

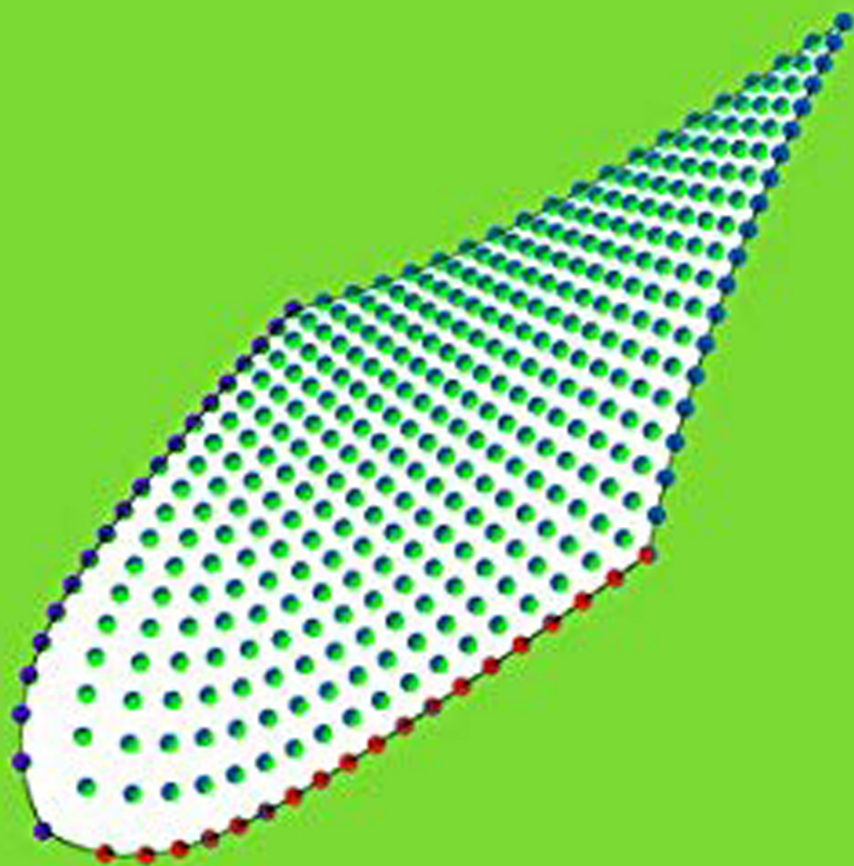
DE GRUYTER

GRADUATE

*Galina Filipuk, Andrzej Kozłowski*

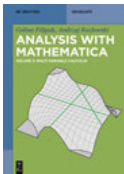
# ANALYSIS WITH MATHEMATICA

VOLUME 1: SINGLE VARIABLE CALCULUS



Galina Filipuk, Andrzej Kozłowski  
**Analysis with Mathematica®**

## Also of Interest



*Analysis with Mathematica®. Volume 2: Multi-variable Calculus*  
Galina Filipuk, Andrzej Kozłowski, 2020  
ISBN 978-3-11-066038-8, e-ISBN (PDF) 978-3-11-066039-5,  
e-ISBN (EPUB) 978-3-11-066041-8



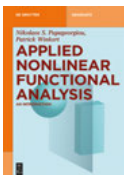
*Numerical Analysis. An Introduction*  
Timo Heister, Leo G. Rebholz, Fei Xue, 2019  
ISBN 978-3-11-057330-5, e-ISBN (PDF) 978-3-11-057332-9,  
e-ISBN (EPUB) 978-3-11-057333-6



*Web Applications with Javascript or Java. Volume 1: Constraint Validation, Enumerations, Special Datatypes*  
Gerd Wagner, Mircea Diaconescu, 2017  
ISBN 978-3-11-049993-3, e-ISBN (PDF) 978-3-11-049995-7,  
e-ISBN (EPUB) 978-3-11-049724-3



*Real Analysis. Measure and Integration*  
Marat V. Markin, 2019  
ISBN 978-3-11-060097-1, e-ISBN (PDF) 978-3-11-060099-5,  
e-ISBN (EPUB) 978-3-11-059882-7



*Applied Nonlinear Functional Analysis. An Introduction*  
Nikolaos S. Papageorgiou, Patrick Winkert, 2018  
ISBN 978-3-11-051622-7, e-ISBN (PDF) 978-3-11-053298-2,  
e-ISBN (EPUB) 978-3-11-053183-1



Galina Filipuk, Andrzej Kozłowski

# **Analysis with Mathematica<sup>®</sup>**

---

Volume 1: Single Variable Calculus

**DE GRUYTER**

**Mathematics Subject Classification 2010**

97I20, 97I30, 97I40, 97I50, 97N80

**Authors**

Dr. hab. Galina Filipuk  
University of Warsaw  
Faculty of Mathematics, Informatics and  
Mechanics  
Banacha 2  
02-097 Warsaw  
Poland  
filipuk@mimuw.edu.pl

Dr. Andrzej Kozłowski  
University of Warsaw  
Faculty of Mathematics, Informatics and  
Mechanics  
Banacha 2  
02-097 Warsaw  
Poland  
akoz@mimuw.edu.pl

The citation of registered names, trade names, trade marks, etc. in this work does not imply, even in the absence of a specific statement, that such names are exempt from laws and regulations protecting trade marks etc. and therefore free for general use. The various Wolfram trademarks and screenshots are used with the permission of Wolfram Research, Inc., and Wolfram Group, LLC.

ISBN 978-3-11-059013-5

e-ISBN (PDF) 978-3-11-059014-2

e-ISBN (EPUB) 978-3-11-059015-9

**Library of Congress Control Number: 2019947509**

**Bibliographic information published by the Deutsche Nationalbibliothek**

The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at <http://dnb.dnb.de>.

© 2019 Walter de Gruyter GmbH, Berlin/Boston

Cover image: Created by the authors with the help of Mathematica®

Typesetting: VTeX UAB, Lithuania

Printing and binding: CPI books GmbH, Leck

[www.degruyter.com](http://www.degruyter.com)

# Contents

## Preface — IX

### 1 Number systems — 1

- 1.1 Sets — 1
- 1.2 Domains — 2
- 1.3 Assumptions in Mathematica® — 5
- 1.4 Quantifiers — 7
- 1.5 Complex numbers — 10
- 1.6 Real numbers — 11
- 1.7 Infinities — 13
- 1.8 Integers and the Principle of Mathematical Induction — 15
  - 1.8.1 Example — 16
  - 1.8.2 Example — 16
- 1.9 Algebraic equations and algebraic numbers — 18
- 1.10 Non-algebraic equations — 21
- 1.11 Sequences of real numbers and their limits — 22
  - 1.11.1 Example — 25
  - 1.11.2 Example: the number  $e$  — 25
- 1.12 Supremum and infimum — 27
  - 1.12.1 Example — 29

### 2 Recursive sequences, discrete dynamical systems and their limits — 33

- 2.1 Example — 34
- 2.2 Example: the Fibonacci sequence — 41

### 3 Series — 47

- 3.1 Sequences and series — 47
- 3.2 The functions `Sum` and `NSum` — 49
- 3.3 Absolute convergence — 51
- 3.4 Convergence of series with terms of constant signs — 52
  - 3.4.1 Example — 53
- 3.5 Convergence of series with terms of non-constant signs — 55
  - 3.5.1 Grouping of terms — 55
  - 3.5.2 Example — 56
  - 3.5.3 Abel's summation formula — 57
  - 3.5.4 Dirichlet's and Abel's tests — 58
  - 3.5.5 Example — 60
- 3.6 The function `SumConvergence` — 62
  - 3.6.1 Example — 63

3.7	Riemann's theorem on conditionally convergent series —	65
3.8	The Cauchy product of series —	67
3.9	Divergent series —	68
3.10	Power series —	71
<b>4</b>	<b>Limits of functions and continuity —</b>	<b>75</b>
4.1	Limits of functions —	75
4.2	One-sided limits —	76
4.3	Continuous functions —	78
4.4	Discontinuous functions —	79
4.4.1	Example: the Dirichlet function —	82
4.5	The main theorems on continuous functions —	84
4.5.1	Example —	85
4.6	Inverse functions and their continuity —	86
4.7	Example: recursive sequences and continuity —	88
4.8	Uniform continuity and the Lipschitz property —	90
4.8.1	Example —	92
<b>5</b>	<b>Differentiation —</b>	<b>95</b>
5.1	Difference quotient and derivative of a function —	95
5.2	Differentiation in Mathematica <sup>®</sup> —	102
5.2.1	Differentiation of expressions using <code>D</code> —	102
5.2.2	Differentiation of functions using <code>Derivative</code> —	105
5.2.3	Algebraic rules of differentiation —	107
5.2.4	Example: user-defined derivative —	108
5.3	Main properties of differentiable functions —	109
5.3.1	Example: global and local extrema —	111
5.3.2	Example —	114
5.3.3	Example: the inverse function of the hyperbolic sine —	117
5.4	Convex functions —	119
5.4.1	Jensen's inequality —	123
<b>6</b>	<b>Sequences and series of functions —</b>	<b>125</b>
6.1	Power series continued —	125
6.1.1	Example —	126
6.1.2	Example —	127
6.2	Taylor polynomials and Taylor series —	129
6.2.1	Example —	135
6.2.2	Example —	136
6.2.3	Approximating functions by Taylor polynomials —	137
6.2.4	Example: rational approximation of $\sqrt{e}$ —	139



- 6.2.5 Example: illustration of approximation of functions with Taylor polynomials — **140**
- 6.3 Convergence of sequences and series of functions — **142**
- 6.3.1 Examples: pointwise, uniform and almost uniform convergence of function sequences — **145**
- 6.3.2 Continuity and differentiability of limits and sums — **149**
- 6.3.3 Examples: pointwise, uniform and almost uniform convergence of function series — **153**
- 6.3.4 Example — **156**
- 7 Integration — 159**
- 7.1 Indefinite integrals — **159**
- 7.2 The Risch algorithm — **163**
- 7.2.1 Differential algebras — **163**
- 7.2.2 Example 1: integration of rational functions — **165**
- 7.2.3 Example 2: the Risch algorithm for an exponential extension — **167**
- 7.2.4 Limitations of Mathematica<sup>®</sup>'s integration — **168**
- 7.3 The Riemann integral — **169**
- 7.3.1 Using Integrate and NIntegrate with definite integrals — **175**
- 7.3.2 Riemann sums — **176**
- 7.4 Improper integrals — **178**
- 7.4.1 Integrals over infinite intervals (improper integrals of the first type) — **178**
- 7.4.2 Improper integrals of the first type and infinite sums — **180**
- 7.4.3 Integrals of unbounded functions (improper integrals of the second type) — **184**

**Bibliography — 187**

**Index — 189**



# Preface

The main objective of this two-volume book is to convince the readers that *Mathematica*<sup>1</sup> can be used effectively as an aid in solving mathematical problems or at least to inspire the idea of a solution. Since we do not want to devote much space or time to repeating material covered in many other places, we will assume that the readers are already familiar with the basics of real analysis (e. g., as presented in the book [14] or in the more advanced book [12], which contains more than we need) and of *Mathematica*<sup>®</sup> (including some elements of the Wolfram Programming Language<sup>™</sup> roughly of the same scope as in Sections 1–6 of [9]).

Our approach is to describe some of the theory and to guide the readers through solutions to a number of classroom problems. Most of these problems come from the lecture notes [13] for the course of analysis for computer science students, given at the University of Warsaw during 2011–2018. The problems we chose were the ones in which *Mathematica*<sup>®</sup> could be genuinely useful in a way that did not involve simply applying some standard algorithm or plotting a graph, but which required the students to think and come up with some mathematical idea. Sometimes this idea is a guess inspired by a *Mathematica*<sup>®</sup> calculation or a visual representation of the problem. In some special cases *Mathematica*<sup>®</sup> can only verify this guess and sometimes it can prove its correctness in full generality. In all such cases we normally still want to obtain a rigorous mathematical proof. In some cases we will provide such proofs in detail, although sometimes we will only sketch them or limit ourselves to a special case, leaving the details and greater generality to the readers.

We will sometimes also use *Mathematica*<sup>®</sup>'s abilities to solve mathematical problems which are too difficult for us to solve by hand (sometimes because of their complexity and sometimes because the solution requires mathematical knowledge that lies beyond the scope of this book). We believe that such examples are useful and interesting because they show how the strengths of *Mathematica*<sup>®</sup> (or other computer programs) can complement human weaknesses and vice versa. When using a computer program as an aid in mathematics it is important to understand what such programs are well suited to and what they are not, and also what kind of problems (including incorrect answers) one can run into. To learn to judiciously use a computer is becoming a necessity in many areas of science and mathematics and we hope our book will prove useful in this respect.

We shall frequently state well-known mathematical theorems and often omit their proofs. Usually we will refer the readers to books included in the bibliography (except when the proof is simple enough for the readers to supply it themselves). In some cases

---

<sup>1</sup> Wolfram *Mathematica*<sup>®</sup> is a system for modern technical computing, see <http://www.wolfram.com/> for further information.

we will only give a “proof” based on Mathematica<sup>®</sup> or an illustration. There is a natural question that has been much discussed in recent years: when can results obtained by means of a computer software be regarded as mathematical proofs? The most popular view is that such results can be considered as proofs provided the source code of the mathematical software is open so that it can be examined to make sure that it is correct. In the case of commercial programs like Mathematica<sup>®</sup> the source code is not publicly available so one should consider results obtained by means of such programs as tentative and prove them by mathematical means or at least confirm their validity by means of other, independent programs. Since our purpose is to use Mathematica<sup>®</sup> as an educational tool, we shall not be concerned with the correctness of Mathematica<sup>®</sup>'s implementation of various algorithms and we will usually not try to describe them. However, the readers should be aware that the methods used by mathematical computer programs are usually very different from the ones used by humans. In addition, often the methods (algorithms) used by Mathematica<sup>®</sup> to obtain even quite elementary results depend on mathematical tools much more advanced than the scope of this book. Among examples of this kind are solving transcendental (non-polynomial) equations and inequalities, computing finite sums and sums of series, integration, etc.

We have been teaching analysis with Mathematica<sup>®</sup> at the University of Warsaw for computer science students for several years and we have found the above described approach quite effective. A typical mathematics course at the University of Warsaw consists of lectures at which the theory is taught and of accompanying problem classes. This book is primarily based on problem classes and therefore has a similar style. We only briefly sketch the necessary theoretical material and refer the readers for further details to standard textbooks, some of which can be found in the bibliography. Our book can also be seen as a supplement for any standard textbook on calculus or analysis and we shall cover standard first and second year calculus (analysis) topics, where we can benefit from using Mathematica<sup>®</sup> a lot. Volume 1 is devoted to single-variable calculus and volume 2 to the many-variable case. Our examples belong to pure mathematics and we do not consider any applications in physics and other sciences. Many textbooks exist which deal with such applications.

Although this book is based on a course of analysis for computer science students, it was actually a traditional mathematical course illustrated with examples from Mathematica<sup>®</sup> rather than a course of constructive or algorithmic mathematics (i. e., the kind of mathematics that Mathematica<sup>®</sup> itself makes use of). An excellent example of an introduction to analysis of the latter kind is [4], which we highly recommend. The difference between the constructive approach and the traditional approach is that the former avoids all indirect methods, such as “proofs by contradiction” which cannot be implemented on a computer. These are replaced, whenever possible, by “constructions”, which in principle could be turned into computer programs. Not everything in analysis can be done constructively (and thus not everything can be implemented on a computer) and the extent to which this is possible is an active area of research. Although in this book we do not adopt a constructive approach, we only try to attract the

attention of readers to such issues when they arise. We also do not consider the algorithms used by Mathematica<sup>®</sup> in detail (we recommend the already mentioned book [9] for some of such discussions, especially in connection with the Risch integration algorithm). We will make some comments on the algorithms from time to time in the hope that some readers will be inspired to pursue such topics further. Finally, there is another very important issue that we will only touch the surface of. It does not belong to traditional mathematics but is something of which one soon becomes aware of (often painfully) when one tries to apply Mathematica<sup>®</sup>'s symbolic capabilities to “real life” mathematical problems. This issue is computational complexity. In traditional mathematics a problem is often considered as solved as soon as a method (an algorithm) of solving it is found. These algorithms may, however, have such high complexity that in practice they allow only very simple cases to be solved in acceptable time. Some of the most impressive algorithms (e. g., quantifier elimination discussed in Chapter 1) have a very high complexity. This motivates the search for methods of solving problems quite different from the standard ones (an example of this is the standard algorithm for integrating rational functions, taught in all traditional Calculus courses, which is unusable in practice due to its need to factorize arbitrary polynomials). Again, in this book we only point out the existence of these important issues.

As mentioned earlier, we assume some familiarity with the programming language used by Mathematica<sup>®</sup> (Wolfram Language<sup>™</sup>), which means its basic syntax, the structure of Mathematica<sup>®</sup> expressions, different forms of expressions (InputForm, TraditionalForm, FullForm), patterns and pattern matching, etc. Recent versions of Mathematica<sup>®</sup> can use the so called “free-form input”, which allows the user to use “natural language” to tell Mathematica<sup>®</sup> to solve certain problems. Natural language has obvious advantages over formal languages. For example, Mathematica<sup>®</sup> has functions `Element` and `MemberQ` but only `MemberQ[{1, 2, 3}, 1]` and `Element[1, Integers]` will produce the answer `True`. If we use free-form input, the questions “is 1 an element of the set {1, 2, 3}?”, “is 1 a member of the set {1, 2, 3}?” and “is 1 a member of the rationals?” will return the same answer `True`. Although free-form input is improving, it is still quite limited in scope and we have decided not to use it. Using a formal language has the advantage of clarity, which is essential when we want to do something more complicated than a single operation like solving an equation or plotting a graph. Of course there is a price. Not only do we have to use the correct syntax (for example, in the Wolfram Language different kind of brackets `()`, `{}`, `[]` have different meanings and must not be mixed up) but to use Mathematica<sup>®</sup> effectively one needs to use its built-in functions. However, there are many thousands of them, each having a large number of options. In practice it means that one of the most important Mathematica<sup>®</sup> skills is efficient use of its extensive documentation (the Wolfram Language & System Documentation Center<sup>™</sup>). Whenever the readers come across an unfamiliar function or an unfamiliar option of a familiar function they should use “?FunctionName” to obtain information about it. In fact, it is not necessary to remember the whole function name as long as Mathematica<sup>®</sup>'s autocomplete feature is turned on (it is by default).

We expect the readers to look up the necessary information themselves but we will sometimes provide links to some selected help topics. We would also like to point out that this book is written in  $\text{\LaTeX}$  and it is not possible to make the input and output exactly the way they would look in Mathematica<sup>®</sup>. Most of our input cells will resemble `InputForm` with a mixture of `StandardForm`, while the output cells will mostly resemble `StandardForm`. Large input and output cells are sometimes shown as graphics. We shall use the large letter `I` in both input and output to represent the complex number  $i$ , where  $i^2 = -1$ , although in `StandardForm` Mathematica<sup>®</sup> actually uses a special symbol for it. The same is true for the letter `E` for `exp(1)`.

Although Mathematica<sup>®</sup> has become quite affordable recently, still not everybody has access to the latest version. All examples in this book should work in the same way with versions higher than 11.3 but this may not always be true for earlier ones. Some functions, such as `DiscreteLimit`, were introduced only in version 11.2 and will not work in earlier ones. Usually one can obtain the same results in earlier versions by means of a more complicated code. For example, `DiscreteLimit` can in many cases be replaced by `Limit` but not always (we give an example below). Sometimes (although comparatively rarely) the behavior of the built-in functions changes between different versions. For example, this happened to the function `Limit`: in Mathematica<sup>®</sup> 10 it always returns (by default) one-sided limits, in Mathematica<sup>®</sup> 11 it returns two-sided ones. Thus, in Mathematica<sup>®</sup> 10, `Limit[1/n, n -> 0]` returns  $\infty$ , while in Mathematica<sup>®</sup> 11 it returns `Indeterminate`. This is because the left sided and the right sided limits are different:  $-\infty$  and  $\infty$ .

Apart from such differences, any version of Mathematica<sup>®</sup> later than 6.0 should be suitable for most of the Mathematica<sup>®</sup> code in this book. Earlier versions are much less suitable because the graphics, both static and interactive, will not work. We strongly encourage the readers to consult the documentation for more detailed and up-to-date information and possible further issues.

We also recommend reading this book from the beginning since many pieces of advice are scattered throughout the text (since the material can be grouped differently and sometimes we digress to some interesting related topics).

Finally, we would like to say a few words about our use of the word “function” throughout the book. We assume that all readers are familiar with the ordinary use of the word “function” in mathematics. Informally it is a “black box” mechanism that, for a given element of a set, usually a number  $x$ , produces an element of another set, e. g.,  $x^2$ . More formally, a function or a mapping is a special case of a relation between two sets. The word function is also used in computer science (where it is a special case of what is called a “subroutine”). It is a program which, when supplied with certain input data, produces an output (the usual way to express this is by saying “it returns an output”). Applying a function to data (which must be given as function’s arguments) and computing the result is called “evaluation”. To be a function a subroutine should always return output. Subroutines that only change some data in computer’s memory

are not functions. Like mathematical functions, computer science functions have arguments and can be composed. The programming style based on writing programs as compositions of (preferably simple) functions is called “functional”. This is the style of programming most suited to mathematics and Mathematica<sup>®</sup> programs written in this style (which implies avoiding looping constructions such as `Do`, `For`, `While`, `which`, since they do not produce output, are not functions) tend to be more efficient than those written in the more common “procedural” style. Throughout the book we use the word “function” in both mathematical and computer science sense without trying to distinguish them. We hope that after reading this introduction the readers will not find it confusing.

Although there exist a number of books on calculus with Mathematica<sup>®</sup>, we hope that ours offers a different perspective and will inspire the reader to experiment with other topics in analysis, which we have had to leave out. This book is primarily intended for educational purposes, but we believe that some parts of it can be of interest to researchers as well.





# 1 Number systems

Usually, courses of analysis begin with number systems. Real analysis begins with the real numbers, which are either introduced axiomatically [12], [1] or constructed from something more “basic”, e. g., natural numbers or integers (which themselves are either introduced axiomatically [14] or defined in terms of sets [15]).

In this chapter we shall explain how Mathematica<sup>®</sup> deals with sets, numbers and sequences of numbers and consider many other related topics.

## 1.1 Sets

One of the first notions one has to learn to study modern mathematics is the notion of a set, since the natural language of modern mathematics is provided by set theory. Mathematica<sup>®</sup> uses the so called Wolfram Language, which does not have a built-in function corresponding to the mathematical notion of “set”. It is important to remark that the built-in function “Set” means something completely different. A finite ordered set exists in Mathematica<sup>®</sup> as List. One can also study the properties of ordinary finite sets by using the fact that Mathematica<sup>®</sup> has a natural way to sort (or order) any list of objects. This is done either by the function Sort or by the function Union:

```
In[.]:= Sort[{b, 7, 3, "dog", "cat", a}]  
Out[.]:= {3, 7, cat, dog, a, b}
```

In the example above the list contains objects of different kinds (symbols, numbers and strings of texts), but Mathematica<sup>®</sup> has a canonical way to arrange them. Because of this, any finite set of objects has a canonical representation as an ordered set so we can study sets by means of these canonical representations. If we want to check if two sets are equal, it is enough to check whether their canonical representations are equal. For example, the lists {1,2,3} and {2,1,3} are clearly not equal:

```
In[.]:= {1, 2, 3} == {2, 1, 3}  
Out[.]:= False
```

but they represent the same sets, therefore applying Sort or Union will return True:

```
In[.]:= Union[{1, 2, 3}] == Union[{2, 1, 3}]  
Out[.]:= True
```

Note that the function Join will not give the same result:

```
In[.]:= Join[{1, 2, 3}] == Join[{2, 1, 3}]  
Out[.]:= False
```

since it just concatenates lists without sorting them:

```
In[.]:= Join[{2, 1, 3}, {1}]  
Out[.]:= {2, 1, 3, 1}
```

Standard mathematical operations on finite sets (like union, intersection, complement) are implemented in Mathematica<sup>®</sup> as functions. For instance, `Union` applied to lists performs the usual sum operation on sets:

```
In[.]:= Union[{2, 3, 4}, {2, 3, 5}]
Out[.]:= {2, 3, 4, 5}
```

Intersection and Complement are other built in set-theoretic functions. They only work on lists.

Membership in a list is determined by the function `MemberQ` (which should not be confused with `Element`, see below):

```
In[.]:= MemberQ[{a, b, c, d, 1}, d]
Out[.]:= True
```

## 1.2 Domains

Certain sets of numbers are built in Mathematica<sup>®</sup> as “domains”: Integers, Rationals, Reals, Complexes, Algebraics and Primes. As their names suggest, the domain Reals is used for real numbers, Complexes is for complex numbers, Algebraics is for numbers that solve polynomial equations with rational coefficients and so on. As expected, the usual inclusions of sets are preserved (that is, the domain of Integers is a subset of the domain of Rationals). There is one more domain, Booleans, which consists of symbols True and False. Many functions will give the answer True or False, for instance comparing two numbers and determining which one is greater or less. You can type in either `1 < 2` or `Less[1, 2]` to get an output “True”.

Domains are generally used together with the function `Element`, which allows one to test for membership in these domains (for checking whether a given number is an element of a certain domain) and often with certain functions, e. g., `Simplify`, `FullSimplify`, `Reduce`, `Solve`, `Resolve`, `Minimize`, `Maximize`, `FindInstance`. We will consider them later on.

