, 104, 126, 161, 162, 205

V

VectorPlot, 119

VectorQ, 93
Vectors, 97
ViewPoint, 123, 184, 193, 205, 207, 208

W

Which, 44, 125, 205
While, 47
With, 50, 120, 139, 173–175, 179, 181–182,
184, 188, 202–205
WorkingPrecision, 77
Write, 101

Z

Zeta, 14, 77, 165, 170