Let us have a look at the following examples:

```
In[.]:= Element[2, Reals]
Out[.]:= True

In[.]:= Element[Sqrt[2], Rationals]
Out[.]:= False
```

As can be seen from these examples, the function `Element` takes two arguments, a number and a domain, and it is a Boolean function, which means that we can think of it as answering the following question: is the number an element of the domain? If the answer is not immediately obvious, Mathematica<sup>®</sup> returns the input unevaluated:

```
In[.]:= Element[(Sqrt[3] - Sqrt[2])*(Sqrt[2] + Sqrt[3]),
  Integers]
Out[.]:=  $(-\sqrt{2} + \sqrt{3})(\sqrt{2} + \sqrt{3}) \in \mathbb{Z}$ 
```

This illustrates one of Mathematica<sup>®</sup>'s basic principles: when there is an input for which Mathematica<sup>®</sup> has no rules to apply, it returns the input. We can then apply to it additional transformations, not performed by default by the Mathematica<sup>®</sup> kernel. The simplest way is to use the function `Simplify` or its more powerful (but more time consuming) variant `FullSimplify`:

```
In[.]:= Simplify[Element[(Sqrt[3] - Sqrt[2])*(Sqrt[2] +
  Sqrt[3]), Integers]]
Out[.]:= True
```

Instead of using `Simplify` or `FullSimplify` one can apply one of many built-in function that Mathematica<sup>®</sup> has for transforming expressions. For example, in this case `Expand` will show us that the expression above is, in fact, 1, and then we can check whether it is an integer or not:

```
In[.]:= Expand[(Sqrt[3] - Sqrt[2])*(Sqrt[2] + Sqrt[3])]
Out[.]:= 1

In[.]:= Element[%, Integers]
Out[.]:= True
```

Some functions accept domains as one of their arguments. For example, to solve a given algebraic equation over the reals we can use

```
In[.]:= x /. Solve[x^3 == 1, x, Reals]
Out[.]:= {1}
```

In many cases if we do not specify the domain, Mathematica<sup>®</sup> assumes that it is working with complex numbers. For instance, just solving the cubic equation above without specifying the domain yields three roots:

```
In[.]:= x /. Solve[x^3 == 1, x]
Out[.]:= {1,  $(-1)^{1/3}$ ,  $(-1)^{2/3}$ }
```

We can also use a different method to obtain the real solutions: to find first the complex ones and then select only the real ones:

```
In[.]:= Select[x /. Solve[x^3 == 1, x], Element[#1, Reals] & ]
Out[.]:= {1}
```

Returning to the functions `Simplify` and `FullSimplify`, there are several useful options. We can add an option `TimeConstraint` to `FullSimplify` to control the time spent on evaluation. This forces Mathematica<sup>®</sup> to abort the evaluation once the time set in `TimeConstraint` is reached and to return the simplest form of the expression found so far.

What `Simplify` actually does is to try to simplify the given expression using built-in rules as well as possibly additional rules and information provided by the users. There is a default sense in which one expression is considered as simpler than another one.

The build-in notion of complexity of an expression (corresponding roughly to its `LeafCount`, which is based on the `FullForm` representation of expressions or the representation of expressions by means of the graph-theoretic notion of trees) will not always agree with what appears “simpler” to human eyes. For instance, `Mathematica`® by default considers the expression  $\sqrt{xy}$  as “simpler” than  $\sqrt{x}\sqrt{y}$ :

```
In[.]:= {LeafCount[Sqrt[x*y]], LeafCount[Sqrt[x]*Sqrt[y]]}
Out[.]:= {7, 11}
```

```
In[.]:= FullForm[Sqrt[x*y]]
Out[.]:= Power[Times[x,y], Rational[1,2]]
```

```
In[.]:= FullForm[Sqrt[x]*Sqrt[y]]
Out[.]:= Times[Power[x, Rational[1, 2]], Power[y,
Rational[1,2]]]
```

In fact, `Mathematica`®’s default notion of “simplicity” (or rather complexity) can be changed by the user by means of the options `ComplexityFunction` and `TransformationFunctions`. For example, `Mathematica`® cannot find any simpler form of the following expression by using the default `ComplexityFunction`:

```
In[.]:= FullSimplify[Sin[x]*Cos[x]]
Out[.]:= Cos[x] Sin[x]
```

However, suppose we want to find an expression without any cosines. For this purpose we need to define some function which will “penalize” the presence of cosines. For example, the complexity function below adds to the default `ComplexityFunction` (which is essentially `LeafCount`) three times the number of occurrences of cosines in the expression:

```
In[.]:= Simplify[Sin[x]*Cos[x], ComplexityFunction ->
(LeafCount[#1] + 3*Count[#1, _Cos, Infinity] & )]
Out[.]:=  $\frac{1}{2} \sin[2x]$ 
```

Of course, in this example instead of changing `ComplexityFunction` one can just use the build-in function `TrigReduce`:

```
In[.]:= TrigReduce[Sin[x]*Cos[x]]
Out[.]:=  $\frac{1}{2} \sin[2x]$ 
```

`Simplify` has another important option `TransformationFunctions`. By default, `Simplify` uses a number of transformations, which replace an expression by an equiv-

alent expression. However, some useful functions are not used, since they are generally time consuming. For example, the function `Factor`, which tries to factorize polynomials, has a very high time complexity, and therefore this function is not used by default. However, one can always add it to all functions used automatically as follows:

```
In[.]:= FullSimplify[(x^4 - 16)/(2 + x),
  TransformationFunctions -> {Automatic, Factor}]
```

```
Out[.]:= (-2 + x)(4 + x^2)
```

Compare this with `FullSimplify` with default options:

```
In[.]:= FullSimplify[(x^4 - 16)/(2 + x)]
```

```
Out[.]:=  $\frac{-16 + x^4}{2 + x}$ 
```

### 1.3 Assumptions in Mathematica®

It is possible to specify arbitrary symbolic assumptions about variables in the Wolfram Language. `Assumptions` is an option for functions like `Simplify`, `Refine` and `Integrate` and can take the form of equations, inequalities and domains. This option should not be confused with `$Assumptions`, which is a default setting for `Assumptions` and which can be modified temporarily by the function `Assuming`.

Sometimes it is necessary to specify the domain to which the values of the symbols belong. Let us compare the following outputs:

```
In[.]:= Simplify[Abs[x^2]]
```

```
Out[.]:= Abs[x]^2
```

```
In[.]:= Assuming[Element[x, Reals], Simplify[Abs[x^2]]]
```

```
Out[.]:= x^2
```

```
In[.]:= Simplify[Abs[Cos[x]] <= 1]
```

```
Out[.]:= Abs[Cos[x]] ≤ 1
```

```
In[.]:= Assuming[Element[x, Reals], Simplify[
  Abs[Cos[x]] <= 1]]
```

```
Out[.]:= True
```

Note that the function `Refine`, unlike `Simplify`, just applies assumptions but does not attempt to make any other simplifications, not related to the assumptions:

```
In[.]:= Refine[Sqrt[x^2], Assumptions -> {Element[x, Reals]}]
```

```
Out[.]:= Abs[x]
```

```
In[.]:= Assuming[x > 0, Refine[Sqrt[x^2] +
  Cos[x]^2 + Sin[x]^2]]
```

```
Out[.]:= x + Cos[x]^2 + Sin[x]^2
```

```
In[.]:= Simplify[Sqrt[x^2] + Cos[x]^2 + Sin[x]^2,
Assumptions -> {x > 0}]
```

```
Out[.]:= 1 + x
```

By default Mathematica<sup>®</sup> makes simplifications which are valid for complex numbers. If the user wants some simplifications which require additional assumptions which do not hold in general, one has to use the Assumptions mechanism. This is, for example, why Mathematica<sup>®</sup> does not give the answer True to the following identity:

```
In[.]:= Simplify[Sqrt[x*y] == Sqrt[x]*Sqrt[y]]
```

```
Out[.]:=  $\sqrt{xy} == \sqrt{x} \sqrt{y}$ 
```

However,

```
In[.]:= Assuming[x >= 0 && y >= 0, Simplify[
Sqrt[x*y] == Sqrt[x]*Sqrt[y]]]
```

```
Out[.]:= True
```

The alternative form, which is more useful, is

```
In[.]:= Simplify[Sqrt[x]*Sqrt[y], Assumptions ->
{x >= 0 && y >= 0}]
```

```
Out[.]:=  $\sqrt{xy}$ 
```

Note that the assumption  $x > 0$ , or any assumption that uses an inequality sign, automatically implies that  $x$  is real.

One can also use a global variable \$Assumptions to define certain assumptions that will hold throughout the session. For example,

```
In[.]:= $Assumptions = {Element[_ , Reals]};
```

```
In[.]:= $Assumptions
```

```
Out[.]:= {_ ∈ ℝ}
```

This is a quick way to make the assumption that everything (this is what the pattern `_` stands for) in any expression is real, for example,

```
In[.]:= Simplify[Re[x*y]]
```

```
Out[.]:= x y
```

(of course we could have used the more limited assumption that the symbols  $x$  and  $y$  represent real numbers).

To remove the global assumptions we should use

```
In[.]:= $Assumptions = True
```

and we can see that the following expression above will then not be simplified:

```
In[.]:= Simplify[Re[x*y]]
```

```
Out[.]:= Re[x y]
```

We can also locally change assumptions:

```
In[.]:= Block[{$Assumptions = a > 0}, Refine[Sqrt[a^2]]]
Out[.]:= a
```

The function `Assuming` is effectively equivalent to this block construction.

There exists one exception to the discussion above: the function `PowerExpand`. This function, without any assumptions, will perform a transformation of powers in the expression, which might not be true in general. For example,

```
In[.]:= PowerExpand[Sqrt[x*y]]
Out[.]:=  $\sqrt{x} \sqrt{y}$ 
```

As we know, this only holds if  $x$  and  $y$  are both positive. The reason for that behavior is simply practical convenience. If `PowerExpand` is used with `Assumptions`, then it behaves like any other `Mathematica`<sup>®</sup> function; in other words, it only performs expansions that are valid under the given assumptions. For example, the following expansion is valid only for  $x \geq 0$ :

```
In[.]:= PowerExpand[Sqrt[x^2]]
Out[.]:= x
```

Suppose we want to add the assumption  $x < 0$ . Then `PowerExpand` gives the correct answer:

```
In[.]:= PowerExpand[Sqrt[x^2], Assumptions -> x < 0]
Out[.]:= -x
```

Note that this is the only case in which `Assuming` and `Assumptions` are not equivalent. For example, in the following expression `PowerExpand` ignores conditions in the function `Assuming` and gives an incorrect answer:

```
In[.]:= Assuming[x < 0, PowerExpand[Sqrt[x^2]]]
Out[.]:= x
```

Therefore, one needs to be aware of the fact that assumptions in `PowerExpand` must be given by means of the `Assumptions` option and any other way (even the global setting with `$Assumptions`) will be ignored.

## 1.4 Quantifiers

Quantifiers play a big role in formulating axioms and theorems in pure mathematics. In algorithmic (computer) mathematics, they can also be used to solve non-trivial problems, by means of an algorithm called “quantifier elimination”.

In `Mathematica`<sup>®</sup> quantifiers are expressed by functions `ForAll` and `Exists`. They can be entered with the help of the Writing Assistant palette.

**In[.]**:= ForAll[x, Element[x, Reals], x > 0]

**Out[.]**:=  $\forall_{x,x \in \mathbb{R}} x > 0$

**In[.]**:=  $\forall_{x,x \in \mathbb{R}} x > 0$

**Out[.]**:=  $\forall_{x,x \in \mathbb{R}} x > 0$

**In[.]**:= Exists[x, Element[x, Reals], x > 0]

**Out[.]**:=  $\exists_{x,x \in \mathbb{R}} x > 0$

By default **Mathematica**<sup>®</sup> does not apply any rules to quantified expressions, and therefore we have the same expression as the input and the output.

Applying **Reduce** or **Resolve** (and also **FullSimplify**) to an expression with quantifiers will cause **Mathematica**<sup>®</sup> to use a famous algorithm called “quantifier elimination”, which (in certain situations) will return an equivalent expression without any quantifiers and without the associated bound variables (variables to which the quantifiers apply). The free variables (the ones that are not quantified) will remain. For instance, in the following example there is one bound variable  $x$  and a free variable  $a$ . After quantifier elimination an explicit condition on the parameter  $a$  is obtained:

**In[.]**:= **Resolve**[ForAll[x, x^2 + a > 0], Reals]

**Out[.]**:=  $a > 0$

If there are no free variables then the result should be either **True** or **False**.

The difference between **Resolve** and **Reduce** is that **Reduce** will often return a more “reduced” expression, but at the cost of longer computation. In simple cases both functions will return the same answer:

**In[.]**:= **Resolve**[ForAll[x, Element[x, Reals], x > 0]]

**Out[.]**:= **False**

**In[.]**:= **Reduce**[ForAll[x, Element[x, Reals], x > 0]]

**Out[.]**:= **False**

**In[.]**:= **Resolve**[Exists[x, Element[x, Reals], x^2 < 0]]

**Out[.]**:= **False**

**In[.]**:= **Reduce**[Exists[x, Element[x, Reals], x^2 < 0]]

**Out[.]**:= **False**

Let us try something more complicated. We ask: “what is the condition on  $b$  and  $c$  which will make the quadratic below positive for all values of  $x$ ?”

**In[.]**:= **Resolve**[ForAll[x, Element[x, Reals],  
x^2 + b\*x + c > 0]]

**Out[.]**:=  $b \in \mathbb{R} \ \&\& \ b^2 - 4c < 0 \ \&\& \ c > 0$

This is an example where we can see that **Reduce** gives a more “reduced” answer than **Resolve**:



```
In[.]:= Reduce[ForAll[x, Element[x, Reals],
  x^2 + b*x + c > 0]]
Out[.]:= c > 0 && -2*sqrt(c) < b < 2*sqrt(c)
```

Here is another interesting problem: we ask Mathematica<sup>®</sup> for the conditions when the quadratic  $ax^2 + bx + c$  has two equal roots:

```
In[.]:= Resolve[ForAll[{x, y}, a*x^2 + b*x + c == 0 &&
  a*y^2 + b*y + c == 0, x == y], {a, b, c}]
Out[.]:= (a = 0 && b != 0) || (a = 0 && c != 0) || (a != 0 && c = b^2/4a)
```

Unfortunately the quantifier elimination algorithm, although surprising and beautiful, has a very high complexity, which means that if we try to use it on expressions with a larger number of free parameters, it will not finish in a reasonable time even on a very fast computer.

It is important to know that Mathematica<sup>®</sup> assumes that the symbols are complex numbers (unless they appear in an inequality) if no information about the domain is given. This is demonstrated by the following example, where Mathematica<sup>®</sup> assumed that  $x$  is a complex number:

```
In[.]:= Reduce[Exists[x, x^2 == -1]]
Out[.]:= True
```

If we want  $x$  to be real, we have to explicitly specify this:

```
In[.]:= Reduce[Exists[x, Element[x, Reals], x^2 == -1]]
Out[.]:= False
```

There are certain situations where we may get a surprising answer, for example,

```
In[.]:= Reduce[Exists[x, Element[x, Complexes], x^2 < 0]]
Out[.]:= False
```

This looks like a mistake, since of course there are complex numbers whose square is negative. However (for reasons connected with efficiency) the function `Reduce` by default assumes that anything that appears in an inequality is real. So if we really want to work with complex numbers we must inform `Reduce` about that explicitly by specifying `Complexes` as the domain of `Reduce` (it is not enough just to use `Complexes` inside the quantifier):

```
In[.]:= Reduce[Exists[x, x^2 < 0], Complexes]
Out[.]:= True
```

We state this here only for completeness, because we will be generally working with real numbers and such issues will not arise.

## 1.5 Complex numbers

There is a subject called Complex Analysis, which deals with functions of complex numbers. In this book we shall be primarily concerned with real analysis (although we will sometimes digress into complex topics) but we will begin with complex numbers.

In Wolfram Language we can work with both explicit complex numbers and symbolic complex variables. There are many built-in functions to work with complex numbers (e. g., to evaluate the real and imaginary part, the absolute value and the argument, to compute the complex conjugate of a given complex number, to convert to different forms of a complex number and so on).

When dealing with complex numbers, one of the most useful functions is `ComplexExpand`. `ComplexExpand` can work with numerical complex expressions, which it arranges in the standard form for representing complex numbers (the real part + the imaginary part):

```
In[.]:= ComplexExpand[1/(1 + I)]
Out[.]:=  $\frac{1}{2} - \frac{I}{2}$ 
```

`ComplexExpand` can also work with symbolic complex variables. By default, without the second argument in `ComplexExpand`, all variables are assumed to be real, whereas if we specify that one or more variables are complex in the second argument, then the output is different:

```
In[.]:= ComplexExpand[a + I*a + I*b]
Out[.]:= a + I (a + b)
```

```
In[.]:= ComplexExpand[a + b, b]
Out[.]:= a + I Im[b] + Re[b]
```

```
In[.]:= ComplexExpand[a + b, {a, b}]
Out[.]:= I (Im[a] + Im[b]) + Re[a] + Re[b]
```

If an expression is real, then `ComplexExpand` acts just like `Expand`:

```
In[.]:= ComplexExpand[(a + b)^3]
Out[.]:=  $a^3 + 3 a^2 b + 3 a b^2 + b^3$ 
```

If the expression is complex, then it will be arranged in the standard form:

```
In[.]:= ComplexExpand[(a + I*b)^3]
Out[.]:=  $a^3 - 3 a b^2 + I(3 a^2 b - b^3)$ 
```

There is an option `TargetFunctions` of the function `ComplexExpand`. With this option `ComplexExpand` will try to give the result in terms of functions that are specified by the user from the list `Re`, `Im`, `Abs`, `Arg`, `Conjugate`, `Sign`. By default `Re` and `Im` are used. This is illustrated by the following examples, if the expression contains a complex symbol:

```

In[.]:= ComplexExpand[a + b*I, a]
Out[.]:= I (b + Im[a]) + Re[a]

In[.]:= ComplexExpand[a + b*I, a, TargetFunctions ->
  {Arg, Abs}]
Out[.]:= Abs[a] Cos[Arg[a]] + I (b + Abs[a] Sin[Arg[a]])

```

## 1.6 Real numbers

Rigorous courses of analysis (or any other branch of mathematics) begin with certain statements (called axioms) about undefined “primitive objects”, which are assumed to be true (because one always has to begin somewhere). Then everything else is proved by using these statements and basic rules of logic (which, we assume, everyone is familiar with). However, the choice of both the “primitive objects” and the “axioms” is not unique. This is true even if we want to make the number of axioms as small as possible or as “basic” level as possible.

It is commonly agreed that the most basic level is that of set theory, and one can indeed set up the foundations of mathematics (and hence also of analysis) beginning with axioms of set theory [15]. There are, in fact, several such “axiomatic set theories” (e. g., the Zermelo–Fraenkel set theory). Each of them allows one to construct the natural numbers beginning with sets. Once natural numbers have been constructed, Peano axioms, which will be discussed in Section 1.8, then become theorems, and one can then construct the rationals, and eventually the real numbers.

Another possible approach is to start with the natural numbers (satisfying the Peano axioms) and then construct the rationals, reals and complexes. To do this we still need to use the language of set theory, but we can treat it as part of “logic”.

Finally, we can take the quickest approach but also one that makes the most assumptions and introduces axioms that uniquely define the real numbers. The integers and the rationals are then defined as special kinds of real numbers.

Here we will take the last axiomatic approach to real numbers but we shall omit the axioms, assuming that the reader is already familiar with them (see, for example, [14]). In other words, we will assume that there exists a set  $\mathbb{R}$  with three binary operations  $+$ ,  $\cdot$  and  $>$  such that  $\mathbb{R}$  is a complete (commutative) ordered field. The meaning of “complete” will be explained later.

Mathematica<sup>®</sup> has a number of built-in “rules” which effectively implement these axioms for both complex and real numbers (recall that by default Mathematica<sup>®</sup> treats all variables as complex numbers).

Certain properties of complex numbers are carried out automatically on evaluation. For example, Mathematica<sup>®</sup> by default assumes that both addition and multiplication are commutative and associative so it immediately converts  $b+a$  to  $a+b$ , and  $b a$  to  $a b$  (i. e., variables are automatically arranged in the canonical lexicographic order).

Similarly, the associative laws for addition and multiplication are used automatically; for instance

**In[.]:** = (a + b) + c  
**Out[.]:** = a + b + c

However, the distributive law is not applied automatically:

**In[.]:** = a\*(b + c)  
**Out[.]:** = a (b + c)

If we want Mathematica<sup>®</sup> to apply it we need to use some other function, for example Expand or Distribute:

**In[.]:** = Expand[a\*(b + c)]  
**Out[.]:** = a b + a c

**In[.]:** = Distribute[a\*(b + c)]  
**Out[.]:** = a b + a c

The reason why Mathematica<sup>®</sup> applies certain basic operations automatically is due to the need for efficiency in computations. This need even forces Mathematica<sup>®</sup> to perform certain operations, like cancelations, without checking that no division by 0 is involved:

**In[.]:** = (x\*(x - y))/(x - y)  
**Out[.]:** = x

The question of mathematical correctness of such a cancelation is moot. One could argue that the very fact that we write  $1/x$  implies that  $x$  is non-zero. However, automatic canceling of common factors can (very rarely) have surprising consequences and even lead to wrong answers, so one should be aware of it. This problem is unavoidable since there is no algorithm that can decide in general if an expression is zero or not.

Mathematica<sup>®</sup> also has a built-in non-commutative multiplication, denoted by \*\* (though we shall not use it here):

**In[.]:** = Distribute[(a + b) \*\* (a + b)]  
**Out[.]:** = a \*\* a + a \*\* b + b \*\* a + b \*\* b

One can check in Mathematica<sup>®</sup> that the method of “quantifier elimination” will verify the axioms of an ordered field for real numbers. For example, we use Resolve (or Reduce) and quantifiers to check that the distributive rule holds:

**In[.]:** = Resolve[ForAll[{x, y, z}, Element[{x, y, z}, Reals],  
 $x*(y + z) == x*y + x*z$ ]]  
**Out[.]:** = True

Similarly, other axioms can be checked, for instance

```
In[.]:= Resolve[ForAll[{x, y, z}, Element[{x, y, z}, Reals]
  && x <= y, x + z <= y + z]]
```

```
Out[.]:= True
```

```
In[.]:= Resolve[ForAll[{x, y, z}, z >= 0 &&
  x <= y, x*z <= y*z]]
```

```
Out[.]:= True
```

The following expression contains one quantified (bound) variable  $x$  and two free variables  $y$  and  $z$ :

```
In[.]:= Exists[{x}, Element[{x, y, z}, Reals],
  x + y == 0 && x + z == 0]
```

The statement says that there exists a (complex) number  $x$  which has two additive inverses,  $y$  and  $z$ . Eliminating quantifiers will tell us that  $y$  and  $z$  must be equal. This can be viewed as a proof that every number has a unique additive inverse.

```
In[.]:= Reduce[Exists[{x}, Element[{x, y, z}, Reals],
  x + y == 0 && x + z == 0]]
```

```
Out[.]:= z ∈ ℝ && y == z
```

The axioms of ordered field are insufficient to determine the real numbers uniquely. In other words, there exist many different ordered fields, which are not “isomorphic” to  $\mathbb{R}$ . (Two sets with additional mathematical structure are said to be isomorphic if there exists a bijective mapping between them, such that both the mapping and the inverse preserve the structure.) In fact, the set  $\mathbb{Q}$  of rational numbers is also an ordered field, and the inclusion  $\mathbb{Q} \rightarrow \mathbb{R}$  preserves all the structure, but, of course, it is not a bijection (the real numbers are uncountable). In fact, there is just one additional axiom (of completeness) which is needed to determine the real numbers uniquely, but we shall leave it for later (see Section 1.12).

One consequence of the fact that only real numbers can be ordered is that any function that involves ordering (e. g., functions that maximize or minimize) can be applied only to subsets of real numbers or to real-valued functions.

## 1.7 Infinities

As usual in analysis we extend the real line by two symbols  $\infty$  and  $-\infty$ . In Mathematica® we can simply input them as `Infinity` ( $\infty$  in TraditionalForm) and `-Infinity` ( $-\infty$ ), but it is useful to note that the full forms of these expressions are `DirectedInfinity[1]` and `DirectedInfinity[-1]`:

```
In[.]:= FullForm[Infinity]
```

```
Out[.]:= DirectedInfinity[1]
```

In fact, Mathematica<sup>®</sup>'s function `DirectedInfinity` gives an “infinity” in the direction of each complex number  $z$  (for instance, `DirectedInfinity[I]` denotes infinity in the direction pointing up along the imaginary axis). This may be useful in certain problems of complex analysis, for instance

```
In[.]:= Limit[E^z, z -> DirectedInfinity[-1]]
Out[.]:= 0
```

```
In[.]:= Limit[E^z, z -> DirectedInfinity[1]]
Out[.]:= ∞
```

There is also a non-directed complex infinity, `ComplexInfinity`, whose `FullForm` is `DirectedInfinity[ ]`:

```
In[.]:= FullForm[ComplexInfinity]
Out[.]:= DirectedInfinity[ ]
```

`ComplexInfinity` will play no role in this book but it will sometimes occur in the output of some computations.

There is also another symbol, `Indeterminate`, which is used whenever a certain operation cannot be performed:

```
In[.]:= 0/0
...Power: Infinite expression 1/0 encountered.
...Infinity: Indeterminate expression 0 ComplexInfinity encountered.
Out[.]:= Indeterminate
```

Certain arithmetical operations are defined on infinities. Those operations which are not defined return `Indeterminate`. Here we give a couple of examples of the ones that are defined:

```
In[.]:= Infinity + Infinity
Out[.]:= ∞
```

```
In[.]:= Infinity*Infinity
Out[.]:= ∞
```

```
In[.]:= Infinity + 1
Out[.]:= ∞
```

```
In[.]:= 2*Infinity
Out[.]:= ∞
```

```
In[.]:= 1/Infinity
Out[.]:= 0
```

Examples of undefined operations are

```
In[.]:= Infinity - Infinity
...Infinity: Indeterminate expression -∞ + ∞ encountered.
Out[.]:= Indeterminate
```

**In[.]**:= Infinity/Infinity

...Infinity: Indeterminate expression  $0\infty$  encountered.

**Out[.]**:= Indeterminate

**In[.]**:=  $0\cdot\text{Infinity}$

...Infinity: Indeterminate expression  $0\infty$  encountered.

**Out[.]**:= Indeterminate

Note also that

**In[.]**:=  $1/0$

...Power: Infinite expression  $\frac{1}{0}$  encountered.

**Out[.]**:= ComplexInfinity

Whenever operations involving infinities are defined, they obey all the axioms of an ordered field:

**In[.]**:=  $2 < \text{Infinity}$

**Out[.]**:= True

**In[.]**:=  $-\text{Infinity} < -10$

**Out[.]**:= True

## 1.8 Integers and the Principle of Mathematical Induction

As we have already mentioned before, the most common axiomatic approach to analysis is based on constructing the real numbers after beginning with sets or with integers. For example, one can begin with the so called system of axioms of Peano ([14]).

*Peano's axioms.* There exists a set  $\mathbb{N}$  with an element  $1 \in \mathbb{N}$  and a function  $S : \mathbb{N} \rightarrow \mathbb{N}$  (called the successor function) which satisfies the following axioms:

1.  $\forall n, n \in \mathbb{N} \exists_{S(n)} S(n) \in \mathbb{N}$  (each  $n \in \mathbb{N}$  has a successor);
2.  $S(n) \neq 1$  (1 is not the successor of any  $n \in \mathbb{N}$ );
3.  $S(n) = S(m) \Leftrightarrow n = m$  (if  $m$  and  $n$  in  $\mathbb{N}$  have the same successor, then  $m = n$ , i. e., two distinct elements in  $\mathbb{N}$  cannot have the same successor);
4. if  $A \subset \mathbb{N}$  has the properties
  - $1 \in A$  and
  - $n \in A \Rightarrow S(n) \in A$ ,
 then  $A = \mathbb{N}$ .

The last axiom is called the *induction axiom*.

It turns out that by starting only with these axioms we can construct the set of real numbers satisfying all the axioms (including the completeness axiom below). Alternatively, if we start with the axioms of real numbers, we first define  $S : \mathbb{R}_+ \rightarrow \mathbb{R}_+$  (where  $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x > 0\}$ ) by  $S(x) = x + 1$ . A subset  $A$  of the real numbers possessing the property in the hypothesis of axiom 4 is called an inductive subset of  $\mathbb{R}$ . For example,

$\mathbb{R}_+$  is an inductive subset of  $\mathbb{R}$ . We now define  $\mathbb{N}$  as the intersection of all inductive subsets of  $\mathbb{R}_+$ . It is easy to prove that  $\mathbb{N}$  defined in this way satisfies all of Peano's axioms. In particular, axiom 4 now becomes a theorem (known as the "Principle of Mathematical Induction"), whose proof follows almost trivially from the definition. The result is extremely useful in proving theorems in which some statement is asserted to hold for all natural numbers (or possibly integers larger than some given integer).

The *Principle of Mathematical Induction* works as follows. Suppose we wish to prove that some sentence  $A_n$ , which depends on the integer  $n$ , holds for all  $n \geq 1$ . We consider the set  $A = \{k \in \mathbb{N} \mid A_k \text{ is true}\}$ . We first prove that  $1 \in A$ , i. e.,  $A_1$  is true. Finally we prove that  $n \in A \Rightarrow n + 1 \in A$ . Thus,  $A$  satisfies the hypothesis of axiom 4, hence it is equal to  $\mathbb{N}$ . Thus  $A_n$  is true for all  $n \in \mathbb{N}$ .

Let us consider two examples.

### 1.8.1 Example

Consider the finite sum

$$\sum_{k=1}^n k^3 = 1^3 + 2^3 + \dots + n^3.$$

Let us ask Mathematica<sup>®</sup> to find a closed form formula for this sum:

**In[.]**:= Sum[i^3, {i, 1, n}]

**Out[.]**:=  $\frac{1}{4}n^2(1+n)^2$

Now that we know the answer let us try to prove it by induction. Let  $A_1$  state that  $1^3 = 1^2 \cdot 2 / 4$ , which is true. Now suppose  $A_n$  is true for some  $n \in \mathbb{N}$ . Then

$$\begin{aligned} \sum_{k=1}^{n+1} k^3 &= \sum_{k=1}^n k^3 + (n+1)^3 = \frac{1}{4}n^2(n+1)^2 + (n+1)^3 \\ &= \frac{1}{4}(n+1)^2(n^2 + 4n + 4) = \frac{1}{4}(n+1)^2(n+2)^2. \end{aligned}$$

Hence,  $A_{n+1}$  is also true.

Next we give a different example, which illustrates how Mathematica<sup>®</sup> can be used to suggest a solution to a problem, which it alone cannot solve.

### 1.8.2 Example

Let  $p_n$  denote the  $n$ -th prime number. Show that  $\forall_{n, n \in \mathbb{N} \ \&\& \ n \geq 12} p_n \geq 3n$ . Before we come to the proof (where we will use induction), let us try to see how we could discover this statement with the help of Mathematica<sup>®</sup>. Mathematica<sup>®</sup> has a built-in function



$\text{Prime}[n]$ , which gives the  $n$ -th prime number. To check if the theorem is true for small integers  $n$  we can plot the functions  $\text{Prime}[n]$  and  $3n$  together:

```
In[ ]:= DiscretePlot[{Prime[n], 3*n}, {n, 1, 50}, PlotMarkers
-> Automatic, Filling -> None]
```

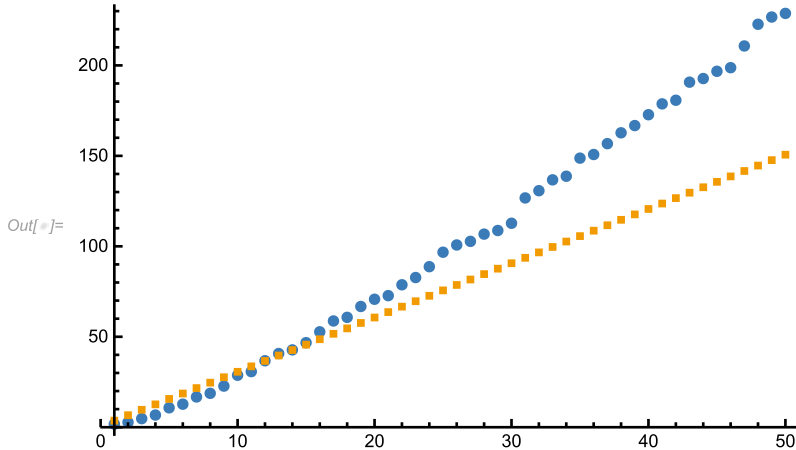


Figure 1.1

We see that once the graph of the function  $\text{Prime}[n]$  (discs) crosses that of  $3n$  (squares), it seems to stay above the latter, although the rate of growth of  $\text{Prime}[n]$  is not always greater than that of  $3n$  (which is three). To see how  $\text{Prime}[n]$  increases, we will look at the graph of the function that measures the differences between successive primes:

```
In[ ]:= DiscretePlot[Prime[i + 1] - Prime[i],
{i, 1, 50, 1}, Joined -> True]
```

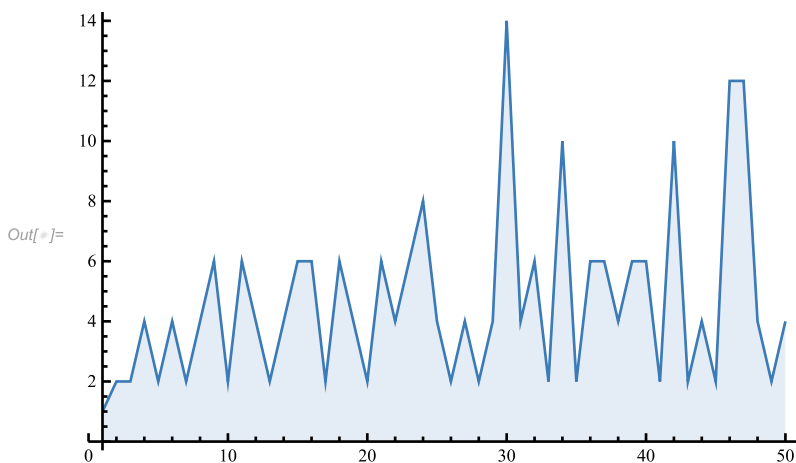


Figure 1.2

We see (what is actually quite obvious) that the differences must be at least 2. The question of whether there are infinitely many pairs of primes differing by 2 is a famous unsolved problem (recently there have been some remarkable results<sup>2</sup>). We can also see that two successive “jumps” of size 2 occur once (3, 5, 7) and it is easy to prove that this can never happen again. So now it should be easy to complete a proof: basically the function  $3n$  goes up in steps of 3, the function `Prime` can go up by 2 but in the next step it has to go up by at least 4. Once it is ahead by at least 2, it can never fall behind.

Let us now give a formal inductive proof. We first check that the statement is true for  $n = 12$ . Indeed,

**In[.]**:= Prime[12]  
**Out[.]**:= 37

**In[.]**:= 12\*3  
**Out[.]**:= 36

We now assume that the result holds for some  $n \geq 12$ , that is,  $p_n > 3n$ , and we will try to show that  $p_{n+1} > 3n + 3$ . There are two possibilities. Either  $p_{n+1} - p_n > 2$ , in which case  $p_{n+1} - p_n \geq 4$  (the difference must be even) and the inductive step follows, or  $p_{n+1} - p_n = 2$ . But then, by the inductive hypothesis  $p_{n+1} = p_n + 2 > 3n + 2$ , hence  $p_{n+1} \geq 3n + 3$ . But, of course, we cannot have  $p_{n+1} = 3n + 3$ , hence  $p_{n+1} > 3n + 3$  and the inductive step is complete.

## 1.9 Algebraic equations and algebraic numbers

Although real analysis can be studied without any mention of complex numbers, it is not a good approach when one is interested in the way computer programs do analysis. That is because many algorithms, even those that compute ultimately real quantities, make use of complex numbers. Probably the most profound example of this phenomenon is the Risch algorithm for computing indefinite integrals, which we will discuss later on in this book. The most famous classic example is solving polynomial equations (this example, of course, really belongs to algebra rather than analysis but we shall not engage in such pedantry here). `Mathematica`<sup>®</sup> by default assumes, unless specified otherwise, that one is working over the field of complex numbers when solving algebraic equations. As mentioned in Section 1.2, in order to obtain a real answer one needs to specify additionally the third argument, the domain `Reals`.

Instead of the function `Solve` we will often use the function `Reduce`, which can solve a variety of equations and inequalities but gives answers in a different form.

---

<sup>2</sup> <https://www.nature.com/news/first-proof-that-infinitely-many-prime-numbers-come-in-pairs-1.12989>

Both functions have their advantages, but to understand them better the reader should consult the documentation.

Now let us now try an equation with a parameter:

```
In[.]:= sols = Solve[x^3 - a == 0, x]
Out[.]:= {{x -> a1/3}, {x -> -(-1)1/3 a1/3}, {x -> (-1)2/3 a1/3}}
```

It is easy to prove that a cubic polynomial equation must possess at least one real root. Mathematica<sup>®</sup> can verify that they satisfy the original equation:

```
In[.]:= Simplify[x^3 - a /. sols]
Out[.]:= {0, 0, 0}
```

Now, suppose we only wanted to know the real root. We can ask Mathematica<sup>®</sup> to find it (using either Solve or Reduce):

```
In[.]:= Reduce[x^3 - a == 0, x, Reals]
Out[.]:= x == Root[-a + #1^3 & , 1]
```

This strange looking answer tells us that there is exactly one real root. To express it, Mathematica<sup>®</sup> does not use radicals but the “root object” Root. It is a symbolic expression beginning with the head Root, whose second argument is a number from 1 to the degree of the equation (in this case 1). You should think of these “root objects” as symbolic expressions like  $\sqrt{2}$  or  $\sqrt{a}$ . If we replace  $a$  by a number, Mathematica<sup>®</sup> can compute this expression with an arbitrary precision. For example

```
In[.]:= N[Root[-2 + #1^3 & , 1], 3]
Out[.]:= 1.26
```

This confirms that the root found by Mathematica<sup>®</sup> above is indeed real. We can also check that the other roots are not:

```
In[.]:= N[Reduce[x^3 - 2 == 0, x], 3]
Out[.]:= x == -0.630 - 1.091 I || x == 1.26 ||
x == -0.630 + 1.091 I
```

One more useful function, the function Roots (not Root), is used to obtain a disjunction (factorization) of equations which represent the roots of a polynomial equation:

```
In[.]:= Roots[(x - 1)*(x - 2)^2*(x - 3)^2 == 0, x]
Out[.]:= x == 3 || x == 3 || x == 2 || x == 2 || x == 1
```

Compare this with the output given by Reduce, where the multiple roots are not taken into account:

```
In[.]:= Reduce[(x - 1)*(x - 2)^2*(x - 3)^3 == 0, x]
Out[.]:= x == 1 || x == 2 || x == 3
```

Let us consider another example:

```
In[.]:= rts = x /. Solve[x^3 - 15*x + 2 == 0, x]
Out[.]:= { $\frac{5}{(-1 + 2 I \sqrt{31})^{1/3}} + (-1 + 2 I \sqrt{31})^{1/3}$ ,
 $-\frac{5(1 + I \sqrt{3})}{2(-1 + 2 I \sqrt{31})^{1/3}} - \frac{1}{2}(1 - I \sqrt{3})(-1 + 2 I \sqrt{31})^{1/3}$ ,
 $-\frac{5(1 - I \sqrt{3})}{2(-1 + 2 I \sqrt{31})^{1/3}} - \frac{1}{2}(1 + I \sqrt{3})(-1 + 2 I \sqrt{31})^{1/3}$ }
```

The reason why this may appear strange is that actually all the roots of this equation are real, as can be shown by asking Mathematica<sup>®</sup>:

```
In[.]:= FullSimplify[(Element[#1, Reals] & ) /@ rts]
Out[.]:= {True, True, True}
```

The problem here is that, although the roots are real, there is no way to express them in terms of radicals without using complex  $i$ . One can, of course, show that the roots are real by evaluating them numerically, or by using the function `ComplexExpand`:

```
In[.]:= Simplify[ComplexExpand[rts]]
Out[.]:= { $\sqrt{5} \left( \cos\left[\frac{1}{3}\text{ArcTan}[2\sqrt{31}]\right] + \sqrt{3} \sin\left[\frac{1}{3}\text{ArcTan}[2\sqrt{31}]\right] \right)$ ,
 $-2\sqrt{5} \cos\left[\frac{1}{3}\text{ArcTan}[2\sqrt{31}]\right]$ ,
 $\sqrt{5} \left( \cos\left[\frac{1}{3}\text{ArcTan}[2\sqrt{31}]\right] - \sqrt{3} \sin\left[\frac{1}{3}\text{ArcTan}[2\sqrt{31}]\right] \right)$ }
```

We have obtained expressions for all three roots that are clearly real but they are not expressed in terms of radicals. The famous theorems due to Abel and (more generally) to Galois state that the roots of arbitrary polynomial equations of degree greater or equal to five cannot be expressed in terms of radicals (like  $\sqrt[3]{2+i}$ ). Mathematica<sup>®</sup> can still solve such equations, but expresses the solutions using `Root`:

```
In[.]:= x /. Solve[x^5 + 2*x^3 + x - 2 == 0, x]
Out[.]:= {Root[-2 + #1 + 2 #1^3 + #1^5 &, 1],
Root[-2 + #1 + 2 #1^3 + #1^5 &, 2], Root[-2 + #1 + 2 #1^3 + #1^5 &, 3],
Root[-2 + #1 + 2 #1^3 + #1^5 &, 4], Root[-2 + #1 + 2 #1^3 + #1^5 &, 5]}
```

Both `Root` objects and radicals are examples of algebraic numbers.<sup>3</sup> Of course, some special higher degree equations can be solved in radicals:

<sup>3</sup> <http://reference.wolfram.com/language/ref/AlgebraicNumber.html>

**In[.]**:= sols = x /. Solve[x^5 + 2\*x^3 + x - 1 == 0, x]

**Out[.]**:=  $\{(-1)^{1/3}, (-1)^{2/3}, \frac{1}{3} \left(1 - 5 \left(\frac{2}{11 + 3\sqrt{69}}\right)^{1/3} + \left(\frac{1}{2}(11 + 3\sqrt{69})\right)^{1/3}\right),$   
 $\frac{1}{3} - \frac{1}{6}(1 + I\sqrt{3}) \left(\frac{1}{2}(11 + 3\sqrt{69})\right)^{1/3} + \frac{5(1 - I\sqrt{3})}{3 \times 2^{2/3}(11 + 3\sqrt{69})^{1/3}},$   
 $\frac{1}{3} - \frac{1}{6}(1 - I\sqrt{3}) \left(\frac{1}{2}(11 + 3\sqrt{69})\right)^{1/3} + \frac{5(1 + I\sqrt{3})}{3 \times 2^{2/3}(11 + 3\sqrt{69})^{1/3}}\}$

RootReduce will try to reduce the solutions to a simpler form or to convert the radicals to Root objects (except for the case of roots of degree 1 and 2):

**In[.]**:= RootReduce[sols]

**Out[.]**:=  $\{\frac{1}{2}(-1 - I\sqrt{3}), \frac{1}{2}(-1 + I\sqrt{3}), \text{Root}[-1 + 2\#1 - \#1^2 + \#1^3 \&, 1],$   
 $\text{Root}[-1 + 2\#1 - \#1^2 + \#1^3 \&, 2], \text{Root}[-1 + 2\#1 - \#1^2 + \#1^3 \&, 3]\}$

On the other hand, the inverse operation, ToRadicals, only works in special cases, e. g.,

**In[.]**:= ToRadicals[Root[5 - 4\*#1^3 + #1^6 &, 6]]

**Out[.]**:=  $(2 + I)^{1/3}$

## 1.10 Non-algebraic equations

In this section we will discuss one more feature of Mathematica<sup>®</sup>, which appeared only in version 9, the ability to solve exactly non-algebraic equations, that is, equations in which functions other than polynomials and rational functions (and also expressions that can be reduced to polynomials, e. g., by a substitution) appear.

Consider, for example, the equation

**In[.]**:= Exp[x] == Sin[x] + x

**Out[.]**:=  $E^x == x + \text{Sin}[x]$

This is, of course, a transcendental (non-algebraic) equation and most computer programs can only solve it approximately. If we try Solve or Reduce to solve this equation, without any additional information we will have

**In[.]**:= Solve[Exp[x] == Sin[x] + x, x]

... Solve: This system cannot be solved with the methods available to Solve.

**Out[.]**:= Solve[E<sup>x</sup> == x + Sin[x], x]

However, Mathematica<sup>®</sup> can prove that this equation has no real solutions:

**In[.]**:= Solve[Exp[x] == Sin[x] + x, x, Reals]

**Out[.]**:= {}

It can also find exact complex solutions in any bounded region in  $\mathbb{C}$ . For instance, we have two solutions that lie in the unit disk:

```
In[.] := sols = x /. Solve[Exp[x] == Sin[x] + x && Abs[x] <= 1,
  x]
```

```
Out[.] := {Root[
  {-E#1 + Sin[#1] + #1 &, 0.69512021780453778395228754266096116534888601484275672699242 -
  0.71872357647417749581491533428206539507394356773736262990844 I}], Root[
  {-E#1 + Sin[#1] + #1 &, 0.69512021780453778395228754266096116534888601484275672699242 +
  0.71872357647417749581491533428206539507394356773736262990844 I}]}
```

Mathematica<sup>®</sup> uses its own system for naming and distinguishing these solutions. These solutions are represented by Root objects. In the case of algebraic equations Root objects are ordered and numbered. In the case of non-algebraic equations, instead of ordering and numbering them, Mathematica<sup>®</sup> gives an approximation which distinguishes different roots. These roots can of course be computed to an arbitrary precision:

```
In[.] := N[First[sols], 3]
```

```
Out[.] := 0.695 - 0.719 I
```

Traditional methods (e.g., Newton's method) can only solve such equations approximately and require starting values. The function FindRoot uses Newton's method (or a variant of the secant method) to compute numerical solution to the equation with a given initial condition (which can be complex):

```
In[.] := FindRoot[Exp[x] == Sin[x] + x, {x, 1 + I}]
```

```
Out[.] := {x -> 0.69512 + 0.718724 I}
```

Unlike Solve and Reduce, FindRoot can return a "false" root:

```
In[.] := FindRoot[Exp[x] == Sin[x] + x, {x, 1}]
```

```
...FindRoot: The line search decreased the step size to within
tolerance specified by AccuracyGoal and PrecisionGoal but was
unable to find a sufficient decrease in the merit function.
You may need more than MachinePrecision digits of working
precision to meet these tolerances.
```

```
Out[.] := {x -> 0.601346}
```

Note that FindRoot can also work with systems of equations.

In general, Mathematica<sup>®</sup> can find the roots of any equation in one variable given by an analytic function (i. e., one that has a convergent Taylor expansion at every point of its domain; see Chapter 6). The methods that it uses are too advanced to be described in this book, but we shall make use of this ability later on.

## 1.11 Sequences of real numbers and their limits

A *sequence* of real numbers  $a_1, a_2, \dots, a_n, \dots$  is a function  $a : \mathbb{N} \rightarrow \mathbb{R}$ , where  $a_n = a(n)$ . We shall usually denote such a sequence by  $\{a_n\}$ . Mathematica<sup>®</sup> has no special

objects corresponding to the notion of a sequence (the function `Sequence` is a programming rather than a mathematical concept) but sometimes it is useful to simulate the behavior of sequences by using lists. For example

```
In[ ]:= Table[(-1)^n*(1/n), {n, 1, 8}]
Out[ ]:= {-1, 1/2, -1/3, 1/4, -1/5, 1/6, -1/7, 1/8}
```

can be viewed as the beginning (the first eight terms) of the sequence  $\{(-1)^n/n\}$ .

A sequence  $\{a_n\}$  is said to be *monotone* if it is increasing, i. e.,  $a_{n+1} \geq a_n$  for all  $n$ , or if it is decreasing, i. e.,  $a_{n+1} \leq a_n$  for all  $n$ . If a strict inequality holds, then we say that the sequence is *strictly monotone*.

We say that a sequence  $\{a_n\}$  tends to a limit  $a \in \mathbb{R}$ , as  $n$  tends to infinity, and write  $\lim_{n \rightarrow \infty} a_n = a$  (or  $a_n \rightarrow a$ ), if given any  $\varepsilon > 0$  we can find an integer  $n_0(\varepsilon)$  (the notation is meant to remind us that this integer depends on  $\varepsilon$ ) such that  $|a_n - a| < \varepsilon$  for all  $n \geq n_0(\varepsilon)$ . We say that a sequence  $\{a_n\}$  tends to  $\infty$  ( $-\infty$ ) if given any  $M \in \mathbb{R}$  there exists  $n_0(M)$  such that  $a_n > M$  ( $a_n < M$ ) for all  $n \geq n_0(M)$ . A sequence  $\{a_n\}$  is called *convergent* if it has a limit in  $\mathbb{R}$ . When  $\lim_{n \rightarrow \infty} a_n = \infty$  (or  $-\infty$ ) we say that  $\{a_n\}$  is divergent to  $\infty$  or  $-\infty$ .

One can easily prove the following well-known properties of limits (see, for instance [14, Section 2], [13]).

*Properties of limits:*

- (i) the limit of a convergent sequence is unique, i. e., if  $a_n \rightarrow a$  and  $a_n \rightarrow b$ , then  $a = b$ ;
- (ii) if  $a_n \rightarrow a$  and  $f : \mathbb{N} \rightarrow \mathbb{N}$  is such that  $\lim_{n \rightarrow \infty} f(n) = \infty$ , then  $a_{f(n)} \rightarrow a$ ;
- (iii) if  $a_n \rightarrow a$  and  $b_n \rightarrow b$ , then  $a_n + b_n \rightarrow a + b$ ;
- (iv) if  $a_n \rightarrow a$  and  $b_n \rightarrow b$ , then  $a_n b_n \rightarrow ab$ ;
- (v) if  $a_n \rightarrow a$  and  $a_n \neq 0$  for each  $n$  and  $a \neq 0$ , then  $a_n^{-1} \rightarrow a^{-1}$ ;
- (vi) if  $a_n \leq A$  for each  $n$  and  $a_n \rightarrow a$ , then  $a \leq A$ ; if  $a_n \geq B$  for each  $n$  and  $a_n \rightarrow a$ , then  $a \geq B$ ;
- (vii) if  $a_n = c$  for all  $n$ , then  $a_n \rightarrow c$ .

Statements (i) to (vi) are also valid when  $a$  and  $b$  are either  $\infty$  or  $-\infty$ , provided that the operations on the right hand side (involving  $a$  and  $b$ ) are well-defined (i. e., `Mathematica`<sup>®</sup> does not return `Indeterminate`).

Note also the following important fact: a sequence  $\{a_n\}_{n=1}^{\infty}$  has a limit  $a$  if and only if the sequence  $\{a_n\}_{n=k}^{\infty}$  has  $a$  as its limit for some  $k \geq 1$ . In other words, when considering limits of sequences one can always ignore any finite number of initial terms of the sequence.

The following theorems are also very important (see, for instance, [14, Section 2] for their proofs).

**Theorem 1** (The Monotone Convergence Theorem). *Every bounded monotonic sequence  $\{a_n\}$  of real numbers converges. Equivalently, a monotonic sequence converges if and only if it is bounded.*

Moreover, if the sequence is increasing, then  $\lim_{n \rightarrow \infty} a_n = \sup(\{a_n\})$ . If it is decreasing, then  $\lim_{n \rightarrow \infty} a_n = \inf(\{a_n\})$ .

**Theorem 2** (Bolzano–Weierstrass Theorem). *Every bounded sequence has a convergent subsequence.*

In versions of **Mathematica**<sup>®</sup> older than version 11.2, limits of sequences are computed using the function `Limit`, which is actually intended for computing limits of sequences that are obtained by restricting a continuous function  $a : \mathbb{R} \rightarrow \mathbb{R}$  to  $\mathbb{N}$ . In version 11.2 of **Mathematica**<sup>®</sup> a new function `DiscreteLimit` was introduced, which can compute the limits of many sequences which do not arise in this way. For example

```
In[.]:= DiscreteLimit[1/Prime[n], n -> Infinity]
```

```
Out[.]:= 0
```

while

```
In[.]:= Limit[1/Prime[n], n -> Infinity]
```

```
Out[.]:=  $\lim_{n \rightarrow \infty} \frac{1}{\text{Prime}[n]}$ 
```

Note that when an expression is formatted in `TraditionalForm` the functions `DiscreteLimit` and `Limit` will look exactly the same:

```
In[.]:= TraditionalForm[DiscreteLimit[f[n], n -> Infinity]]
```

```
Out[.]//TraditionalForm=  $\lim_{n \rightarrow \infty} f(n)$ 
```

```
In[.]:= TraditionalForm[Limit[f[n], n -> Infinity]]
```

```
Out[.]//TraditionalForm=  $\lim_{n \rightarrow \infty} f(n)$ 
```

There also exist `DiscreteMaxLimit` (`MaxLimit`) and `DiscreteMinLimit` (`MinLimit`) for computing limit superior and limit inferior.

We will use `DiscreteLimit` although almost all examples that we will consider can equally well be solved with `Limit`. However, here is an example<sup>4</sup> where the results after using `DiscreteLimit` and `Limit` are different:

```
In[.]:= DiscreteLimit[n*Sin[2*Pi*E*n!], n -> Infinity]
```

```
Out[.]:=  $2\pi$ 
```

```
In[.]:= Limit[n*Sin[2*Pi*E*n!], n -> Infinity]
```

```
Out[.]:= Indeterminate
```

Here we assume that the reader is familiar with the number  $\pi$  and trigonometric functions.

<sup>4</sup> <https://math.stackexchange.com/questions/76097/what-is-the-limit-of-n-sin-2-pi-cdot-e-cdot-n-as-n-goes-to-infinity>



Now let us try some other limits.

**In[ ]:=** Discretelimit[(1 - 1/n)^n, n -> Infinity]

**Out[ ]:=**  $\frac{1}{e}$

**In[ ]:=** Discretelimit[(1.0001 - 1/n)^n, n -> Infinity]

**Out[ ]:=**  $\infty$

**In[ ]:=** Discretelimit[(0.99999 + 1/n)^n, n -> Infinity]

**Out[ ]:=** 0.

Note that Mathematica<sup>®</sup> returned an approximate zero. This is a general principle: if a formula contains any *inexact number* (all decimal fractions represent inexact numbers in Mathematica<sup>®</sup>) the answer will always be inexact. Generally it is better to avoid using inexact numbers in symbolic computations. The following expression gives the exact answer to the example above:

**In[ ]:=** Discretelimit[(99999/100000 + 1/n)^n, n -> Infinity]

**Out[ ]:=** 0

### 1.11.1 Example

Often a result is much easier to prove if we somehow discover the answer (by guessing, using a graphic program or Mathematica<sup>®</sup>). Let us compute  $\lfloor nx \rfloor / n$ , where  $x$  is a real number.

**In[ ]:=** Discretelimit[Floor[n\*x]/n, n -> Infinity]

**Out[ ]:=**  $x$

In the case of limits, it is often the case that to prove that  $a_n \rightarrow x$  it is easier to prove the equivalent result  $|a_n - x| \rightarrow 0$ . In this case we have

$$0 \leq \left| x - \frac{\lfloor nx \rfloor}{n} \right| = \frac{|nx - \lfloor nx \rfloor|}{n} \leq \frac{1}{n}$$

since  $|nx - \lfloor nx \rfloor| \leq 1$ .

### 1.11.2 Example: the number $e$

Let us start with a well-known limit

**In[ ]:=** Discretelimit[(1 + 1/n)^n, n -> Infinity]

**Out[ ]:=**  $e$

This is usually taken as the definition of the number  $e$ . However, one needs to prove that the limit exists. The argument is based on applying the Monotone Convergence Theorem. To prove that the sequence  $(1 + 1/n)^n$  is convergent it is enough to show that it is bounded and increasing. Let us first verify this with Mathematica<sup>®</sup>:

**In[.]**:= Reduce[(1 + 1/n)^n < 3 && n >= 1, Integers]

**Out[.]**:= n ∈ ℤ && n ≥ 1

This means that Mathematica<sup>®</sup> reduced the given condition to “n is an integer greater than or equal to 1”. Now let us verify that the sequence is increasing. This will take longer:

**In[.]**:= Reduce[(1 + 1/n)^n < (1 + 1/(n + 1))^(n + 1) &&  
n >= 1, Integers]

**Out[.]**:= n ∈ ℤ && n ≥ 1

As we have explained in the preface we will (usually) not try to describe the algorithms that Mathematica<sup>®</sup> uses to obtain results such as these. This is a fascinating subject but it is not within the scope of this book.

Let us now consider the mathematical way of proving that the sequence is bounded, i. e.,  $(1 + 1/n)^n < 3$ . For  $n = 1$  the inequality is obvious. The proof for  $n ≥ 2$  is very simple and uses only the *Binomial Theorem*. This theorem is known to Mathematica<sup>®</sup> in the form

**In[.]**:= Sum[Binomial[n, i]\*a^i\*b^(n - i), {i, 0, n}]

**Out[.]**:= (a + b)<sup>n</sup>

where the binomial coefficient  $\binom{n}{k}$  is given by

**In[.]**:= FunctionExpand[Binomial[n, k], Assumptions ->  
{Element[{n, k}, Integers], n >= k >= 0}]

**Out[.]**:=  $\frac{\text{Gamma}[1 + n]}{\text{Gamma}[1 + k] \text{Gamma}[1 - k + n]}$

Mathematica<sup>®</sup> expresses the answer in terms of the Gamma function, which is the so called “special function” defined for all complex (and in particular real) numbers which has the property

**In[.]**:= FullSimplify[Gamma[n + 1], Assumptions ->  
Element[n, Integers] && n >= 0]

**Out[.]**:= n!

Now we can easily see that for  $n ≥ 2$

$$\begin{aligned} \left(1 + \frac{1}{n}\right)^n &= 1 + 1 + \frac{1}{2!} \left(1 - \frac{1}{n}\right) + \frac{1}{3!} \left(1 - \frac{1}{n}\right) \left(1 - \frac{2}{n}\right) + \dots + \frac{1}{n^n} \\ &\leq 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \dots \leq 1 + 1 + \frac{1}{2} + \frac{1}{2^2} + \dots = 3. \end{aligned}$$

The same binomial formula shows at once that

$$\left(1 + \frac{1}{n}\right)^n \leq \left(1 + \frac{1}{n+1}\right)^{n+1}$$

(just expanding both sides and comparing the terms). This kind of method cannot be used by Mathematica<sup>®</sup>. Although it can expand expressions such as

```
In[.]:= Expand[(a + b)^5]
Out[.]:= a5 + 5a4b + 10a3b2 + 10a2b3 + 5ab4 + b5
```

it cannot do so with a general  $n$ :

```
In[.]:= Expand[(a + b)^n]
Out[.]:= (a + b)n
```

## 1.12 Supremum and infimum

To state the last of the axioms that characterize the real numbers we need the concepts of the *supremum* (*least upper bound*) and the *infimum* (*greatest lower bound*) of a subset of  $\mathbb{R}$ .

Let  $A$  be a non-empty subset of  $\mathbb{R}$ . A real number  $M$  such that for all  $a \in A$  we have  $a \leq M$  is called an *upper bound* for  $A$ . Similarly, if there is a number  $m$  such that  $a \geq m$  for all  $a \in A$ , we say that  $m$  is a *lower bound* of  $A$ . Of course a set may have no upper or lower bound (when it has one we say that it is bounded above or below) and if it has one, it has infinitely many bounds. If  $M$  is an upper bound of  $A$  and also for every upper bound  $N$  we have  $N \geq M$ , then we say that  $M$  is the *supremum* (or *the least upper bound*) of  $A$ . The infimum (greatest lower bound) is defined similarly. When the set  $A$  has only finitely many elements, it has both the maximum and the minimum element and these are precisely the supremum and the infimum of the set. Infinite subsets of  $\mathbb{R}$  do not, of course, necessarily have maximum or minimum elements (think of open intervals). However, we have the *Axiom of Completeness*: every non-empty subset  $A \subset \mathbb{R}$  which is bounded above has a supremum. By replacing  $A$  with  $-A = \{-a \mid a \in \mathbb{R}\}$  we see that the analogous statement holds also for sets bounded below, with supremum replaced by infimum. For sets that are not bounded above we consider the supremum to be equal to  $\infty$  and for sets not bounded below we consider the infimum to be equal to  $-\infty$ . The supremum of the empty set is  $-\infty$  and the infimum is  $\infty$ .

A reformulation of the definition of supremum (similarly for infimum) that is often useful in practical problems is: “a supremum of a set  $A$  is an upper bound  $M$  of  $A$ , such that  $M - \varepsilon$  is not an upper bound of  $A$  for every  $\varepsilon > 0$ ”.

Let us consider the set

$$A = \{x \in \mathbb{R} \mid x^2 < 2\}. \quad (1.1)$$

Actually, we can check with Mathematica<sup>®</sup> that there exists an element satisfying our definition of the least upper bound, that is, one that is larger than all reals whose squares are less than two, but such that anything smaller than it no longer has this property:

```
In[.]:= Reduce[Exists[M, ForAll[y, y^2 <= 2, y <= M && ForAll[
  e, e > 0, Exists[z, z^2 <= 2 && M - e <= z]]]]]
Out[.]:= True
```

A typical example of the use of the axiom of completeness is proving the existence of roots, for example the square root of 2. It was already known to Ancient Greeks that there is no rational number whose square is equal to 2 (the discovery of this fact is attributed to Pythagoras). Now let us sketch an argument [12] which uses the axiom of completeness to show that there exists a real number whose square is 2. We consider the set  $A$  defined by (1.1). This set is bounded above. For example, 3 is an upper bound, since for any integer larger than 3 its square will be larger than 2. Hence the set has a least upper bound  $a$ . Now we need to show that  $a^2 = 2$ , which is done by indirect argument [12, Prop. 4.2]. Of course this information is built into Mathematica®:

```
In[ ]:= Reduce[Exists[x, Element[x, Rationals], x^2 == 2]]
Out[ ]:= False
```

```
In[ ]:= Reduce[Exists[x, Element[x, Reals], x^2 == 2]]
Out[ ]:= True
```

Finding suprema and infima is one of the main tasks of analysis. A large part of this book will be concerned with this problem and we will see that only in certain special cases complete solutions are possible.

Mathematica® has many built-in functions that try to find suprema and infima automatically. The most important ones are `Maximize` and `Minimize`. As their names suggest, these functions try to find the maximum and minimum of a set (normally the set of values of some function).

Suppose, for example, we want to find the supremum and the infimum of the set of real numbers of the form  $x^3 - 3x^2 + 1$ , where  $x$  is a real number in the open interval  $(-1, 1)$ . We can do this with `Maximize` and `Minimize` as follows:

```
In[ ]:= Maximize[{x^3 - 3*x^2 + 1, -1 < x < 1}, x]
Out[ ]:= {1, {x -> 0}}
```

The set (function) has a maximum value 1 attained at the point where  $x = 0$ .

```
In[ ]:= Minimize[{x^3 - 3*x^2 + 1, -1 < x < 1}, x]
...Minimize: Warning: there is no minimum in the region in
which the objective function is defined and the constraints
are satisfied; returning a result on the boundary.
Out[ ]:= {-3, {x -> -1}}
```

This time Mathematica® produces a warning that the minimum is not attained at any point within the region but it returns a point on the boundary and a minimum is attained there. That, as we shall prove later, is exactly equivalent to the statement that the infimum is  $-3$ .

To verify the result, we can plot the graph of the function  $x \mapsto x^3 - 3x^2 + 1$  by using `Plot[x^3 - 3*x^2 + 1, {x, -1, 1}]`.

We can solve the same problem by using quantifiers. In the first case both the supremum and the infimum are actually attained, i. e., they are the maximum and minimum values. In the second case the infimum is not attained.

```
In[.]:= Reduce[Exists[x, -1 <= x <= 1, y == x^3 - 3*x^2 + 1],
  Reals]
```

```
Out[.]:= -3 ≤ y ≤ 1
```

```
In[.]:= Reduce[Exists[x, -1 < x < 1, y == x^3 - 3*x^2 + 1],
  Reals]
```

```
Out[.]:= -3 < y ≤ 1
```

Maximize and Minimize work with any number of variables. For example:

```
In[.]:= Maximize[{x + y, x^2 + y^2 == 1}, {x, y}]
```

```
Out[.]:= {√2, {x -> 1/√2, y -> 1/√2}}
```

Even if the function is real-valued but is expressed in terms of complex numbers (which quite often happens when using Mathematica<sup>®</sup>) Maximize will not work:

```
In[.]:= Maximize[{Abs[2*x + I*y], x^2 + y^2 == 1}, {x, y}]
...Maximize: The objective function Abs[2 x+I y] contains a
nonreal constant I.
```

```
Out[.]:= Maximize[{Abs[2 x + I y], x^2 + y^2 == 1}, {x, y}]
```

In such cases it is necessary to use the function ComplexExpand to explicitly express the function in terms of the real and imaginary parts of the complex variable:

```
In[.]:= ComplexExpand[Abs[2*x + I*y]]
```

```
Out[.]:= √(4x^2 + y^2)
```

```
In[.]:= Maximize[{ComplexExpand[Abs[2*x + I*y]],
  x^2 + y^2 == 1}, {x, y}]
```

```
Out[.]:= {2, {x -> -1, y -> 0}}
```

### 1.12.1 Example

Let us find the supremum and infimum of the following sets of real numbers:

$$A = \left\{ \frac{7n+9k}{9n+7k}, n > 0, k > 0 \right\}, \quad B = \left\{ \frac{7n+9k}{9n+7k}, n \in \mathbb{N}, k \in \mathbb{N} \right\}. \quad (1.2)$$

The ability to compute limits can be very useful in proving that an upper (lower) bound of a set is a supremum (infimum). This is due to the following trivial lemma.

**Lemma 3.** *Let  $A$  be a set and  $M$  its upper bound. Then  $M$  is the supremum of  $A$  if and only if there exists a sequence  $\{a_n\}$ ,  $a_n \in A$  such that  $\lim_{n \rightarrow \infty} a_n = M$ .*

Let us see how this is applied in practice. We approach the problem first by looking at it graphically. We will use the functions `Plot3D` and `DiscretePlot3D`. First, let us look at the graphs which correspond to both sets:

```
In[.]:= g1 = Plot3D[(7*n + 9*k)/(9*n + 7*k), {n, 1, 20},
  {k, 1, 20}, ColorFunction -> ({Opacity[0.5], Green}
  & ), Mesh -> False];
```

```
In[.]:= g2 = DiscretePlot3D[(7*n + 9*k)/(9*n + 7*k), {n, 1, 20},
  {k, 1, 20}, AxesLabel -> {"n", "k"}];
```

```
In[.]:= Show[g1, g2]
```

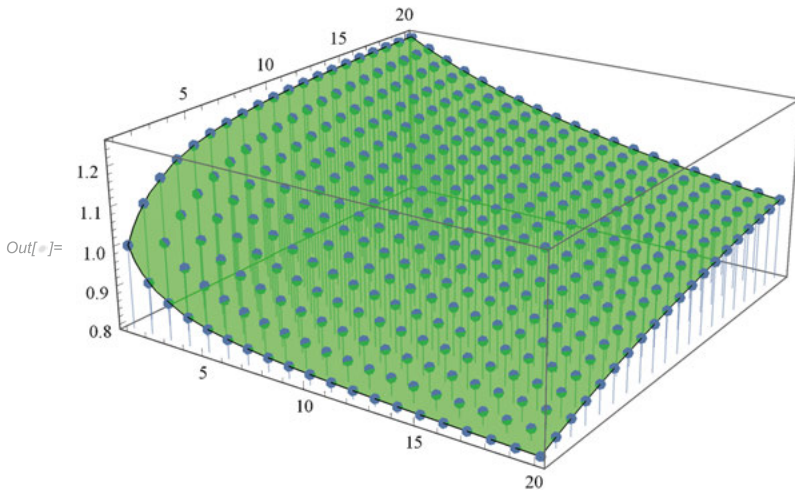


Figure 1.3

Clearly all the points belonging to the set  $B$  (represented by the discrete lattice of points) belong to  $A$  (represented by the surface). Looking at the graph and possibly expanding the range of  $n$  and  $k$  it seems that the surface and the lattice lie inside the box. The question is how to find the size of the enclosing box. In fact, for the surface this can be done using `Maximize` and `Minimize`:

```
In[.]:= Maximize[{(7*n + 9*k)/(9*n + 7*k), n > 0, k > 0},
  {n, k}]
```

```
Out[.]:= {9/7, {n -> 9/98, k -> 1/14}}
```

```
In[.]:= Minimize[{(7*n + 9*k)/(9*n + 7*k), n > 0, k > 0},
  {n, k}]
```

```
Out[.]:= {7/9, {n -> 1/18, k -> 7/162}}
```

Sometimes `Mathematica`<sup>®</sup> is not able to solve the problem over an unbounded domain of the function.

Clearly in our example  $9/7$  and  $7/9$  are the upper and lower bounds for the surface. Let us prove this by using Reduce:

```
In[.]:= Reduce[7/9 < (7*n + 9*k)/(9*n + 7*k) < 9/7 &&
n > 0 && k > 0, {n, k}]
```

```
Out[.]:= n > 0 && k > 0
```

This says that the inequality “reduces” to the conditions  $n > 0$  and  $k > 0$ , which mean that indeed these are an upper and lower bound (the reader can prove it easily by hand). So once we have found upper and lower bounds, we can try to prove that the first is the least one (i. e., supremum) and the second is the greatest one (i. e., infimum). So now, according to the lemma, we need to find sequences lying in  $A$  and  $B$  whose limits are the two bounds. In fact, our sequences will be in  $B$  (and hence also in  $A$ ) which will prove that the bounds are actually the supremum and infimum of both sets.

For the first sequence we fix  $k = 1$ , and we consider the sequence  $a_n = (7n + 9)/(9n + 7)$ . We have

```
In[.]:= Discretelimit[(7*n + 9)/(9*n + 7), n -> Infinity]
```

```
Out[.]:= 7/9
```

Similarly we fix  $n = 1$  and define  $b_k = (7 + 9k)/(9 + 7k)$ . We have

```
In[.]:= Discretelimit[(7 + 9*k)/(9 + 7*k), k -> Infinity]
```

```
Out[.]:= 9/7
```

We can illustrate this as follows.

```
In[.]:= g3 = Graphics3D[{Red, PointSize[0.025], Point[Table[
{n, 1, (7*n + 9)/(9*n + 7)}, {n, 1, 20}]]]}];
```

```
In[.]:= g4 = Graphics3D[{Purple, PointSize[0.025], Point[
Table[{1, k, (7 + 9*k)/(9 + 7*k)}, {k, 1, 20}]]]}];
```

```
In[.]:= Show[g1, g2, g3, g4]
```

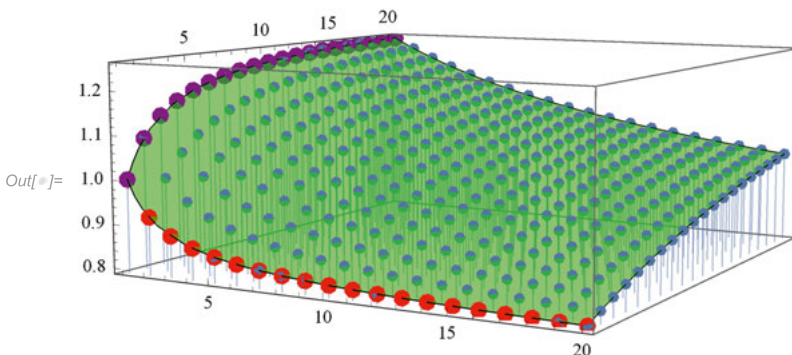


Figure 1.4

Of course there are many other sequences of points in  $B$  whose limits are  $9/7$  and  $7/9$ . Also, the supremum and infimum of the surface and a lattice of points need not to be the same, as in this example.

Note an important fact: although Mathematica® was able to solve the problem of finding the supremum and infimum of the set of points lying on the surface, it cannot do the same with the lattice (although it can solve the problem on any finite part of the lattice). Indeed, the expression

```
In[.]:= Maximize[{(7*n + 9*k)/(9*n + 7*k), Element[{n, k},
Integers] && n > 0 && k > 0}, {n, k}]
```

is returned not evaluated.

Although Maximize (and Minimize) do work over the integers in finite cases,

```
In[.]:= Maximize[{(9*k + 7*n)/(7*k + 9*n), Element[{n, k},
Integers] && 10 > n > 0 && 10 > k > 0}, {n, k}]
```

```
Out[.]:= { $\frac{97}{79}$ , {n -> 1, k -> 10}}
```

the problem of finding the least and greatest bounds of infinite discrete sets of points is actually harder. The function NMaximize, which uses numerical approximate methods, is more successful, but its answer is not guaranteed to be correct even approximately:

```
In[.]:= NMaximize[{(7*n + 9*k)/(9*n + 7*k), Element[{n, k},
Integers] && n > 0 && k > 0}, {n, k}]
```

```
...NMaximize: Failed to converge to the requested accuracy
or precision within 100 iterations.
```

```
Out[.]:= {1.2819, {n -> 1, k -> 170}}
```

Here only the first three digits are correct, as compared with

```
In[.]:= N[9/7]
```

```
Out[.]:= 1.28571
```



## 2 Recursive sequences, discrete dynamical systems and their limits

A *recursive sequence* is the sequence which is defined as follows: several initial values are given and the remaining ones are defined in terms of the previous ones. In this chapter we will show how to define recursive sequences by various methods, solve recurrence equations and also digress to other related topics (computing with inexact numbers with a given precision and so on).

Let us start with a simple case and define a recurrence

$$a_1 = 1, \quad a_n = a_{n-1} + 1.$$

Let us first look at some explicit terms of this sequence. There are a number of ways to do this. The fastest is the built-in function `RecurrenceTable`:

```
In[ ]:= RecurrenceTable[{a[n] == a[n - 1] + 1, a[1] == 1},  
a, {n, 1, 5}]
```

```
Out[ ]:= {1, 2, 3, 4, 5}
```

To get a better feeling for what is actually going on when `Mathematica`<sup>®</sup> builds this kind of lists, we will first do so using more basic programming constructs. Readers familiar with other programming languages might be tempted to use looping constructs. It is possible to do so with `Mathematica`<sup>®</sup> because `Mathematica`<sup>®</sup> has looping constructs such as `Do`, `For` and `While`, but it is generally better, both from the point of view of elegance and efficiency, to try to write programs in functional style and avoid them.

There are other ways to define recurrences. For instance a direct recursive definition:

```
In[ ]:= Clear[a]  
In[ ]:= a[1] := 1; a[n_] := a[n - 1] + 1  
In[ ]:= a[3]  
Out[ ]:= 3
```

`Mathematica`<sup>®</sup> remembers only the following information about `a`:

```
In[ ]:= ?a  
Out[ ]:= Global`a  
a[1] := 1  
a[n_] := a[n-1] + 1
```

and, therefore, computations might become slow.

We can modify the definition of  $a[n]$  above by using *dynamic programming* [16] such that in each evaluation the value of  $a[n]$  will be memorized and therefore the calculations become faster at the expense of using a little memory.

```

In[.]:= Clear[a]
In[.]:= a[1] := 1; a[n_] := a[n] = a[n - 1] + 1
In[.]:= a[3]
Out[.]:= 3

```

Now Mathematica<sup>®</sup> remembers the following data about  $a$ :

```

In[.]:= ?a
Out[.]:= Global`a
      a[1] := 1
      a[2] = 2
      a[2] = 3
      a[n_] := a[n] = a[n-1] + 1

```

Mathematica<sup>®</sup> has several functions designed to perform iterations. Here we will use `Nest` and `NestList`. The function `Nest` takes three arguments and works as follows:

```

In[.]:= Clear[f]
In[.]:= Nest[f, a, 5]
Out[.]:= f[f[f[f[f[a]]]]]

```

There is also a related function, `NestList`, which returns a list of all the values (beginning with the initial one) obtained during iteration:

```

In[.]:= NestList[f, a, 4]
Out[.]:= {a, f[a], f[f[a]], f[f[f[a]]], f[f[f[f[a]]]}

```

We will see later on how to apply these functions to recursive sequences.

Mathematica<sup>®</sup> has a function `RSolve` designed to solve recursive equations. Only certain types of equations can be solved as most are too difficult for `RSolve`:

```

In[.]:= RSolve[{a[n + 1] == a[n]*(a[n] + 1), a[1] == 1},
      a[n], n]
Out[.]:= RSolve[{a[n + 1] == a[n] (a[n] + 1), a[1] == 1},
      a[n], n]

```

Fortunately, usually one can find the limits of sequences defined by such equations (when they exist) without solving the equations.

In the next examples we will consider more complicated recurrences and methods of computing the terms of recursive sequences and finding their limits.

## 2.1 Example

Consider the sequence  $\{a_n\}$  given by

$$a_1 = 1, \quad a_{n+1} = \frac{1}{2} \left( a_n + \frac{2}{a_n} \right). \quad (2.3)$$

Using `RecurrenceTable` we can find the first few terms of the sequence:

```
In[.]:= RecurrenceTable[{a[n] == (1/2)*(a[n - 1] + 2/a[n - 1]),
  a[1] == 1}, a, {n, 1, 5}]
Out[.]:= {1,  $\frac{3}{2}$ ,  $\frac{17}{12}$ ,  $\frac{577}{408}$ ,  $\frac{665857}{470832}$ }
```

Let us compare how much time is required to compute a few terms of the sequence by using a simple recursive definition approach and by using the so called dynamic programming technique. We set the initial value

```
In[.]:= a[1] = 1
Out[.]:= 1
```

and then we define the following elements of the sequence by

```
In[.]:= a[n_] := (1/2)*(a[n - 1] + 2/a[n - 1])
```

To compute several elements of the sequence and to check the amount of time `Mathematica`<sup>®</sup> needs for this computation we use the function `Timing` (the function `Timing` evaluates the expression and returns a list of the time in seconds used and the result obtained):

```
In[.]:= Timing[Table[a[i], {i, 1, 20}]; ]
Out[.]:= {2.70313, Null}
```

In the expression above the semi-column actually suppresses the result of the evaluation (since it is very long) and therefore we see `Null` as the second element of the output. We see that it takes a lot of time to compute the elements of the sequence. Using dynamic programming we compute the same result faster:

```
In[.]:= Clear[a]
In[.]:= a[1] = 1; a[n_] := a[n] = (1/2)*(a[n - 1] + 2/a[n - 1])
In[.]:= Timing[Table[a[i], {i, 1, 20}]; ]
Out[.]:= {0.046875, Null}
```

The terms of our sequence (2.3) are obtained by iterating the function `Function[x, 1/2 (x+2/x)]` and evaluating it at 1. In an abbreviated (pure) form the function can be written as `1/2(# + 2/#)&`.

```
In[.]:= NestList[(1/2)*(#1 + 2/#1) & , 1, 5]
Out[.]:= {1,  $\frac{3}{2}$ ,  $\frac{17}{12}$ ,  $\frac{577}{408}$ ,  $\frac{665857}{470832}$ ,  $\frac{886731088897}{627013566048}$ }
```

Because we started with an exact initial value 1 and our function also contains only exact numbers, the whole computation is performed using exact arithmetic, which is very slow and not very informative. Let us replace the initial value 1 by an *inexact number* (1.). All computations will then be with inexact numbers. Recall that in `Mathematica`<sup>®</sup> decimals always represent inexact numbers, or more precisely so called `MachinePrecision` numbers.

```
In[.]:= NestList[(1/2)*(#1 + 2/#1) & , 1., 6]
Out[.]:= {1., 1.5, 1.41667, 1.41422, 1.41421, 1.41421, 1.41421}
```

It now looks like the iteration stabilizes after only a few steps. We can see better what is going on by expressing the output in `InputForm`, because otherwise `Mathematica`<sup>®</sup> (by default) shows only a few leading digits:

```
In[.]:= InputForm[NestList[(1/2)*(#1 + 2/#1) & , 1., 6]]
Out[.]//InputForm= {1., 1.5, 1.4166666666666665, 1.4142156862745097,
1.4142135623746899, 1.414213562373095,
1.414213562373095}
```

The sequence still stabilizes, but that is only because a point is reached where the difference between successive numbers is smaller than the precision with which the computations have been performed (the so called machine precision, which is in this case 15 digits).

In `Mathematica`<sup>®</sup> Precision of an approximate number is not exactly equal to the number of digits after the decimal point and in fact it is not even an integer but a real number which measures the relative error in the number. It is approximately equal to the number of significant digits.

When the differences between successive terms becomes sufficiently small `Mathematica`<sup>®</sup> no longer distinguishes between them and the sequence appears to reach a fixed point of iteration. We could reach this fixed point quicker if instead of `NestList` we use the function `FixedPoint` (`FixedPointList`) which performs iteration until a fixed point is reached. `FixedPoint` starts with an expression and then applies the function repeatedly until the result no longer changes:

```
In[.]:= InputForm[FixedPointList[(1/2)*(#1 + 2/#1) & , 1.]]
Out[.]//InputForm= {1., 1.5, 1.4166666666666665, 1.4142156862745097,
1.4142135623746899, 1.414213562373095,
1.414213562373095}
```

```
In[.]:= Length[%]
Out[.]:= 7
```

It took six iterations to reach the fixed point. Of course we have to be careful not to use `FixedPoint` with an exact initial value since the exact sequence has no fixed point! (We could, however, use the form `FixedPoint[a, f, n]` which will stop either after a fixed point is reached or after  $n$  iterations, whichever happens sooner.)

In addition to `MachinePrecision` arithmetic `Mathematica`<sup>®</sup> also has *arbitrary precision arithmetic*. A number  $a$  with precision  $p$  is most easily entered as `N[a, p]`. Here  $a$  should either be an exact number (which has precision infinity) or a number with precision higher than  $p$ . For example,

```
In[.]:= N[1, 5]
Out[.]:= 1.0000
```

```
In[.] := N[1/4, 20]
Out[.] := 0.25000000000000000000
```

```
In[.] := N[%, 5]
Out[.] := 0.25000
```

```
In[.] := Precision[%]
Out[.] := 5.
```

Note the difference between using `$MachinePrecision` and `MachinePrecision`:

```
In[.] := N[1, $MachinePrecision]
Out[.] := 1.000000000000000000
```

```
In[.] := Precision[%]
Out[.] := 15.9546
```

But

```
In[.] := N[1, MachinePrecision]
Out[.] := 1.
```

```
In[.] := Precision[%]
Out[.] := MachinePrecision
```

As we have already mentioned before, exact numbers have infinite precision:

```
In[.] := Precision[1]
Out[.] := ∞
```

We can compute our example with higher precision:

```
In[.] := FixedPointList[(1/2)*(#1 + 2/#1) & , N[1, 30]]
Out[.] := {1.00000000000000000000000000000000,
1.50000000000000000000000000000000000000,
1.4166666666666666666666666666666666667,
1.41421568627450980392156862745,
1.41421356237468991062629557889,
1.41421356237309504880168962350,
1.41421356237309504880168872421,
1.41421356237309504880168872421}
```

```
In[.] := Length[%]
Out[.] := 8
```

In fact, we need 100 digits of precision to make the sequence longer by just a few elements:

```
In[.] := Length[FixedPointList[(1/2)*(#1 + 2/#1) & , N[1, 100]]]
Out[.] := 10
```

Mathematica<sup>®</sup> can completely solve our recurrence:

```
In[.]:= sol[n_] = First[a[n] /. RSolve[{a[n] == (1/2)*
(a[n - 1] + 2/a[n - 1]), a[1] == 1}, a[n], n]]
...Solve: Inverse functions are being used by Solve, so some
solutions may not be found; use Reduce for complete
solution information.
```

```
Out[.]:=  $\sqrt{2} \operatorname{Coth} \left[ 2^{-1+n} \operatorname{ArcCoth} \left[ \frac{1}{\sqrt{2}} \right] \right]$ 
```

The function RSolve warns us that there may be also other solutions; we shall ignore this warning and compute the limit.

```
In[.]:= DiscreteLimit[sol[n], n -> Infinity]
Out[.]:=  $\sqrt{2}$ 
```

Without any initial conditions we can obtain the general solution

```
In[.]:= First[a[n] /. RSolve[{a[n] == (1/2)*
(a[n - 1] + 2/a[n - 1])}, a[n], n]]
Out[.]:=  $-I \sqrt{2} \operatorname{Cot}[2^n C[1]]$ 
```

Here C[1] is an arbitrary constant to be determined from initial conditions. However, this answer is not satisfactory to us, since we get the complex  $i$  in the answer and we do not clearly see the dependence on initial data. To make things clearer, we obtain the answer which explicitly depends on the initial conditions:

```
In[.]:= First[a[n] /. RSolve[{a[n] == (1/2)*
(a[n - 1] + 2/a[n - 1]), a[1] == x}, a[n], n]]
...Solve: Inverse functions are being used by Solve, so some
solutions may not be found; use Reduce for complete
solution information.
```

```
Out[.]:=  $\sqrt{2} \operatorname{Coth} \left[ 2^{-1+n} \operatorname{ArcCoth} \left[ \frac{x}{\sqrt{2}} \right] \right]$ 
```

and we can check the limits:

```
In[.]:= DiscreteLimit[%, n -> Infinity, Assumptions -> {x > 0}]
Out[.]:=  $\sqrt{2}$ 
```

```
In[.]:= DiscreteLimit[%, n -> Infinity, Assumptions -> {x < 0}]
Out[.]:=  $-\sqrt{2}$ 
```

Note that the last two evaluations take a lot of time.

Now we are going to find the limit of our sequence (2.3) without solving the recurrence equation. We will start by assuming that the sequence  $a_n$  has a limit. Under this assumption, we will find all the possible values that this limit can have. Finally we will prove this assumption (that the limit exists) and determine its actual values.

Consider the sequence given by the equation  $a_{n+1} = 1/2(a_n + 2/a_n)$  and assume it has a limit  $a$  (possibly  $\infty$  or  $-\infty$ ). We can suppose that  $a_n \neq 0$  for all  $n$ ; otherwise the

sequence could not be defined. We can consider the equation as an equality of two sequences  $\{a_{n+1}\}$  and  $\{1/2(a_n + 2/a_n)\}$ . The sequence  $\{a_{n+1}\}$  is a subsequence of  $\{a_n\}$  and therefore has the same limit  $a$ . Looking at the equation we immediately see that  $a \neq 0$ . Then the sequence  $\{1/2(a_n + 2/a_n)\}$  converges to  $1/2(a + 2/a)$ . Thus we obtain the equation

$$a = \frac{1}{2}\left(a + \frac{2}{a}\right).$$

Solving it we get:

**In[.]**:= a /. Solve[a == (1/2)\*(a + 2/a), a, Reals]

**Out[.]**:=  $\{-\sqrt{2}, \sqrt{2}\}$

There are also two other possibilities that we will still need to consider:  $\infty$  and  $-\infty$  (which also satisfy the equation).

We now turn to the next stage: proving the existence of a limit and finally determining its value. Now we will need the information (which we have not used so far) about the initial condition. We can prove the following statements. If  $a_1 > 0$ , then  $a_n \geq \sqrt{2}$  for  $n > 1$ , which follows from the well-known inequality between the geometric and arithmetic means of two numbers:  $1/2(a_n + 2/a_n) \geq \sqrt{2}$  and induction. If  $a_1 < 0$ , then  $a_n \leq -\sqrt{2}$  for  $n > 1$ . The sequence is decreasing exactly when

**In[.]**:= Reduce[(1/2)\*(x + 2/x) <= x, x]

**Out[.]**:=  $-\sqrt{2} \leq x < 0 \vee x \geq \sqrt{2}$

Since  $a_1 = 1$ , we have  $a_n \geq \sqrt{2}$  for  $n > 1$  and the next term of the sequence  $a_{n+1}$  will be smaller and it can never fall below  $\sqrt{2}$ . Thus the sequence is monotone decreasing and bounded below by  $\sqrt{2}$ , hence it has a limit greater than or equal to  $\sqrt{2}$ . But the only possibilities are  $\infty$  and  $\sqrt{2}$  and since the sequence is decreasing, the only possibility for the limit is  $\sqrt{2}$ . Arguing in the same way, we see that if the initial value  $a_1$  is negative, then the limit of the sequence is  $-\sqrt{2}$ .

It is natural to think of a sequence given by a recursive equation as of the motion of a particle, which at the starting time is at  $a_1$ , then moves to  $a_2$ , etc. The equation  $a_{n+1} = f(a_n)$  (in our case  $f(x) = 1/2(x + 2/x)$ ) can be thought of as an equation of motion of a particle, giving its position at time  $n + 1$  in terms of its position at time  $n$ . This is known as a *discrete dynamical system*. A *fixed point* of a function  $f : X \rightarrow X$ , where  $X \subset \mathbb{R}$ , is a point  $x \in X$  such that  $f(x) = x$ . In other words, if the particle is located at a fixed point it stops moving. As we will see later, when  $f$  is a continuous function, a limit of such a sequence will always have to be a fixed point of  $f$ . Of course a fixed point is always the limit of the constant sequence starting at that point.

Mathematica<sup>®</sup>'s ability to quickly create interactive visualizations using Manipulate (or Dynamic) makes it easy to study discrete dynamical systems even in cases when they are difficult to deal with mathematically. In the interactive illustration below we show the movement of a particle (red point) whose position is  $a_{n+1}$  after  $n$  iterations. Initially the slider for the number of iterations should be set to zero. The starting point can be chosen by moving the gray circle (locator in Mathematica<sup>®</sup>) to any point

in the interval  $[-4, 4]$  on the real line. By default the starting point is chosen as 2. The blue points correspond to the fixed points. Because it is difficult to manually move the initial point to the position of the fixed points, one can do it by clicking the plus sign in the upper right corner and choose one of the bookmarked positions. All motions of the particle are on the real line but for visual convenience we use 2-dimensional graphics. We also use the function `Quiet` to suppress any unwanted messages from `Mathematica`<sup>®</sup>. Once locator is used to choose the starting point, then by increasing the number of iterations we can watch the point move to a limit, which is one of the two fixed points:  $\sqrt{2}$  and  $-\sqrt{2}$ .

```
In[ ]:= Manipulate[Quiet[Module[{fix}, fix =
  FixedPoint[(1/2)*(#1 + 2/#1) &, First[p], m];
  Graphics[{PointSize[0.02], Point[p], Red,
  PointSize[0.02], Point[{fix, 0}], Blue, Point[
  {{Sqrt[2], 0}, {-Sqrt[2], 0}}]}, PlotRange ->
  {{-4, 4}, {-1, 1}}, Axes -> True]], {{p, {2, 0}},
  {-4, 0}, {4, 0}, Locator}, {{m, 0, "number of
  iterations"}, 0, 10, 1, Appearance -> "Labeled"},
  Bookmarks -> {"fixed point 1" :> (p = {Sqrt[2], 0}),
  "fixed point 2" :> (p = {-Sqrt[2], 0})}]
```

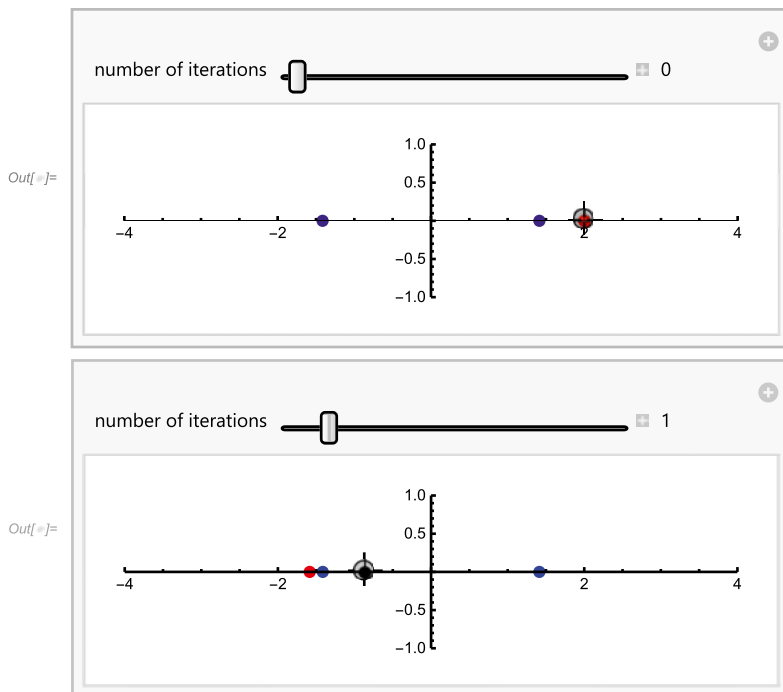


Figure 2.1



In the example above the sequence obtained from the recurrence eventually becomes monotonic, making it possible to use the Monotone Convergence Theorem to conclude the existence of a limit. We will now consider a case where this is not true.

## 2.2 Example: the Fibonacci sequence

The famous Fibonacci sequence is given by

$$a_1 = 1, \quad a_2 = 1, \quad a_{n+1} = a_n + a_{n-1}.$$

We can generate its terms with

```
In[.]:= RecurrenceTable[{a[n + 1] == a[n] + a[n - 1], a[1] == 1,
  a[2] == 1}, a, {n, 1, 14}]
```

```
Out[.]:= {1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377}
```

We can also solve the recurrence with `RSolve`:

```
In[.]:= RSolve[{a[1] == 1, a[2] == 1, a[n] == a[n - 1] +
  a[n - 2]}, a[n], n]
```

```
Out[.]:= {{a[n] -> Fibonacci[n]}}
```

```
In[.]:= First[a[n] /. RSolve[{a[1] == 1, a[2] == 1,
  a[n] == a[n - 1] + a[n - 2]}, a[n], n]]
```

```
Out[.]:= Fibonacci[n]
```

Mathematica® returns a built-in symbol `Fibonacci[n]`. Below we use notation  $F_n$  for our sequence. We can expand it using

```
In[.]:= FunctionExpand[Fibonacci[n]]
```

```
Out[.]:= 
$$\frac{\left(\frac{1}{2}(1 + \sqrt{5})\right)^n - \left(\frac{2}{1 + \sqrt{5}}\right)^n \cos[n\pi]}{\sqrt{5}}$$

```

Note that this can be simplified under the assumption that  $n$  is either even or odd:

```
In[.]:= Simplify[FunctionExpand[Fibonacci[n]],
  Assumptions -> Element[n/2, Integers]]
```

```
Out[.]:= 
$$\frac{\left(\frac{1}{2}(1 + \sqrt{5})\right)^n - \left(\frac{2}{1 + \sqrt{5}}\right)^n}{\sqrt{5}}$$

```

```
In[.]:= Simplify[FunctionExpand[Fibonacci[n]],
  Assumptions -> Element[(n + 1)/2, Integers]]
```

```
Out[.]:= 
$$\frac{\left(\frac{2}{1 + \sqrt{5}}\right)^n + \left(\frac{1}{2}(1 + \sqrt{5})\right)^n}{\sqrt{5}}$$

```

The sequence clearly diverges monotonically to  $\infty$ :

**In[.]**:= DiscreteLimit[Fibonacci[n], n -> Infinity]

**Out[.]**:=  $\infty$

But the sequence

$$s_n = \frac{F_{n+1}}{F_n}$$

has an interesting limit:

**In[.]**:= DiscreteLimit[Fibonacci[n + 1]/Fibonacci[n],  
n -> Infinity]

**Out[.]**:=  $\frac{1}{2}(1 + \sqrt{5})$

The sequence  $s_n$  is obviously given by the recurrence equation

$$s_1 = 1, \quad s_n = 1 + \frac{1}{s_{n-1}}.$$

It can be easily solved by RSolve

**In[.]**:= First[s[n] /. RSolve[{s[n]\*s[n - 1] == s[n - 1] + 1,  
s[1] == 1}, s[n], n]]

**Out[.]**:= (Fibonacci[n] + Fibonacci[1 + n] - LucasL[n]  
- LucasL[1 + n])/(Fibonacci[1 + n] - LucasL[1 + n])

The answer involves another built-in function LucasL:

**In[.]**:= FunctionExpand[LucasL[n]]

**Out[.]**:=  $\left(\frac{1}{2}(1 + \sqrt{5})\right)^n + \left(\frac{2}{1 + \sqrt{5}}\right)^n \cos[n\pi]$

Of course we can generate the terms of the sequence  $\{s_n\}$  using RecurrenceTable or FixedPointList:

**In[.]**:= N[RecurrenceTable[{s[n]\*s[n - 1] == s[n - 1] + 1,  
s[1] == 1}, s, {n, 1, 8}]]

**Out[.]**:= {1., 2., 1.5, 1.66667, 1.6, 1.625, 1.61538, 1.61905}

**In[.]**:= InputForm[FixedPointList[1 + 1/#1 &, 1.]]

**Out[.]**//InputForm=

{1., 2., 1.5, 1.6666666666666665, 1.6, 1.625, 1.6153846153846154, 1.619047619047619,  
1.6176470588235294, 1.6181818181818182, 1.6179775280898876, 1.6180555555555556,  
1.6180257510729614, 1.6180371352785146, 1.6180327868852458, 1.618034447821682,  
1.618033813400125, 1.6180340557275543, 1.6180339631667064, 1.618033985218035,  
1.618033985017358, 1.6180339901755971, 1.6180339882053252, 1.6180339889579018,  
1.6180339886704433, 1.6180339887802426, 1.618033988738303, 1.6180339887543225,  
1.6180339887482038, 1.6180339887505406, 1.6180339887496482, 1.618033988749989,  
1.618033988749859, 1.6180339887499087, 1.6180339887498896, 1.618033988749897,  
1.618033988749894, 1.6180339887498951, 1.618033988749895}

```
In[.]:= Length[%]
Out[.]:= 39
```

This sequence has reached a fixed point after 38 steps. We know that the exact sequence will never reach its limit, but if we take any approximation up to some precision  $p$ , eventually all the elements of the sequence will have the same first  $p$  digits. So the numbers will be equal to each other up to this precision  $p$  and Mathematica<sup>®</sup> will decide that it has reached the fixed point. But we know that this is not a real fixed point. Suppose we want to get a longer sequence by using more precision. The above computation is done using the so called machine arithmetic, which is done using `MachinePrecision`, whose value depends on the type of CPU used. As mentioned earlier, Mathematica<sup>®</sup> can also have arbitrary precision arithmetic. Recall (see the previous example in Section 2.1) that the easiest way to enter a number with a given precision  $p$  is `N[a, p]`, where  $a$  is an exact number. For example the number

```
In[.]:= N[1, 20]
Out[.]:= 1.00000000000000000000
```

is 1 with 20 correct digits. If all numbers in a formula have non-machine precision, Mathematica<sup>®</sup> performs computations using its special model of arithmetic, which actually keeps track of the precision. Note however that the precision of an answer can either be higher or lower than the precision of the input. In fact, this is what happens in our example, the `Precision` of the result of each iteration is actually higher than that of the previous one:

```
In[.]:= rt = RecurrenceTable[{s[n + 1] == 1 + 1/s[n],
    s[1] == N[1, 1]}, s, {n, 1, 9}]
Out[.]:= {1., 2., 1.5, 1.7, 1.60, 1.63, 1.615, 1.619, 1.6176}

In[.]:= Map[Precision, rt]
Out[.]:= {1., 1.30103, 1.77815, 2.17609, 2.60206, 3.01703,
    3.43616, 3.8537, 4.27184}
```

We see that the precision is actually getting larger, hence a fixed point will never be reached! Hence, the next computation will never finish:

```
In[.]:= Length[FixedPointList[1 + 1/#1 &, N[1, 1]]]
Out[.]:= $Aborted
```

We can stop the computation above after inserting the third argument of the function `FixedPointList`:

```
In[.]:= Length[FixedPointList[1 + 1/#1 &, N[1, 1], 500]]
Out[.]:= 501
```

However, we can make sure that a fixed point is reached by forcing Mathematica<sup>®</sup> to perform all iterations using fixed precision (rather than its own variable precision arithmetic). The idea is to make the global variables `$MinPrecision` and `$MaxPrecision`

sion equal to the same number, which will be our chosen fixed precision. By default these variables have the values

```
In[.]:= {$MinPrecision, $MaxPrecision}
Out[.]:= {0, ∞}
```

We can set them to other values; however, since we want to change them only temporarily, we use the construction `Block`, which is often used for such a purpose:

```
In[.]:= Length[Block[{$MinPrecision = 20, $MaxPrecision = 20},
FixedPointList[1 + 1/#1 & , N[1, 20]]]]
Out[.]:= 49
```

We can, of course, easily calculate the possible “candidates” for the limit of the sequence

```
In[.]:= s /. Solve[s == 1 + 1/s, s, Reals]
Out[.]:= {1/2(1 - √5), 1/2(1 + √5)}
```

Note that neither  $\infty$  nor  $-\infty$  satisfies the equation. However proving the existence of a limit is this time harder (see below).

As in the previous example we can study how the sequence behaves by using `Manipulate`. Although the sequence does appear to converge to  $(1 + \sqrt{5})/2$  for all starting points except for  $(1 - \sqrt{5})/2$  (which is the other fixed point), the sequence does not appear to become monotone. Let us confirm this in our case with  $s_1 = 1$ :

```
In[.]:= Sign[Differences[RecurrenceTable[{s[n + 1] ==
1 + 1/s[n], s[1] == 1}, s, {n, 1, 10}]]]
Out[.]:= {1, -1, 1, -1, 1, -1, 1, -1, 1}
```

The function `Differences` applied to a list gives the list of differences between successive elements. The function `Sign` returns the sign of a number, i. e.,  $+1$  for a positive number,  $-1$  for a negative number and  $0$  for  $0$ . So we can see that the signs of the differences between successive elements keep changing and the sequence is not monotonic. This suggests the following idea to prove the convergence.

Let us consider two subsequences of  $\{s_n\}$ , the subsequence of elements with an even index  $\{s_{2n}\}$  and the subsequence of elements with an odd index  $\{s_{2n+1}\}$ . We will prove that both of these subsequences converge to the same limit. Then we will use the following simple lemma.

**Lemma 4.** *Let  $\{a_n\}$  be a sequence of real numbers and suppose its subsequences  $\{a_{2n}\}$  and  $\{a_{2n+1}\}$  both have the same limit  $g$ . Then  $g$  is the limit of  $\{a_n\}$ .*

So to prove that  $\{s_{2n}\}$  converges and that  $\{s_{2n+1}\}$  converges, we will prove that  $\{s_{2n+1}\}$  is increasing and  $\{s_{2n}\}$  is decreasing. In other words, we want to show that  $s_{2n+1} - s_{2n-1} > 0$  and  $s_{2n+2} - s_{2n} < 0$ . We proceed by induction on  $n$ . We can check directly that the result is true for  $n = 1$ :

**In[.]**:= RecurrenceTable[{s[n + 1] == 1 + 1/s[n], s[1] == 1},  
s, {n, 1, 4}]

**Out[.]**:= {1, 2,  $\frac{3}{2}$ ,  $\frac{5}{3}$ }

So suppose now that it is true for  $n = k$ . We have

$$s_{2k+3} - s_{2k+1} = \frac{1}{s_{2k+2}} - \frac{1}{s_{2k}} > 0, \quad s_{2k+4} - s_{2k+2} = \frac{1}{s_{2k+3}} - \frac{1}{s_{2k+1}} < 0.$$

Hence both sequences are monotonic by induction. Note also that  $s_n$  is always bounded, since  $1 \leq s_n \leq 2$  by induction (with the assumption that  $s_1 = 1$ ). Hence both  $s_{2n}$  and  $s_{2n+1}$  have limits, say,  $a$  and  $b$ . The equation  $s_{2n+1} = 1 + 1/s_{2n}$  implies that  $b = 1 + 1/a$ . The equation  $s_{2n+2} = 1 + 1/s_{2n+1}$  implies  $a = 1 + 1/b$ . Solving these together gives

**In[.]**:= {a, b} /. Solve[{b == 1 + 1/a, a == 1 + 1/b}, {a, b}]

**Out[.]**:= {{ $\frac{1}{2}(1 - \sqrt{5})$ ,  $\frac{1}{2}(1 - \sqrt{5})$ }, { $\frac{1}{2}(1 + \sqrt{5})$ ,  $\frac{1}{2}(1 + \sqrt{5})$ }}

Hence  $a = b$  and the sequence  $\{s_n\}$  is convergent.



## 3 Series

In this chapter we define the concept of series and discuss their convergence. We consider various convergence tests and give several examples. We also discuss a number of related questions, for instance, Riemann's theorem on conditionally convergent series, divergent series and power series.

### 3.1 Sequences and series

We will model sequences by lists which can be thought of as sequences that are constant after a certain number of terms. Since Mathematica<sup>®</sup> cannot deal with symbolic lengths, we will fix  $n$  as some small integer but it will be clear that everything we say will work with an arbitrary  $n$ .

```
In[.] := n = 4;  
In[.] := cc = Table[c[i], {i, 1, n}]  
Out[.] := {c[1], c[2], c[3], c[4]}
```

Mathematica<sup>®</sup> has two useful functions (in some sense almost inverse to each other) that take lists as arguments and which naturally extend to sequences. The first one is the function `Differences`, which has already appeared in Chapter 2 and which returns the list of differences between successive elements:

```
In[.] := dd = Differences[cc]  
Out[.] := {-c[1] + c[2], -c[2] + c[3], -c[3] + c[4]}
```

A sequence convergent to 0 is called a *null sequence*. We have the following trivial but useful lemma.

**Lemma 5.** *If a sequence  $a_n$  is convergent, then its difference sequence is a null sequence.*

Results about limits of general sequences can often be reduced to results about null sequences, which often can be proved more easily than for general sequences. Here is a very useful fact about null sequences, which follows directly from the definition of limits.

**Lemma 6.** *A sequence  $a_n$  is a null sequence if and only if the sequence  $|a_n|$  is a null sequence.*

The second useful function `Accumulate` is almost inverse to the function `Differences`:

```
In[.] := Accumulate[cc]  
Out[.] := {c[1], c[1] + c[2], c[1] + c[2] + c[3],  
          c[1] + c[2] + c[3] + c[4]}
```

<https://doi.org/10.1515/9783110590142-003>

For a sequence it gives a sequence of partial sums. Note that

```
In[.]:= Differences[Accumulate[cc]]
Out[.]:= {c[2], c[3], c[4]}
```

differs from the original sequence only by the first element; in other words, it is the original list without the first element:

```
In[.]:= Rest[cc]
Out[.]:= {c[2], c[3], c[4]}
```

If we perform the operations in the reverse order, we get

```
In[.]:= Accumulate[Differences[cc]]
Out[.]:= {-c[1] + c[2], -c[1] + c[3], -c[1] + c[4]}
```

This does not quite get us back to where we started but the following does:

```
In[.]:= Accumulate[Differences[cc]] + c[1]
Out[.]:= {c[2], c[3], c[4]}
```

or

```
In[.]:= Accumulate[Prepend[Differences[cc], c[1]]]
Out[.]:= {c[1], c[2], c[3], c[4]}
```

Since removing or adding a finite number of elements of a sequence does not make any difference to the limit, we can think of the functions `Differences` and `Accumulate` as essentially inverse to each other.

A series is often informally thought of as an infinite sum, but there are several risks in this approach and we do not recommend it. Instead, we define the notion of a series as in [7]. By a *series* we mean the sequence of partial sums of some sequence. If this sequence of partial sums has a limit, it is called the *sum of the series*. If the limit is a real number we say that the series is *convergent* to that number. Therefore, we clearly distinguish between two objects: the series and its sum. The latter may or may not exist. Unfortunately, it is customary to denote both the series and its sum by the same symbol  $\sum_{n=1}^{\infty} a_n$  and we have to be careful in interpreting the notation.

The partial sums of a series  $\sum_{n=0}^{\infty} a_n$  are simply the solutions of the recurrence equation  $s_1 = a_0$ ,  $s_{n+1} = s_n + a_n$ . Thus, instead of the function `Accumulate`, we can use the function `RecurrenceTable`:

```
In[.]:= Clear[n]
In[.]:= RecurrenceTable[{s[1] == a[0], s[n + 1] ==
    s[n] + a[n]}, s, {n, 1, 3}]
Out[.]:= {a[0], a[0] + a[1], a[0] + a[1] + a[2]}
```

It follows from Lemma 5 that a necessary condition for a series associated with  $a_n$  to converge is that  $a_n$  is a null sequence. However, we will soon see that the series of a null sequence is not always convergent (e. g., the harmonic series).



### 3.2 The functions Sum and NSum

Mathematica<sup>®</sup>'s function Sum has multiple uses. First of all, it can be used to simply add up a list of numbers, e. g.,

**In[.]**:= Sum[1, {10}]

**Out[.]**:= 10

The index in Sum can run over an arbitrary list:

**In[.]**:= Sum[i^2, {i, {2, 4, 7, 9, 11}}]

**Out[.]**:= 271

The function Sum is able to compute many finite sums explicitly, sometimes expressing them in terms of special functions. For example,

**In[.]**:= Sum[i^3, {i, 1, n}]

**Out[.]**:=  $\frac{1}{4}n^2(1+n)^2$

**In[.]**:= Sum[i, {i, 1, n, 2}]

**Out[.]**:=  $\left(1 + \text{Floor}\left[\frac{1}{2}(-1+n)\right]\right)^2$

**In[.]**:= Sum[1/k^2, {k, 1, n}]

**Out[.]**:= HarmonicNumber[n, 2]

This means that the sequence of partial sums of a sequence  $a_i$  can also be computed as

**In[.]**:= Table[Sum[a[i], {i, 1, k}], {k, 1, 3}]

**Out[.]**:= {a[1], a[1] + a[2], a[1] + a[2] + a[3]}

The sum of a series can be computed in Mathematica<sup>®</sup> simply by combining the functions Sum and Limit:

**In[.]**:= DiscreteLimit[Sum[1/k^2, {k, 1, n}], n -> Infinity]

**Out[.]**:=  $\frac{\pi^2}{6}$

In fact, one can obtain the result more simply:

**In[.]**:= Sum[1/k^2, {k, 1, Infinity}]

**Out[.]**:=  $\frac{\pi^2}{6}$

Note, however, that while in the case of convergent series there is no difference between the outputs above in Mathematica<sup>®</sup>, in the case of divergent series there is one. For example,

**In[.]**:= Sum[1/n, {n, 1, Infinity}]

... Sum: Sum does not converge.

**Out[.]**:=  $\sum_{n=1}^{\infty} \frac{1}{n}$

```
In[.]:= DiscreteLimit[Sum[1/n, {n, 1, k}], k -> Infinity]
Out[.]:= ∞
```

```
In[.]:= Sum[(-1)^n, {n, 1, Infinity}]
... Sum: Sum does not converge.
```

```
Out[.]:=  $\sum_{n=1}^{\infty} (-1)^n$ 
```

```
In[.]:= DiscreteLimit[Sum[(-1)^n, {n, 1, k}], k -> Infinity]
Out[.]:= Indeterminate
```

So we see that in these cases using an explicit `DiscreteLimit` gives us more information since it distinguishes between divergence to infinity and the other kind of divergence (non-existence of a limit).

The function `Sum` has an option `VerifyConvergence` which by default is set to `True`. It will therefore always try to verify convergence and will inform us if it can show that the series is not convergent, as in the example above.

Now let us look at the sum

```
In[.]:= Sum[1/n^n, {n, 1, Infinity}]
Out[.]:=  $\sum_{n=1}^{\infty} n^{-n}$ 
```

`Mathematica`<sup>®</sup> returns the original input unchanged. What can we conclude from this? We can conclude that one of two things happened. One possibility is that `Mathematica`<sup>®</sup> verified that the series is convergent but could not find any closed form expression for the sum. The other possibility is that `Mathematica`<sup>®</sup> could not decide whether the series is convergent or not. To distinguish between these possibilities we need to use the function `SumConvergence`, which will be discussed in greater detail in Section 3.6.

```
In[.]:= SumConvergence[1/n^n, n]
Out[.]:= True
```

Now we know that the series is convergent but `Mathematica`<sup>®</sup> cannot find any closed form formula. In such cases we can now use the function `NSum` to compute the numerical value of this sum to any desired precision:

```
In[.]:= NSum[1/n^n, {n, 1, Infinity}, WorkingPrecision -> 20]
Out[.]:= 1.2912859970626635404
```

Alternatively we can write

```
In[.]:= N[Sum[1/n^n, {n, 1, Infinity}], 20]
Out[.]:= 1.2912859970626635404
```

Here the function `Sum` passed the task of computation to `NSum`. For simple problems we can use the second approach, but for more complicated ones some of the options offered by the function `NSum` may be needed.

Of course there many series for which Mathematica<sup>®</sup> cannot decide whether they are convergent or not. For instance, in Section 3.5.5 we shall prove that the next series is convergent but Mathematica<sup>®</sup> cannot do it:

```
In[.]:= Sum[Sin[1/n]*Sin[n], {n, 1, Infinity}]
Out[.]:=  $\sum_{n=1}^{\infty} \sin\left[\frac{1}{n}\right] \sin[n]$ 
```

Again the function Sum returns the input. But this time Mathematica<sup>®</sup> cannot decide whether this series converges:

```
In[.]:= SumConvergence[Sin[1/n]*Sin[n], n]
Out[.]:= SumConvergence[Sin[1/n]*Sin[n], n]
```

If nevertheless we try to compute this sum numerically to a high degree of precision, i. e., we write N[Sum[Sin[1/n]\*Sin[n], {n, 1, Infinity}], 30], we will get a lot of warning messages from Mathematica<sup>®</sup>. This is because Mathematica<sup>®</sup> fails to notice that the series has non-constant signs and is using a wrong computational method. By changing the method, we can successfully compute the answer to any precision:

```
In[.]:= NSum[Sin[1/n]*Sin[n], {n, 1, Infinity},
WorkingPrecision -> 30, Method -> "AlternatingSigns"]
Out[.]:= 0.98629511964584094677185426431
```

Sometimes when Mathematica<sup>®</sup> is doing numerical computations with MachinePrecision, even if we know that the answer should be real, we may still get a complex number with a tiny imaginary part:

```
In[.]:= NSum[Sin[1/n]*(-1)^(101*n), {n, 1, Infinity}]
Out[.]:= -0.550797 - 1.565212x10-13 I
```

There are two ways to deal with this problem. We can either use the function Chop which replaces small numbers by zero:

```
In[.]:= Chop[NSum[Sin[1/n]*(-1)^(101*n), {n, 1, Infinity}]]
Out[.]:= -0.550797
```

or use extended precision in our calculation:

```
In[.]:= NSum[Sin[1/n]*(-1)^(101*n), {n, 1, Infinity},
WorkingPrecision -> 20]
Out[.]:= -0.550796848133929
```

### 3.3 Absolute convergence

We first introduce the notion of *absolute convergence*. We say that a series  $\sum_{n=1}^{\infty} a_n$  is *absolutely convergent* if the series  $\sum_{n=1}^{\infty} |a_n|$  is convergent. Note that when  $a$  is complex, then  $|a|$  is the modulus of  $a$ , which is a positive real number.

The following properties hold.

1. If  $\sum_{n=1}^{\infty} a_n$  is absolutely convergent, then  $\sum_{n=1}^{\infty} a_n$  is convergent.
2. A convergent series with terms of constant sign (either positive or negative) is absolutely convergent. If  $a_n > 0$  ( $a_n < 0$ ) for all  $n$ , then  $\sum_{n=1}^{\infty} a_n$  is convergent if and only if its sequence of partial sums is bounded above (below). When a series with positive (negative) terms is convergent we write  $\sum_{n=1}^{\infty} a_n < \infty$  ( $\sum_{n=1}^{\infty} a_n > -\infty$ ). We use this notation only for series whose terms have a constant sign.
3. If  $\sum_{n=1}^{\infty} a_n$  is absolutely convergent and  $\pi : \mathbb{N} \rightarrow \mathbb{N}$  is any bijection, then  $\sum_{n=1}^{\infty} a_{\pi(n)} = \sum_{n=1}^{\infty} a_n$ .

As we will see in Section 3.7, property 3 is not true for series which are not absolutely convergent.

The following tests can be used to show that the series is absolutely convergent. They can also sometimes be used to show that a series which is known to be not absolutely convergent is actually divergent.

*D'Alembert's ratio test.* Assume that  $a_n \neq 0$ . If  $\lim_{n \rightarrow \infty} |a_{n+1}/a_n| = g \in \mathbb{R} \cup \{\infty, -\infty\}$ , then the following statements hold.

- If  $g < 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is absolutely convergent.
- If  $g > 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is divergent.

*Cauchy's root test.* If  $\lim_{n \rightarrow \infty} \sup |a_n|^{1/n} = g \in \mathbb{R} \cup \{\infty, -\infty\}$ , then the following statements hold.

- If  $g < 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is absolutely convergent.
- If  $g > 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is divergent.

The root test is actually stronger than the ratio test: one can show that in all cases when the ratio test works, so does the root test, but one can construct examples in which d'Alembert's ratio test does not work but the Cauchy root test does (for instance,  $\sum_{n=1}^{\infty} 2^{(-1)^n} 2^{-n}$ ). Note that the root test (and hence also the ratio test) fails when  $g = 1$ .

D'Alembert's ratio test and Cauchy's root test are implemented in the option Method in Mathematica<sup>®</sup>'s function SumConvergence, which we will discuss later on in this chapter.

### 3.4 Convergence of series with terms of constant signs

The theory of convergent series is traditionally divided into two parts. The first part is concerned with series whose terms have the same signs. The second part is concerned with series that have infinitely many of both positive and negative terms. Note that, just like in the case of a sequence, convergence of a series is unaffected by ignoring a finite number of terms. However, unlike the case of the limit of a sequence, removing a finite number of terms from a series may change the value of its sum (if that value is finite).

In this section we state without proof some basic facts about convergence of series with constant signs and describe several tests of convergence. The proofs can be found in [12] and [14].

The following two tests are extremely useful for proving convergence of series of positive terms by hand but are generally unsuitable for present day computer algebra programs. The reason is that they involve comparing a given series (which we are trying to test) with another one, about which we must already know whether it is convergent or not. Choosing a suitable series involves making an “educated guess”, something that humans are still better at than computers.

*The comparison test.* Let  $a_n, b_n > 0$  and suppose that  $a_n \leq b_n$  for all  $n \in \mathbb{N}$ . Then the following statements hold.

- If  $\sum_{n=1}^{\infty} b_n$  is convergent, then  $\sum_{n=1}^{\infty} a_n$  is convergent.
- If  $\sum_{n=1}^{\infty} a_n$  is divergent, then  $\sum_{n=1}^{\infty} b_n$  is divergent.

*The limit comparison test.* Let  $a_n, b_n > 0$  and suppose that  $\lim_{n \rightarrow \infty} a_n/b_n = c$ . Then the following statements hold.

- If  $c > 0$ , then  $\sum_{n=1}^{\infty} b_n$  is convergent if and only if  $\sum_{n=1}^{\infty} a_n$  is convergent (in this case the two sequences  $\{a_n\}$  and  $\{b_n\}$  are said to be similar).
- If  $c = 0$ , then the convergence of  $\sum_{n=1}^{\infty} b_n$  implies the convergence of  $\sum_{n=1}^{\infty} a_n$ .
- If  $c = \infty$ , then the divergence of  $\sum_{n=1}^{\infty} b_n$  implies the divergence of  $\sum_{n=1}^{\infty} a_n$ .

The next two tests are implemented in the option Method in Mathematica<sup>®</sup>,s function SumConvergence.

*Raabe’s test.* Suppose  $a_n > 0$  and  $\lim_{n \rightarrow \infty} n(a_n/a_{n+1} - 1) = g \in \mathbb{R} \cup \{\infty, -\infty\}$ . Then the following statements hold.

- If  $g > 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is convergent.
- If  $g < 1$ , then the series  $\sum_{n=1}^{\infty} a_n$  is divergent.

The last test uses the concept of an improper integral of a continuous function, which will be considered later. However, we shall state the test now (see also Chapter 7) because it is one of the most effective tests at Mathematica<sup>®</sup>,s disposal (due to the fact that Mathematica<sup>®</sup> is much better than an average human mathematician at integration).

*Integral test.* Let  $f$  be a continuous, positive, decreasing function of  $x$  for  $x \geq 1$  and  $\lim_{x \rightarrow \infty} f(x) = 0$ . Then  $\sum_{n=1}^{\infty} f(n)$  is convergent if and only if  $\int_1^{\infty} f(x)dx$  is convergent.

### 3.4.1 Example

Let us study for which values of the parameter  $a$  the following series  $\sum_{n=1}^{\infty} (5^{1/n} - 1)^a$  is convergent.

```
In[.]:= SumConvergence[(5^(1/n) - 1)^a, n, Assumptions ->
{Element[a, Reals]}]
```

```
Out[.]:= a > 1
```

Let us try to use the comparison test. We will see later that it is enough to deal with the case  $a = 1$  so let us consider it first. The trick is to choose the right series to compare. It is often useful to look at some graphs. We would like to find two positive integers  $p, q$  and constants  $c_1$  and  $c_2$  such that the graph of  $5^{1/n} - 1$  lies between that of  $c_1/n^p$  and  $c_2/n^q$  for sufficiently large  $n$ . We could use `Manipulate` to find candidates for such  $p$  and  $q$  but we will illustrate this only with the usual “static” graph:

```
In[.]:= DiscretePlot[{5^(1/n) - 1, 1/n, 3/n}, {n, 1, 20},
PlotMarkers -> {Automatic, 9}]
```

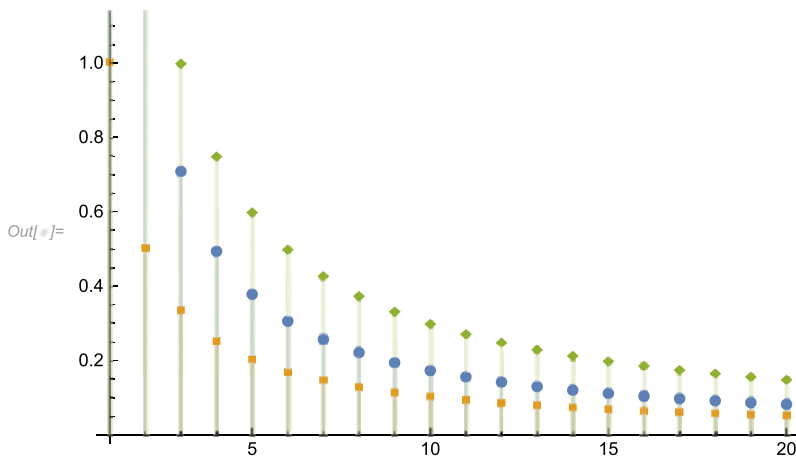


Figure 3.1

So the graph suggests that  $1/n < 5^{1/n} - 1 < 3/n$  for all  $n > 1$ , which we can try to verify by the function `Reduce`:

```
In[.]:= Reduce[1/n < 5^(1/n) - 1 < 3/n && n > 1, n, Integers]
```

```
...Reduce: Unable to decide whether numeric quantity
```

```
Log[5] + 3ProductLog[-Log[5]/(3*5^(1/3))] is equal to zero.
```

```
Out[.]:= n ∈ ℤ && n ≥ 2
```

Mathematica<sup>®</sup> uses advanced methods to prove this inequality. In this particular case in the proof there appeared some identity that Mathematica<sup>®</sup> could not prove exactly, but verified by using high precision computations. This means that the identity almost certainly holds and we can treat this kind of result as valid. Evaluating the same expression again does not give a message as Mathematica<sup>®</sup> from now on assumes that the identity holds (until we quit the session):

**In[.]**:= Reduce[ $1/n < 5^{1/n} - 1 < 3/n$  &&  $n > 1$ ,  $n$ , Integers]

**Out[.]**:=  $n \in \mathbb{Z}$  &&  $n \geq 2$

Mathematica<sup>®</sup> says that almost certainly the inequality is true for  $n \geq 2$ . Hence,  $1/n^a < (5^{1/n} - 1)^a < 3^a/n^a$  and the series  $\sum_{n=1}^{\infty} (5^{1/n} - 1)^a$  is convergent for  $a > 1$  and divergent for  $a \leq 1$ .

The limit comparison test works even quicker since we do not need to bother trying to find suitable  $p$  and  $q$ . However, this requires the ability to compute limits:

**In[.]**:= Discretelimit[ $(5^{1/n} - 1)^a/(1/n^a)$ ,  $n \rightarrow$  Infinity,  
Assumptions  $\rightarrow$  { $a > 0$ }]

**Out[.]**:=  $\text{Log}[5]^a$

We shall see how this limit can be computed by hand later, when we consider the Taylor series. However, somewhat surprisingly the inequalities we use to apply the comparison test can be proved rather easily (which is again not how Mathematica<sup>®</sup> proves them).

Consider the inequality  $5^{1/n} - 1 < 3/n$ . Transforming we easily see that it is equivalent to  $5 < (1 + 3/n)^n$ . Now let us write the right hand side as  $((1 + 1/(n/3))^{n/3})^3$ . We have already seen that this sequence is increasing (in fact its limit is  $e^3$ ). For  $n = 3$  its  $n$ -th term is 8, hence it and all the others are larger than 5 (actually even for  $n = 2$  the inequality already holds but we only need to show that it holds for some  $n$ ).

Next let us consider the inequality  $1/n < 5^{1/n} - 1$ . It is equivalent to  $(1 + 1/n)^n < 5$ . We know that the sequence on the left hand side is monotonically increasing and is always less than 3. Hence it is less than 5.

## 3.5 Convergence of series with terms of non-constant signs

In this section we will discuss some methods of proving convergence of series whose terms do not have constant sign.

### 3.5.1 Grouping of terms

A simple method, which is useful when dealing with convergence of series with non-constant signs is a method of grouping of terms. Namely, consider the relationship between the following two series:

$$\sum_{i=0}^{\infty} a_i = a_0 + a_1 + a_2 + a_3 + \dots \quad (3.4)$$

and

$$\sum_{i=0}^{\infty} (a_{2i+1} + a_{2i}) = (a_0 + a_1) + (a_2 + a_3) + \dots \quad (3.5)$$

First note that if (3.4) is convergent, so is (3.5). This can be seen by looking at the sequences of partial sums and observing that the second sequence is a subsequence of the first. Since any subsequence of a convergent sequence is convergent, the result follows. Now suppose that we know that the second series is convergent. Then it does not in general follow that the first also is. For example, consider the series  $(1-1)+(1-1)+\dots$ . This is just the series  $0+0+\dots$ , so its partial sums are all 0 and the limit of the (constant) sequence of zeros is 0. But the series  $1-1+1-1+\dots = \sum_{n=0}^{\infty} (-1)^n$  is divergent, since its sequence of partial sums is  $1, 0, 1, \dots$

However, suppose now we know that (3.5) is convergent and also that the sequence  $\{a_n\}$  is a null sequence (that is, the necessary condition for convergence of (3.4) is satisfied). In this case from the convergence of (3.5) follows the convergence of (3.4). Indeed, consider the sequence of partial sums of (3.4)  $\{s_n\}$ , where  $s_n = \sum_{i=0}^n a_i$ . We have already seen that a sequence is convergent if and only if its two subsequences of even indexed terms and odd indexed terms are convergent to the same limit (Lemma 4 in Chapter 2). The sequence  $\{s_{2n}\}$  is just the sequence of terms of (3.5), hence it is convergent. The sequence  $\{s_{2n+1}\}$  also converges to the same limit because  $s_{2n+1} = s_{2n} + a_{2n+1}$  and  $a_{2n+1} \rightarrow 0$ . (We see that we only needed the assumption that  $a_{2n+1} \rightarrow 0$  rather than  $a_n \rightarrow 0$ .)

Informally speaking we have shown that if the necessary condition for convergence of a series, i. e., the condition  $a_n \rightarrow 0$ , is satisfied, we can “group” and “un-group” the series into pairs without affecting convergence. The same argument shows that this is also true about grouping the terms (or “inserting brackets”) into groups of terms of arbitrary fixed length (e. g., 3):  $(a_0+a_1+a_2)+(a_3+a_4+a_5)+(a_6+a_7+a_8)+\dots$ . In fact, it can even be proved that we can group a series into segments of various lengths, as long as their lengths are bounded, e. g., all less than 30. However, we shall not need these results here.

### 3.5.2 Example

Let us show that the series  $\sum_{n=1}^{\infty} (-1)^{n+1}/\sqrt{n}$  is convergent. We simply group the terms in pairs and observe that

$$\begin{aligned} \text{In[ ]:} &= \text{Refine[Together[(-1)^(2*k - 1)/Sqrt[2*k - 1]} \\ &\quad + (-1)^(2*k)/Sqrt[2*k]], \text{Element}[k, \text{Integers}]] \\ \text{Out[ ]:} &= \frac{-2\sqrt{k} + \sqrt{2}\sqrt{-1 + 2k}}{2\sqrt{k}\sqrt{-1 + 2k}} \end{aligned}$$

This expression is always negative. It can actually be reduced to a “simpler” form by “rationalizing” the numerator. Mathematica<sup>®</sup>’s functions `Simplify` or `FullSimplify` will not do this automatically, but we already know that we can try to change the default `ComplexityFunction` as follows:



```
In[.]:= g[k_]:=FullSimplify[(Sqrt[2]*Sqrt[2*k - 1] - 2*
      Sqrt[k])/(2*Sqrt[k]*Sqrt[2*k - 1]), ComplexityFunction
      -> Function[x, Count[Numerator[x], _Power, Infinity]]]
```

```
Out[.]:= 
$$\frac{1}{\sqrt{k}(\sqrt{2} - 2\sqrt{2}k - 2\sqrt{k}\sqrt{-1 + 2k})}$$

```

```
In[.]:= Simplify[Sign[g[k]], Assumptions ->
      {Element[k, Integers], k > 0}]
```

```
Out[.]:= -1
```

Since

```
In[.]:= DiscreteLimit[g[k]/(1/k^(3/2)), k -> Infinity]
```

```
Out[.]:= 
$$-\frac{1}{4\sqrt{2}}$$

```

the series is convergent by the limit comparison test.

### 3.5.3 Abel's summation formula

First, recall the definition of the inner or dot product. We can think of it as the product of two lists of the same length (or vectors of the same dimension):

```
In[.]:= l[a_, n_] := Table[a[i], {i, 1, n}]
```

```
In[.]:= l[a, 3]
```

```
Out[.]:= {a[1], a[2], a[3]}
```

```
In[.]:= l[a, 3] . l[b, 3]
```

```
Out[.]:= a[1] b[1] + a[2] b[2] + a[3] b[3]
```

```
In[.]:= Dot[l[a, 3], l[b, 3]]
```

```
Out[.]:= a[1] b[1] + a[2] b[2] + a[3] b[3]
```

In fact, Mathematica<sup>®</sup>'s function `Dot` is a special case of a more general function called `Inner`, but as we will not need it here, we will not say any more about it.

It is important not to omit the dot, because otherwise one gets a very different product:

```
In[.]:= l[a, 3] l[b, 3]
```

```
Out[.]:= {a[1] b[1], a[2] b[2], a[3] b[3]}
```

The dot is also necessary when multiplying matrices; without it we get a very different answer, which is not the usual product of matrices:

```
In[.]:= {{a, b}, {c, d}}.{{e, f}, {g, h}}
```

```
Out[.]:= {{a e + b g, a f + b h}, {c e + d g, c f + d h}}
```

```
In[.]:= {{a, b}, {c, d}} {{e, f}, {g, h}}
```

```
Out[.]:= {{a e, b f}, {c g, d h}}
```

We will now consider two lists, such that the length of the first one is one less than the length of the second one, say, 4 and 5:

```
In[.]:= list1 = 1[a, 4]
Out[.]:= {a[1], a[2], a[3], a[4]}

In[.]:= list2 = 1[b, 5]
Out[.]:= {b[1], b[2], b[3], b[4], b[5]}
```

Note that

```
In[.]:= Length[Differences[list2]]
Out[.]:= 4

In[.]:= Length[Accumulate[list1]]
Out[.]:= 4
```

This means that it makes sense to consider the dot product of Differences[list2] and Accumulate[list1]:

```
In[.]:= Collect[Expand[Differences[list2] .
    Accumulate[list1]], a[_]]
Out[.]:= a[1] (-b[1] + b[5]) + a[2] (-b[2] + b[5]) +
    a[3] (-b[3] + b[5]) + a[4] (-b[4] + b[5])
```

It suggests the following formula:

```
In[.]:= Simplify[Differences[list2] . Accumulate[list1]
    == -Most[list2] . list1 + Total[list1]*Last[list2]]
Out[.]:= True
```

Rearranging and rewriting in usual mathematical (rather than in Mathematica<sup>®</sup>'s) notation we get *Abel's summation formula*, which is easy to prove by induction:

$$\sum_{i=1}^n a_i b_i = \sum_{k=1}^n \left( \sum_{i=1}^k a_i \right) (b_k - b_{k+1}) + b_{n+1} \sum_{i=1}^n a_i. \quad (3.6)$$

### 3.5.4 Dirichlet's and Abel's tests

Suppose that we are given two sequences  $\{a_n\}$  and  $\{b_n\}$ . What are the weakest conditions that we need to impose on both so that we can conclude that the “inner product” (or the “dot product”)  $\sum_{n=1}^{\infty} a_n b_n$  is convergent? For example, if  $a_n > 0$  and  $b_n > 0$  for all  $n$ , and  $\sum_{n=1}^{\infty} a_n < \infty$ ,  $\sum_{n=1}^{\infty} b_n < \infty$ , then it is easy to see that  $\sum_{n=1}^{\infty} a_n b_n < \infty$ . Indeed, for a series of positive terms, convergence is equivalent to boundedness of its sequence of partial sums. The conclusion follows from the inequality  $\sum_{i=1}^m a_i b_i \leq (\sum_{i=1}^m a_i)(\sum_{i=1}^m b_i)$ , which follows simply by expanding the right hand side. But the result is not true in general. For example, let us take  $a_n = (-1)^{n+1}/\sqrt{n}$ ,  $b_n = (-1)^{n+1}/\sqrt{n}$ . We have

**In[.]**:= SumConvergence[ $(-1)^{(n+1)}(1/\text{Sqrt}[n])$ , n]

**Out[.]**:= True

**In[.]**:= SumConvergence[ $(-1)^{(n+1)}(1/\text{Sqrt}[n])*(-1)^{(n+1)}(1/\text{Sqrt}[n])$ , n]

**Out[.]**:= False

The second result is just the fact that the harmonic series is divergent.

We can think of Abel's summation formula (3.6) as an identity involving sequences. On the left hand side we have the sequence of partial sums of the sequence  $\{a_i b_i\}$ , on the right hand side there is the sum of two sequences. Thus, if we find a set of conditions that make both sequences on the right hand side converge, so will the sequence on the left.

Consider first the sequence  $\sum_{k=1}^n (\sum_{i=1}^k a_i)(b_k - b_{k+1})$  (the dot product of the sequence of partial sums of one sequence and the difference sequence of the other). We will always assume that the sequence  $\{b_n\}$  is monotone, so the differences all have the same sign. Since  $\sum_{k=1}^n (b_k - b_{k+1}) = b_1 - b_{n+1}$ , this is convergent if  $\lim_{n \rightarrow \infty} b_n \in \mathbb{R}$ . Since the series of differences has constant signs it is absolutely convergent. Now suppose that all the partial sums  $\sum_{i=1}^k a_i$  are bounded by  $M$ . Then

$$\sum_{k=1}^n \left| \left( \sum_{i=1}^k a_i \right) (b_k - b_{k+1}) \right| \leq \sum_{k=1}^n M |b_k - b_{k+1}| = M \sum_{k=1}^n |b_k - b_{k+1}|,$$

which is bounded. Hence  $\sum_{k=1}^n (\sum_{i=1}^k a_i)(b_k - b_{k+1})$  is absolutely convergent. Now consider the other term  $b_{n+1} \sum_{i=1}^n a_i$ . From the fact that the sequence  $\{b_{n+1}\}$  is convergent and  $\sum_{i=1}^n a_i$  is bounded, it does not follow that  $b_{n+1} \sum_{i=1}^n a_i$  is convergent. We need to make a stronger assumption. One possibility is to assume that  $\lim_{n \rightarrow \infty} b_n = 0$  and  $\sum_{i=1}^n a_i$  are bounded. In this case the term  $b_{n+1} \sum_{i=1}^n a_i$  also converges (to 0) and we can conclude that  $\sum_{i=1}^n a_i b_i$  converges. If we only assume that  $\lim_{n \rightarrow \infty} b_n = g \in \mathbb{R}$ , but  $g$  is not necessarily 0, we need a stronger assumption about  $\{a_n\}$ , namely, that the sequence  $\sum_{i=1}^n a_i$  converges, that is, the series  $\sum_{i=1}^{\infty} a_i$  is convergent. Thus we have proved Dirichlet's and Abel's criteria (tests) for the convergence of inner product of sequences.

**Theorem 7.** *Suppose we have two sequences  $\{a_n\}$  and  $\{b_n\}$ , where  $\{b_n\}$  is monotone convergent to  $g$ . Suppose also that one of the following conditions holds.*

*Dirichlet's test. The partial sums  $\sum_{i=1}^n a_i$  are bounded and  $g = 0$ .*

*Abel's test. The series  $\sum_{i=1}^{\infty} a_i$  is convergent.*

*Then  $\sum_{i=1}^{\infty} a_i b_i$  is convergent.*

Its easy to show that one can deduce the Abel test from the Dirichlet test.

One well-known consequence of Abel's test is the Leibniz test for convergence.

*Leibniz's test.* Let  $\{c_n\}$  be a monotonic null sequence (i. e.,  $\lim_{n \rightarrow \infty} c_n = 0$ ). Then  $\sum_{n=0}^{\infty} (-1)^n c_n$  converges.

Indeed, we simply take in the statement of the Dirichlet test  $a_n = (-1)^n$  and  $b_n = c_n$ . Clearly,  $\sum_{i=1}^n a_i$  are bounded and  $\{b_n\}$  is a monotonic null sequence.

The Dirichlet test is, of course, more general than the Leibniz test. For example, it applies to examples such as

**In[.]:** SumConvergence[(-1)^(n\*((n + 1)/2))/n, n]

**Out[.]:** True

**In[.]:** SumConvergence[(-1)^Quotient[n, 5]/n, n]

**Out[.]:** True

However,

**In[.]:** SumConvergence[(-1)^n^2015/n, n]

**Out[.]:** SumConvergence[ $\left[\frac{(-1)^{n^{2015}}}{n}, n\right]$

Here we need to observe something that Mathematica® fails to notice (this could be changed in future versions), namely, that

**In[.]:** FullSimplify[Mod[n^2015, 2], Element[n, Integers]]

**Out[.]:** Mod[n, 2]

That means that Mathematica® should simplify  $(-1)^{n^{2015}}/n$  to  $(-1)^n/n$ . That it does not do so in version 11.3 we consider an omission. In each case the series can be thought of as a “dot product” of two sequences satisfying the conditions of the Dirichlet test. Sometimes such a decomposition and particularly the fact that one of the sequences has bounded partial sums can be difficult to notice.

### 3.5.5 Example

Consider  $\sum_{n=1}^{\infty} \sin(nx)/n$  and suppose that  $x$  is not an integer multiple of  $\pi$  (otherwise all the terms are zero). Testing for convergence we get:

**In[.]:** SumConvergence[Sin[n\*x]/n, n, Assumptions ->

Element[x, Reals]]

**Out[.]:** True

It is not obvious that one can use the Dirichlet test. The series can be regarded as the dot product of two sequences  $\{1/n\}$  and  $\{\sin(nx)\}$ , but does the latter have bounded partial sums? We can check this with Mathematica®:

**In[.]:** Sum[Sin[n\*x], {n, 1, m}]

**Out[.]:** Csc[ $\frac{x}{2}$ ] Sin[ $\frac{mx}{2}$ ] Sin[ $\frac{1}{2}(1 + m)x$ ]

Since

$$\left| \csc\left(\frac{x}{2}\right) \sin\left(\frac{mx}{2}\right) \sin\left(\frac{1}{2}(m+1)x\right) \right| \leq \left| \csc\left(\frac{x}{2}\right) \right| = \left| \frac{1}{\sin(x/2)} \right|$$

is independent of  $m$ , the partial sums are bounded for fixed  $x$ .

The trigonometric identity

$$\sum_{n=1}^m \sin(nx) = \csc\left(\frac{x}{2}\right) \sin\left(\frac{mx}{2}\right) \sin\left(\frac{1}{2}(m+1)x\right)$$

can be proved by purely trigonometric means (we leave this as an exercise for those readers interested in elementary trigonometry) but the easiest way is by using Euler's formula in complex analysis relating the exponential and trigonometric functions. This can be obtained in **Mathematica**<sup>®</sup> in several ways, e. g.,

```
In[.]:= ComplexExpand[Exp[I*x]]
```

```
Out[.]:= Cos[x] + I Sin[x]
```

```
In[.]:= ExpToTrig[Exp[I*x]]
```

```
Out[.]:= Cos[x] + I Sin[x]
```

We now see that

$$\sum_{n=1}^m \sin(nx) = \operatorname{Im} \left( \sum_{n=1}^m (\cos(nx) + i \sin(nx)) \right) = \operatorname{Im} \left( \sum_{n=1}^m e^{inx} \right).$$

Clearly,

$$\sum_{n=1}^m e^{inx} = \frac{e^{ix}(e^{imx} - 1)}{e^{ix} - 1}$$

as the sum of  $m$  terms of a geometric progression. Hence,

```
In[.]:= Simplify[ComplexExpand[Im[(E^(I*x))*(-1 + E^(I*m*x))]/
(-1 + E^(I*x))]]]
```

```
Out[.]:= Csc[x/2] Sin[m*x/2] Sin[(1+m)*x/2]
```

Note that **Mathematica**<sup>®</sup> can actually find the sum of this series:

```
In[.]:= Sum[Sin[n*x]/n, {n, 1, Infinity}]
```

```
Out[.]:= 1/2 I (Log[1 - E^I*x] - Log[E^-I*x (-1 + E^I*x)])
```

**Mathematica**<sup>®</sup>'s answer unfortunately involves the complex  $i$ , which is inconvenient for some purposes. Even if we use `FullSimplify` with the assumption that  $x$  is real, we get the same answer:

```
In[.]:= Assuming[Element[x, Reals], FullSimplify[Sum[Sin[n*x]/n,
{n, 1, Infinity}]]]
```

```
Out[.]:= 1/2 I (Log[1 - E^I*x] - Log[E^-I*x (-1 + E^I*x)])
```

However, since we know that the sum of real numbers is surely real we can try to get an answer not involving complex  $i$  in a different way:

```
In[.]:= FullSimplify[ComplexExpand[Re[Sum[Sin[n*x]/n,
      {n, 1, Infinity}]], TargetFunctions -> {Im, Re}]]
Out[.]:=  $\frac{1}{2}(-\text{ArcTan}[1 - \text{Cos}[x], -\text{Sin}[x]] + \text{ArcTan}[1 - \text{Cos}[x], \text{Sin}[x]])$ 
```

The answer involves a two-argument version of the inverse trigonometric function  $\tan^{-1}$  but involves no explicit  $i$ . Moreover, we can similarly show that

```
In[.]:= FullSimplify[ComplexExpand[Im[Sum[Sin[n*x]/n,
      {n, 1, Infinity}]], TargetFunctions -> {Im, Re}]]
Out[.]:= 0
```

One should be aware that Dirichlet's and Abel's tests are not really suitable for present day computer programs since they require "guessing" how to decompose an infinite series into a dot product of two sequences with the required properties. For example, the current version of Mathematica<sup>®</sup> cannot determine the convergence of the series  $\sum_{n=1}^{\infty} \sin(n) \sin(1/n)$  even though it is very similar to the above one:

```
In[.]:= SumConvergence[Sin[1/n]*Sin[n], n]
Out[.]:= SumConvergence[Sin[ $\frac{1}{n}$ ] Sin[n], n]
```

We know that  $\{\sin(n)\}$  has bounded partial sums and  $\{\sin(1/n)\}$  monotonically converges to 0, so to human eyes the problem of convergence is not harder than before.

### 3.6 The function SumConvergence

This section will be devoted to the deeper study of the function SumConvergence.

```
In[.]:= Options[SumConvergence]
Out[.]:= {Method -> Automatic, Assumptions -> $Assumptions,
      Direction -> 1}
```

The option Direction gives us the direction of summation, where the indices can go to  $\infty$  or to  $-\infty$ :

```
In[.]:= SumConvergence[1/z^n, n, Direction -> 1]
Out[.]:= Abs[z] > 1

In[.]:= SumConvergence[1/z^n, n, Direction -> -1]
Out[.]:= Abs[z] < 1
```

For us the most interesting option of SumConvergence is Method. It takes four possible values: "IntegralTest", "RaabeTest", "RatioTest", "RootTest". These are not the only tests that Mathematica<sup>®</sup> uses but they are the only ones that the user can choose. When we tell Mathematica<sup>®</sup> to use one of these tests, it will return the answer True or False depending on the result of the test. Let us first look at some examples:

```
In[.]:= tests = {"IntegralTest", "RaabeTest", "RatioTest",
  "RootTest"};
In[.]:= (SumConvergence[1/n^n, n, Method -> #1] & ) /@ tests
Out[.]:= {SumConvergence[n^-n, n, Method -> "IntegralTest"],
  True, True, True}
```

We see that in this case all the tests except for the integral test succeeded.

Note also that the function SumConvergence can also be used with series involving a parameter and that one can use the assumptions mechanism. For example,

```
In[.]:= SumConvergence[n^a, n, Assumptions -> Element[a,
  Reals]]
Out[.]:= 1+a < 0

In[.]:= SumConvergence[(n^(1/n) - 1)^a, n,
  Assumptions -> Element[a, Reals]]
Out[.]:= a > 1
```

Without using Assumptions Mathematica<sup>®</sup> cannot give an answer:

```
In[.]:= SumConvergence[(n^(1/n) - 1)^a, n]
Out[.]:= SumConvergence[(-1 + n^(1/n))^a, n]
```

We can check that none of these tests in our list will work for the alternating harmonic series:

```
In[.]:= (SumConvergence[(-1)^(n + 1)*(1/n),
  n, Method -> #1] & ) /@ tests
Out[.]:= {SumConvergence[(-1)^(1+n)/n, n, Method -> IntegralTest],
  SumConvergence[(-1)^(1+n)/n, n, Method -> RaabeTest],
  SumConvergence[(-1)^(1+n)/n, n, Method -> RatioTest],
  SumConvergence[(-1)^(1+n)/n, n, Method -> RootTest]}
```

Nevertheless Mathematica<sup>®</sup> can show convergence:

```
In[.]:= SumConvergence[(-1)^(n + 1)*(1/n), n]
Out[.]:= True
```

### 3.6.1 Example

Let us consider convergence of the series of the form  $\sum_{n=1}^{\infty} n^{-p}$ :

```
In[.]:= SumConvergence[n^(-p), n]
Out[.]:= Re[p] > 1
```

Mathematica<sup>®</sup>'s answer is actually valid for complex  $p$ . If we wanted to consider only real values we could have used the following expression:

```
In[.]:= SumConvergence[n^(-p), n, Assumptions ->
      Element[p, Reals]]
```

```
Out[.]:= p > 1
```

Let us check which of the tests was successful:

```
In[.]:= (SumConvergence[n^(-p), n, Method -> #1,
      Assumptions -> Element[p, Reals]] & ) /@ tests
```

```
Out[.]:= {p > 1, p > 1, SumConvergence[n^-p, n, Method ->
      RatioTest, Assumptions -> p ∈ ℝ],
      SumConvergence[n^-p, n, Method ->
      RootTest, Assumptions -> p ∈ ℝ]}
```

We see that both the integral test and Raabe's test work. We can verify this with Mathematica<sup>®</sup>:

```
In[.]:= Integrate[x^(-p), {x, 1, Infinity},
      Assumptions -> Element[p, Reals]]
```

```
Out[.]:= ConditionalExpression[ $\frac{1}{-1+p}$ , p > 1]
```

```
In[.]:= Limit[n*((n + 1)^p/n^p - 1), n -> Infinity]
```

```
Out[.]:= p
```

Both Raabe's test and the integral test show that the series is convergent when  $p > 1$ . Raabe's test also shows that the series is divergent for  $p < 1$ .

The series  $\sum_{n=1}^{\infty} 1/n$  is known as the harmonic series. It is divergent and provides an example of a series which satisfies the condition that its  $n$ -th term tends to 0 yet is not convergent. The series  $\sum_{n=1}^{\infty} 1/n^p$  for various  $p$  provide a very useful family which very often turn out to be suitable for using in the comparison or the limit comparison tests.

Let us now consider another problem that Mathematica<sup>®</sup> is unable to solve: determine if the series

$$\sum_{n=1}^{\infty} \left( \frac{1}{n^{9/8}} + \frac{(-1)^n}{n} \right) \quad (3.7)$$

is convergent.

```
In[.]:= SumConvergence[1/n^(9/8) + (-1)^n/n, n]
```

```
Out[.]:= SumConvergence[ $\frac{1}{n^{9/8}} + \frac{(-1)^n}{n}$ , n]
```

Of course, it is very easy to see that this series is convergent. We simply use the well-known and easy to prove fact (which follows from the earlier stated property of limits



of sequences) that the sum of the sum of two series is the sum of their sums, provided everything is defined.

```
In[ ]:= SumConvergence[1/n^(9/8), n]
Out[ ]:= True
```

```
In[ ]:= SumConvergence[(-1)^n/n, n]
Out[ ]:= True
```

Hence, the series (3.7) is convergent.

### 3.7 Riemann's theorem on conditionally convergent series

We now know that the alternate harmonic series  $\sum_{n=1}^{\infty} (-1)^{n+1}/n$  is convergent although it is not absolutely convergent. Such a series is called conditionally convergent. Conditionally convergent series are tricky as their sums do not behave like ordinary sums. For example, these sums (that is, the limits of the sequences of partial sums) can change when the terms are rearranged. In fact, *Riemann's theorem* on conditionally convergent series [14, Section 5.3.3] asserts that for any  $a \in \mathbb{R} \cup \{\infty, -\infty\}$  one can find a permutation of terms such that the sum of the series will be  $a$  (that includes the cases  $a = \infty$  and  $a = -\infty$ ). The proof is an algorithm which is easy to implement in Mathematica<sup>®</sup>. This is done in [17], where some interesting related questions are considered. Here we will give a different implementation which is more suitable for a graphic representation of the theorem.

Consider for instance the alternate harmonic series  $\sum_{n=1}^{\infty} (-1)^{n+1}/n$  and assume that  $a \in \mathbb{R}$ . The algorithm is as follows. Let  $A$  be the set of all positive terms of this series and let  $B$  be the set of all negative terms of this series, all arranged in the order of the appearance in the original series. We start by adding elements of  $A$  in turn until we reach a number  $a_1$  which is greater than  $a$ . Because the sum of the series consisting of elements of  $A$  is infinity, this must always occur after a finite number of additions. Then we start adding elements of  $B$  until our sum becomes  $a_2 < a$ . Then we again add a finite number of the remaining elements of  $A$  until the sum becomes  $a_3 > a$  and we continue in this way. One can easily see that  $|a - a_n| \rightarrow 0$ , hence the sequence of partial sums  $a_n$  of the rearranged alternate harmonic series tends to  $a$ . The reader should supply an algorithm for  $a = \infty (-\infty)$ .

The algorithm written above constructs partial sums that are arbitrary close to the given number  $a$  but it never stops. Of course, when we want to implement the algorithm on a computer, we have to decide when to stop. There are several natural places to do that, for example, when we obtain a sufficiently close approximation or when we have used a given number of terms of the series. Our implementation uses a slightly different approach. We provide a list of positive terms and negative terms and the program stops when it can no longer continue (when there are no more needed elements in one of the lists). The program returns a fragment of the rearranged sequence of

terms of the alternate harmonic series (so that the partial sums of the series are obtained by applying the function `Accumulate` to the list) and approximate sum. Note that our algorithm works for an arbitrary conditionally convergent series.

```
In[ ]:= Rearrangement[a_, pos_, neg_] :=
Module[{p = pos, n = neg, s = {}, sum = 0},
Catch[While[p != {} || n != {},
While[sum <= a, If[p == {}, Throw[{s, N[sum]}],
AppendTo[s, First[p]]; sum = sum + First[p];
p = Rest[p]]; While[sum >= a, If[n == {},
Throw[{s, N[sum]}], AppendTo[s, First[n]];
sum = sum + First[n]; n = Rest[n]]]; {s, N[sum]}]]
```

For the alternate harmonic series we have:

```
In[ ]:= listpos[n_] := Table[(-1)^(k + 1)/k, {k, 1, n, 2}];
In[ ]:= listneg[n_] := Table[(-1)^(k + 1)/k, {k, 2, n, 2}];
In[ ]:= Rearrangement[2, listpos[30], listneg[20]]
Out[ ]:= {{1,  $\frac{1}{3}$ ,  $\frac{1}{5}$ ,  $\frac{1}{7}$ ,  $\frac{1}{9}$ ,  $\frac{1}{11}$ ,  $\frac{1}{13}$ ,  $\frac{1}{15}$ ,  $-\frac{1}{2}$ ,  $\frac{1}{17}$ ,  $\frac{1}{19}$ ,  $\frac{1}{21}$ ,  $\frac{1}{23}$ ,  $\frac{1}{25}$ ,  $\frac{1}{27}$ ,  $\frac{1}{29}$ },
1.83587}
```

We can now see how the partial sums of the rearranged series converge to 2.

```
In[ ]:= ListLinePlot[Accumulate[First[Rearrangement[2,
listpos[2500], listneg[2500]]]]]
```

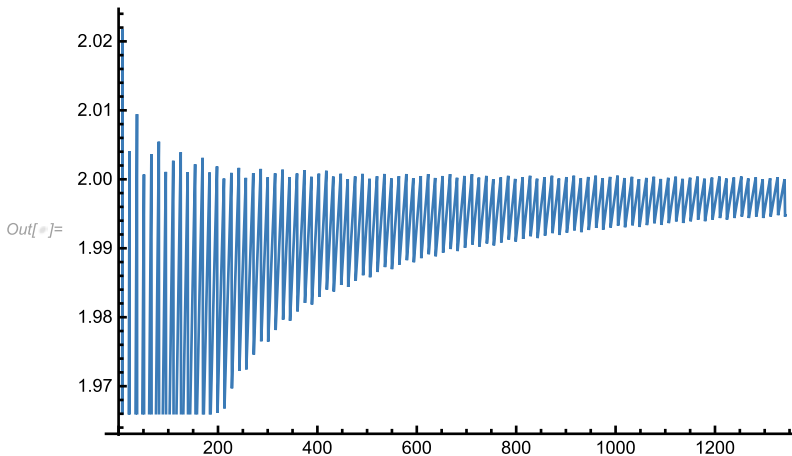


Figure 3.2

### 3.8 The Cauchy product of series

Suppose we have two infinite series  $\sum_{i=0}^{\infty} a_i$  and  $\sum_{i=0}^{\infty} b_i$ . We know that we can multiply a series by a number (just multiply each term) and we can formally add them:

$$\sum_{i=0}^{\infty} a_i + \sum_{i=0}^{\infty} b_i = \sum_{i=0}^{\infty} (a_i + b_i).$$

We know that this equation is also valid for the sums of the series, whenever both sides are defined.

We would naturally like to define the product of two series in such a way that the analogous property holds, that is, the sum of the product series  $\sum_{i=0}^{\infty} a_i \times \sum_{i=0}^{\infty} b_i$  should be the product of the sums of  $\sum_{i=0}^{\infty} a_i$  and  $\sum_{i=0}^{\infty} b_i$ , when again everything is defined. It is obvious that the product must be a sum of the terms  $a_i b_j$  for all  $i$  and  $j$  but the order in which we add them is important (unless we have an absolutely convergent series or a finite sum) for the product series to have the desired property. When we multiply and expand finite sums, Mathematica® arranges the terms in its own standard order but if we used this order for multiplying infinite series, the product series would not have good properties. Therefore, we define the so called Cauchy product of series by “going along the diagonals” in the infinite matrix as in the picture below.

```
In[ ]:= g1 = Graphics[Flatten[Table[If[i - j <= 4,
  Point[{i/3, j/3}], Point[{}]], {i, 0, 4},
  {j, 0, -4, -1}], 1]];
```

```
In[ ]:= g2 = Graphics[Flatten[Table[If[i - j <= 3,
  Text[Subscript[a, i]* Subscript[b, -j],
  {i/3, j/3}], Text["", {i/3, j/3}]], {i, 0, 3},
  {j, 0, -3, -1}], 1]];
```

```
In[ ]:= g3 = Graphics[Table[Line[Table[{(n - i)/3, -i/3},
  {i, 0, n}], {n, 0, 3}]]];
```

```
In[ ]:= Show[g1, g2, g3]
```

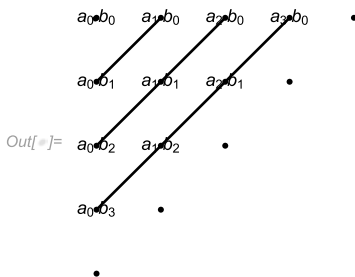


Figure 3.3

Formally, we define the *Cauchy product* by

$$\begin{aligned} \sum_{i=0}^{\infty} a_i \times \sum_{i=0}^{\infty} b_i &= \sum_{n=0}^{\infty} \left( \sum_{i=0}^n a_i b_{n-i} \right) \\ &= a_0 b_0 + (a_0 b_1 + a_1 b_0) + (a_0 b_2 + a_1 b_1 + a_2 b_0) + \dots \end{aligned}$$

The theorem due to Mertens says that if both series  $\sum_{i=0}^{\infty} a_i$  and  $\sum_{i=0}^{\infty} b_i$  are convergent and if one of them is absolutely convergent, then the sum of their Cauchy product is the product of their sums.

### 3.9 Divergent series

We know that a series can be divergent in different ways, it can be divergent to  $\infty$  or  $-\infty$  or have no limit. The fact that a series is divergent to infinity can sometimes be useful. For example:

**In[.]**:= SumConvergence[1/Prime[n], n]

**Out[.]**:= False

This tells us that the sum of the series  $\sum_{n=1}^{\infty} 1/p_n$  is  $\infty$ , which implies that there are infinitely many primes. On the other hand,

**In[.]**:= SumConvergence[(-1)^n, n]

**Out[.]**:= False

This alone does not tell us anything useful about this series.

We will now see that in many cases we can find a different notion of “sum” of series, which gives a finite answer in the case of some divergent series. This is referred to as “regularized sum” and can be calculated using the option *Regularization* in Mathematica®’s function *Sum*. For example, we have

**In[.]**:= Sum[(-1)^n, {n, 0, Infinity}, Regularization -> "Cesaro"]

**Out[.]**:=  $\frac{1}{2}$

This answer looks strange but surprisingly regularization is useful and not only in mathematics (where it is used to prove formulas about sums of convergent series [10, Section 1.2], in solving differential equations [2], etc.) but also in modern physics.

We begin with the notion of regularized sum of a series [10]. In this section we shall use the notation  $\sum_{n=1}^{\infty} a_n$  for a series (sequence of partial sums of the sequence  $\{a_n\}$ ), and we will write  $S(\sum_{n=1}^{\infty} a_n)$  for its sum (limit) if it exists. Then a *regularized sum* is a function  $S^* : \mathcal{S} \rightarrow \mathbb{R}$  (more generally  $\mathbb{C}$ ), where  $\mathcal{S}$  is a set of all series, with the following properties:

1.  $S^*(\sum_{n=1}^{\infty} a_n) = S(\sum_{n=1}^{\infty} a_n)$  for every convergent series  $\sum_{n=1}^{\infty} a_n$ ;

2.  $S^*(a_0 + \sum_{n=1}^{\infty} a_n) = a_0 + S^*(\sum_{n=1}^{\infty} a_n)$  (invariance with respect to addition);
3.  $S^*(\sum_{i=1}^{\infty} (a_i + b_i)) = S^*(\sum_{i=1}^{\infty} a_i) + S^*(\sum_{i=1}^{\infty} b_i)$ ;
4.  $S^*(\sum_{i=1}^{\infty} \lambda a_i) = \lambda S^*(\sum_{i=1}^{\infty} a_i)$ ;
5.  $S^*(\sum_{i=1}^{\infty} a_i \times \sum_{i=1}^{\infty} b_i) = S^*(\sum_{i=1}^{\infty} a_i) S^*(\sum_{i=1}^{\infty} b_i)$ , where  $\times$  is the Cauchy product.

Consider the divergent series  $\sum_{n=0}^{\infty} (-1)^n$ . Let us assume that there is some  $S^*$  which satisfies properties 1–5 above and such that  $S^*(\sum_{n=0}^{\infty} (-1)^n) = S_0 \in \mathbb{R}$ . Then it is easy to find  $S_0$ . Indeed,

$$S_0 = S^*\left(\sum_{n=0}^{\infty} (-1)^n\right) = S^*\left(1 + \sum_{n=1}^{\infty} (-1)^n\right) = 1 + S^*\left(\sum_{n=1}^{\infty} (-1)^n\right) = 1 - S_0.$$

Hence,  $S_0 = 1/2$ . Of course, this does mean that such regularization exists and clearly it should be given explicitly.

To define a regularization one needs to find a function  $S^*$  with the above properties. For example, the Cesàro regularization is based on the following theorem (see, for instance, [14, Theorem 2.64]).

**Theorem 8.** *Let  $a_n$  be a sequence with limit  $g \in \mathbb{R} \cup \{\infty, -\infty\}$ . Then the sequence of arithmetic means  $\{(\sum_{i=1}^n a_i)/n\}$  has limit  $g$ .*

In Mathematica<sup>®</sup> we can create means from lists by using for instance

```
In[.] := (#1/Length[#1] & ) /@ Rest[Accumulate[{a, b, c, d, e}]]
Out[.] := { $\frac{a+b}{2}$ ,  $\frac{1}{3}(a+b+c)$ ,  $\frac{1}{4}(a+b+c+d)$ ,  $\frac{1}{5}(a+b+c+d+e)$ }
```

Note that the sequence of means of a sequence may be convergent when the sequence itself has no limit. The standard example is  $\{(-1)^n\}$ , which does not have a limit, while

```
In[.] := Discretelimit[Sum[(-1)^k, {k, 1, n}]/n, n -> Infinity]
Out[.] := 0
```

Theorem 8 is the basis for the *Cesàro regularization*, in which we define the Cesàro sum of a series as the limit of the sequence of means of the sequence of partial sums of the sequence. That is,

$$S^*\left(\sum_{i=1}^{\infty} a_i\right) = \lim_{m \rightarrow \infty} \frac{\sum_{n=0}^m \sum_{k=0}^n a_k}{m+1}. \quad (3.8)$$

It is easy to check that  $S^*$  defined in (3.8) satisfies conditions 1–5 above.

An important application of Cesàro's regularization is Cesàro's theorem on the Cauchy product of two convergent series. The theorem states that the Cauchy product is Cesàro summable and its Cesàro sum is the product of the sums of two series.

There are other implemented regularization methods in Mathematica<sup>®</sup> (the Abel method, which uses power series discussed in Section 3.10, the Borel method, which

uses integration discussed later on as well, the Dirichlet method, which uses a function series, and the Euler methods for alternating sums).

We will not consider any applications of divergent series here except for remarking that with their help one can prove various identities involving convergent series (see examples in [10]).

For example, let  $t$  be a number such that  $-\pi \leq t \leq \pi$ . Then the following identity holds:

$$\sum_{n=1}^{\infty} \frac{(-1)^{n-1}(1 - \cos(tn))}{n^2} = \frac{t^2}{4}. \quad (3.9)$$

The proof given in [10] involves expanding the series on the left (which we can prove is convergent) in terms of  $t$  and then rearranging and expressing it in terms of divergent series which can be summed using a regularization method. Because the original series is convergent, the answer that one gets using this method must be equal to the ordinary sum of the series.

Mathematica<sup>®</sup> gives a much more complicated answer for the left hand side, namely,

```
In[.]:= FullSimplify[Sum[((-1)^(n - 1)*(1 - Cos[t*n]))/n^2,
    {n, 1, Infinity}], Assumptions -> -Pi <= t <= Pi]
Out[.]:=  $\frac{1}{12}(\pi^2 + 6 \text{PolyLog}[2, -E^{-t}] + 6 \text{PolyLog}[2, -E^t])$ 
```

and Mathematica<sup>®</sup> is unable to prove the above identity. In situations, when we have an identity we believe to be true but which Mathematica<sup>®</sup> cannot prove, it is still possible to provide some evidence for the truth (or possibly disproving it) by substituting random numbers lying within some range. However, using `MachinePrecision` random numbers may sometimes not be sufficient to confirm such an identity, because the inaccuracy of calculations may cause the program to give different answers on both sides. This is a situation in which Mathematica<sup>®</sup>'s ability to control precision can be useful.

```
In[.]:= Sum[((-1)^(n - 1)*(1 - Cos[t*n]))/n^2,
    {n, 1, Infinity}] == t^2/4 /. Transpose[{Thread[t ->
    RandomReal[{-Pi, Pi}, {7}], WorkingPrecision -> 10]]]
Out[.]:= {True, True, True, True, True, True, True}
```

Using only `MachinePrecision` we sometimes can get `False`:

```
In[.]:= Sum[((-1)^(n - 1)*(1 - Cos[t*n]))/n^2,
    {n, 1, Infinity}] == t^2/4 /.
    Transpose[{Thread[t -> RandomReal[{-Pi, Pi}, {7}]]}]
Out[.]:= {False, True, True, True, True, True, True}
```

### 3.10 Power series

A *power series* is a series of the form  $\sum_{n=0}^{\infty} a_n(x - x_0)^n$ , where  $\{a_n\}$  is a sequence of real (or complex) numbers called the coefficients of the power series and  $x$  is in  $\mathbb{R}$  or  $\mathbb{C}$ . The number  $x_0$  is called the *center of the power series*. The set of points  $\{x : \sum_{n=0}^{\infty} a_n(x - x_0)^n \text{ converges}\}$  is called the *region of convergence of the series*. It is always non-empty because it always contains the point  $x_0$ . One can show that the region of convergence is always an interval which could be just the point  $x_0$ , a finite interval (closed, open or half open) or the entire real line (in the complex case the region of convergence is a disk). There are formulas to find the radius of convergence of power series [14] which can be derived from convergence tests. Note that the radius of convergence does not determine the region of convergence and the endpoints of the interval have to be checked separately.

The function `SumConvergence` can also be used to find the region of convergence of power series. Here are a few examples.

**In[.]**:= `SumConvergence[n!*x^n, n]`

**Out[.]**:= `x == 0`

The region of convergence of  $\sum_{n=1}^{\infty} n!x^n$  is just the center of the series. This is easily proved by using either the root or the ratio test.

**In[.]**:= `SumConvergence[(x - 1)^n/n, n]`

**Out[.]**:= `Abs[-1 + x] <= 1 && x != 2`

In the real case this says that the series  $\sum_{n=1}^{\infty} (x-1)^n/n$  converges on the half open interval  $[-2, 2)$ . Convergence inside the open interval  $(-2, 2)$  can again be easily proved by using the root or ratio test, but the endpoints  $-2$  and  $2$  must be considered separately. For  $x = 2$  we get  $\sum_{n=1}^{\infty} (-1)^n/n$  which we know is convergent (by the Dirichlet test) and for  $x = -2$  we get the harmonic series, which we know is divergent. In the complex case the result says that the series is convergent on the entire closed disk  $|x - 1| \leq 1$  except at  $x = 2$ .

**In[.]**:= `SumConvergence[x^n/n!, n]`

**Out[.]**:= `True`

This says that the series  $\sum_{n=1}^{\infty} x^n/n!$  converges on the entire real line (complex plane).

Let  $\sum_{n=0}^{\infty} a_n(x - x_0)^n$  be a series with a region of convergence  $I \subset \mathbb{R}$ . Then  $f(x) := \sum_{n=0}^{\infty} a_n(x - x_0)^n$  defines a function  $f : I \rightarrow \mathbb{R}$ . Functions defined in this way are called *analytic functions* and we will see that they possess many properties of polynomials. Here are some important analytic functions:

$$\exp(x) = \sum_{n=0}^{\infty} \frac{x^n}{n!},$$

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1},$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n}.$$

All of these series have the region of convergence  $\mathbb{R}$  (or  $\mathbb{C}$  in the complex plane), hence the corresponding functions are defined everywhere. Such analytic functions are called *entire*. These functions are equal to the familiar functions usually defined in a different way in elementary calculus courses. One can show that the functions  $\cos$  and  $\sin$  are indeed equal to the usual trigonometric functions defined in terms of ratios of sides of right angled triangles and  $\exp(x) = e^x$ , where  $e$  was defined earlier as a limit of a certain sequence.

We can use the properties of the Cauchy product of series to prove that these functions defined by power series have the expected properties. For example, let us show the identity

$$\exp(x + y) = \exp(x) \exp(y).$$

We will prove this by showing that the same identity holds when  $\exp$  denotes the series itself, rather than its sum and the product on the right denotes the Cauchy product of the series. Then from Mertens' theorem it will follow that our identity (in which  $\exp$  means the sum of the series) is true.

The expression on the left is the sum of a power series in  $z = x + y$ , which after expansion of each term  $(x + y)^n$  becomes a power series in two variables  $x$  and  $y$ . We can find the coefficient of  $x^n y^m$  by using Mathematica<sup>®</sup>'s function `SeriesCoefficient` (which will be considered in greater detail later, when we study the Taylor expansion of functions):

$$\begin{aligned} \mathbf{In[.]} &:= \text{SeriesCoefficient}[\text{Exp}[x + y], \{x, 0, n\}, \{y, 0, m\}] \\ \mathbf{Out[.]} &:= \begin{cases} \frac{1}{m! n!} & n \geq 0 \wedge m \geq 0 \\ 0 & \text{True} \end{cases} \end{aligned}$$

Note that this kind of expression in Mathematica<sup>®</sup> the word `True` should be interpreted as saying “otherwise”.

Now we do the same to the power series on the right hand side and we see that the coefficients are the same. This means that this series is the same as the series on the left hand side:

$$\begin{aligned} \mathbf{In[.]} &:= \text{SeriesCoefficient}[\text{Exp}[x] * \text{Exp}[y], \{x, 0, n\}, \{y, 0, m\}] \\ \mathbf{Out[.]} &:= \begin{cases} \frac{1}{m! n!} & n \geq 0 \wedge m \geq 0 \\ 0 & \text{True} \end{cases} \end{aligned}$$

In some cases Mathematica<sup>®</sup> may not give the same result unless we use `FullSimplify` possibly with certain assumptions. Consider for instance the following trigonometric identity:



```
In[.]:= TrigExpand[Cos[x + y]]
Out[.]:= Cos[x] Cos[y] - Sin[x] Sin[y]
```

If we use the same method as above, Mathematica® will not return the same expressions on the right and the left hand sides. However, we can show that they are equivalent by using FullSimplify with suitable assumptions:

```
In[.]:= FullSimplify[SeriesCoefficient[Cos[x + y], {x, 0, n},
  {y, 0, m}], Assumptions -> Element[{m, n}, Integers]
  && m >= 0 && n >= 0]
Out[.]:= 
$$\frac{\cos\left[\frac{1}{2}(m+n)\pi\right]}{m!n!}$$

```

```
In[.]:= FullSimplify[SeriesCoefficient[Cos[x]*Cos[y] -
  Sin[x]*Sin[y], {x, 0, n}, {y, 0, m}], Assumptions ->
  Element[{m, n}, Integers] && m >= 0 && n >= 0]
Out[.]:= 
$$\frac{\cos\left[\frac{1}{2}(m+n)\pi\right]}{m!n!}$$

```

In general, we cannot expect that applying FullSimplify to two different but equal expressions will produce the same expression. For this reason it is usually better to apply FullSimplify to the difference of the expressions and check whether Mathematica® returns 0:

```
In[.]:= FullSimplify[SeriesCoefficient[Cos[x + y], {x, 0, n},
  {y, 0, m}] - SeriesCoefficient[Cos[x]*Cos[y] -
  Sin[x]*Sin[y], {x, 0, n}, {y, 0, m}], Assumptions ->
  Element[{m, n}, Integers] && m >= 0 && n >= 0]
Out[.]:= 0
```



## 4 Limits of functions and continuity

In this chapter we will extend the concept of limit to arbitrary real-valued functions of one real variable, define continuity and study some properties of continuous functions and related topics. We state without proof several important theorems on continuous functions and give some examples of their applications.

### 4.1 Limits of functions

Let  $A \subset \mathbb{R}$ . We say that a point  $x \in \mathbb{R} \cup \{\infty, -\infty\}$  is a *limit point* of  $A$  if there exists a sequence  $\{a_n\}$  with  $a_n \in A$ ,  $a_n \neq x$  for all  $n$  such that  $\lim_{n \rightarrow \infty} a_n = x$ . Note that  $x$  itself need not be in  $A$ . In fact  $\infty$  is the only limit point of  $\mathbb{N}$ . Another example is an open interval  $A = (a, b)$ , in which case  $a$  and  $b$  are both limit points although they do not belong to  $A$ . Of course, a point of  $A$  can be (but need not be) a limit point. The most common type of limit point that belongs to  $A$  is an interior point, that is, a point that is contained in an open interval which is a subset of  $A$ .

Let  $x_0$  be a limit point of  $A$  and let  $f : A \rightarrow \mathbb{R}$  be a function. We say that  $g \in \mathbb{R} \cup \{\infty, -\infty\}$  is the *limit of function*  $f$  as  $x \rightarrow x_0$  (written as  $\lim_{x \rightarrow x_0} f(x) = g$ ) if for every sequence  $x_n \rightarrow x_0$  we have

```
In[.]:= DiscreteLimit[f[x[n]], n -> Infinity] == g
```

This definition illustrates clearly the distinction between Mathematica<sup>®</sup>'s functions `DiscreteLimit` and `Limit`. `Limit` is defined for functions in terms of `DiscreteLimit`, which is defined for sequences. On the other hand, since a sequence  $a_n$  is a function  $a : \mathbb{N} \rightarrow \mathbb{R}$ , where  $a(n) = a_n$ , the limit of the sequence  $a_n$  (`DiscreteLimit`) is defined as the limit (`Limit`)  $\lim_{n \rightarrow \infty} a(n)$ .

This definition of limit is known as *Heine's definition*. It is equivalent to another definition known as *Cauchy's definition*. Unlike Heine's definition, Cauchy's definition takes a different form in the cases when both  $x_0$  and  $g$  are real numbers and in the cases when any of them is either  $\infty$  or  $-\infty$ . This means that there are nine different definitions, though as they are very similar we will only state two of them.

**Definition 1.** Let  $a \in \mathbb{R}$  be a limit point of  $A \subset \mathbb{R}$ , let  $f : A \rightarrow \mathbb{R}$  be a function and let  $g \in \mathbb{R}$ . Then  $\lim_{x \rightarrow a} f(x) = g$  if and only if for every  $\varepsilon > 0$  there is a  $\delta > 0$  such that  $0 < |x - a| < \delta$  implies  $|f(x) - g| < \varepsilon$ .

Note that  $\delta$  depends on both  $a$  and  $\varepsilon$ .

**Definition 2.** Suppose that  $\infty$  is a limit point of  $A$  (e. g.,  $A = \mathbb{R}$ ). Then  $\lim_{x \rightarrow \infty} f(x) = \infty$  if for every  $M > 0$  there exists  $N > 0$  such that  $x > N$  implies  $f(x) > M$ .

The function `Limit` returns conditional answers and admits assumptions:

```
In[.]:= Limit[x^a, x -> Infinity]
```

```
Out[.]:= ConditionalExpression[∞, a > 0]
```

<https://doi.org/10.1515/9783110590142-004>

**In[.]**:= Limit[x^a, x -> Infinity, Assumptions -> {a < 0}]

**Out[.]**:= 0

If for some reason we wish to get an unconditional answer we can use the option `GenerateConditions`:

**In[.]**:= Limit[x^a, x -> Infinity, GenerateConditions -> False]

**Out[.]**:=  $\infty$

We should do this only in case we are sure that  $a > 0$ .

Limits of functions at a point have properties analogous to those of limits of sequences and their proofs reduce to the proofs of analogous results for sequences. For example, it is easy to extend statements about the limits of sums, products and reciprocals of sequences to sums, products and reciprocals of functions. We will thus omit these statements (see [14, Chapter 3]).

Here are some examples of computation of certain important limits which can be proved by using the Taylor series:

**In[.]**:= Limit[(Exp[x] - 1)/x, x -> 0]

**Out[.]**:= 1

**In[.]**:= Limit[Sin[x]/x, x -> 0]

**Out[.]**:= 1

**In[.]**:= Limit[(Cos[x] - 1)/x^2, x -> 0]

**Out[.]**:=  $-\frac{1}{2}$

## 4.2 One-sided limits

Let  $f : A \rightarrow \mathbb{R}$  be a function and  $a \in \mathbb{R}$  be a limit point. We say that  $g$  is a *limit of  $f$  at  $a$  from the right (or from above)* if for each sequence  $x_n \rightarrow a$  such that  $x_n > a$  for all  $n \in \mathbb{N}$  we have  $\lim_{n \rightarrow \infty} f(x_n) = g$ . In this case we write  $\lim_{x \rightarrow a^+} f(x) = g$ . We say that  $g$  is a *limit of  $f$  at  $a$  from the left (or from below)* if for each sequence  $x_n \rightarrow a$  such that  $x_n < a$  for all  $n \in \mathbb{N}$  we have  $\lim_{n \rightarrow \infty} f(x_n) = g$ . We write  $\lim_{x \rightarrow a^-} f(x) = g$ . We have the following theorem.

**Theorem 9.** *The limit  $\lim_{x \rightarrow a} f(x) = g$  if and only if  $\lim_{x \rightarrow a^-} f(x)$  and  $\lim_{x \rightarrow a^+} f(x)$  both exist and are equal to  $g$ .*

Of course in the case when  $a = \infty$  all limits are actually limits from the left and when  $a = -\infty$  all limits are limits from the right.

As we already know, the function `Limit` computes limits of functions in `Mathematica`<sup>®</sup>. It provides one of the cases in which a significant change took place in `Mathematica`<sup>®</sup> in version 11, which does not preserve “backward compatibility” (i. e., certain outputs will be different when evaluated in earlier versions of `Mathematica`<sup>®</sup>). The main difference is that before version 11 `Mathematica`<sup>®</sup>'s `Limit` by default always computed the limit from the right:

```
In[.]:= Limit[1/x, x -> 0]
Out[.]:= ∞
```

To get the limit from the left one had to use the options `Direction`:

```
In[.]:= Limit[1/x, x -> 0, Direction -> 1]
Out[.]:= -∞
```

The two limits are not equal, hence the two-sided limit does not exist. This is indeed the answer that `Limit` returns in `Mathematica`® 11:

```
In[.]:= Limit[1/x, x -> 0]
Out[.]:= Indeterminate
```

This, as we already know, means that the limit does not exist.

To compute one-sided limits in `Mathematica`® 11, we have to use the option `Direction`, which still can take the values 1 and -1 as in older versions or “FromAbove” and “FromBelow”:

```
In[.]:= Limit[1/x, x -> 0, Direction -> "FromAbove"]
Out[.]:= ∞
```

One can also compute the limit in the complex plane, for example,

```
In[.]:= Limit[1/x, x -> 0, Direction -> "Complexes"]
Out[.]:= ComplexInfinity
```

Here we should mention another difference in the behavior of the function `Limit` in `Mathematica`® 11 and in the earlier versions. Consider the limit of the function  $\sin(1/x)$  as  $x$  tends to 0. Clearly this limit does not exist:

```
In[.]:= Plot[Sin[1/x], {x, -1, 1}, Exclusions -> {x == 0}]
```

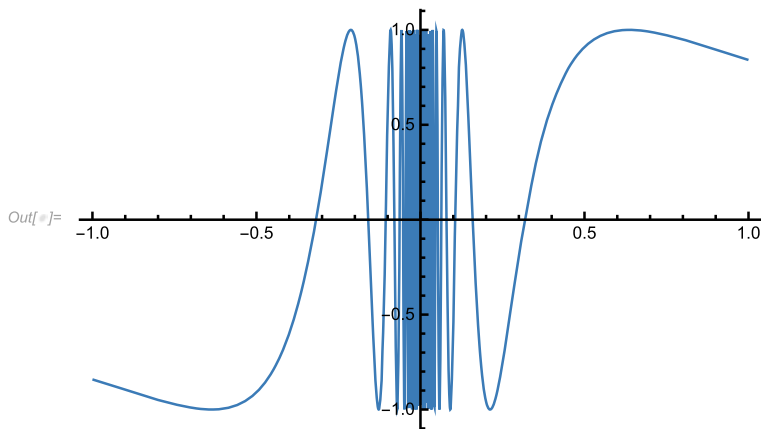


Figure 4.1

In `Mathematica`® 10 we obtain

```
In[.]:= Limit[Sin[1/x], x -> 0, Direction -> -1]
Out[.]:= Interval[{-1, 1}]
```

This is a non-standard answer which reflects the fact that as  $x$  approaches 0 (even from above) the values of the function assume all values in the interval  $[-1, 1]$ . In Mathematica<sup>®</sup> 11, however, we obtain

```
In[ ]:= Limit[Sin[1/x], x -> 0, Direction -> "FromAbove"]
```

```
Out[ ]:= Indeterminate
```

which agrees with the usual convention that this kind of limit does not exist.

### 4.3 Continuous functions

**Definition 3.** Let  $a \in A \subset \mathbb{R}$  and let  $f : A \rightarrow \mathbb{R}$  be a function. We say that  $f$  is *continuous at  $a$*  if for every sequence  $a_n \rightarrow a$  we have  $\lim_{n \rightarrow \infty} f(a_n) = f(\lim_{n \rightarrow \infty} a_n) = f(a)$ . If  $f$  is continuous at all points in its domain we say that it is *continuous*. There is also a natural notion of a left and right continuity at a point  $a$ . If a function is not continuous at  $a$ , then we say that it is *discontinuous at  $a$* .

Note that  $a$  does not have to be a limit point of  $A$  since every function is continuous at an isolated point (because in such a case the only sequences convergent to the point are the constant sequences).

Note also that we do not consider continuity of a function at a point which does not lie in its domain. Thus the function  $f : \mathbb{R} \setminus \{0\} \rightarrow \mathbb{R}$  which takes  $x$  to  $1/x$  is continuous everywhere. However, it cannot be extended to a continuous function  $f : \mathbb{R} \rightarrow \mathbb{R}$ . Any such extended function will be discontinuous at 0. For instance if we extend the function  $f(x) = 1/x$  with  $f(0) = 3$ , then it is obviously discontinuous:

```
In[ ]:= Plot[1/x, {x, -2, 2}, Exclusions -> 0,
Epilog -> Point[{0, 3}]]
```

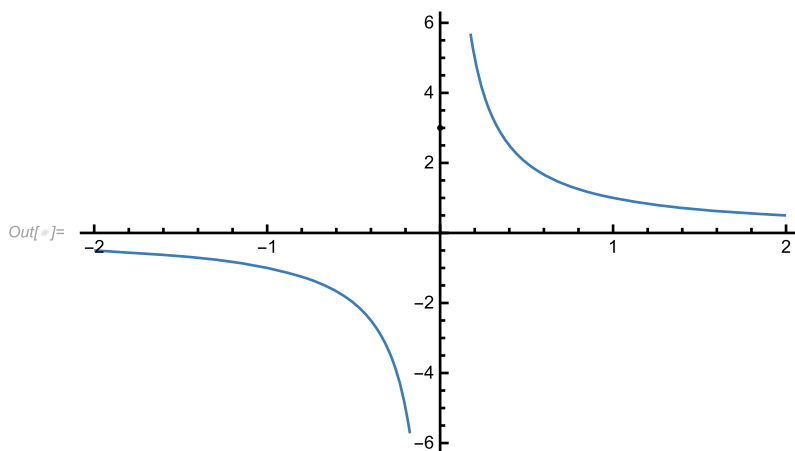


Figure 4.2

The above definition of continuity is given in Heine's form. There is also Cauchy's version, which we state as a theorem.

**Theorem 10.** *A function  $f : A \rightarrow \mathbb{R}$  is continuous at  $a \in A$  if and only if for every  $\varepsilon > 0$  there exists  $\delta > 0$  such that  $|f(x) - f(a)| < \varepsilon$  whenever  $x \in A$  and  $|x - a| < \delta$ .*

An interesting consequence of this theorem is that we can prove the continuity of certain functions by using quantifier elimination. For example, we can prove that the function  $f(x) = x^2 - x + 1$  is continuous everywhere:

```
In[ ]:= f[x_] := x^2 - x + 1
In[ ]:= Resolve[ForAll[{e, x}, e > 0, Exists[d, d > 0,
  ForAll[{y}, Element[{y}, Reals] && Abs[x - y] < d,
  Abs[f[x] - f[y]] < e]]]]
Out[ ]:= True
```

Of course, this is not a practical approach to proving continuity of functions. Instead, one starts by proving that the set of continuous functions (with a fixed domain) is closed under the operations of addition, multiplication, division (when defined) and composition of function, i. e., that the set of continuous functions has the structure of an algebra (see [14]). Since it is obvious that all constant functions are continuous and that the identity function  $x \mapsto x$  is continuous, it follows that all rational functions are continuous.

Another important set of continuous functions is the set of the *analytic functions*, i. e., functions defined by a power series. As we stated in Chapter 3, such a series is always convergent in an interval, which could be open, closed, half open and bounded or unbounded. One can show that a function defined in this way is always continuous in its entire region of convergence (if the interval has endpoints, the function is left or right continuous at these endpoints). This is Abel's Limit Theorem (see [14, p. 416]).

## 4.4 Discontinuous functions

Let us now consider some examples of *discontinuous functions*. The simplest type of discontinuity is a simple jump at a point. In one such situation both one-sided limits exist and are equal but are not equal to the value of the function at the point. Such a function can be made continuous by changing its value at a single point. The other kind of jump discontinuity is when the function is either left continuous or right continuous but not continuous as in the picture below. The best way to produce functions with this property in Mathematica<sup>®</sup> is by using the function `Piecewise`, e. g.,

```
In[ ]:= Plot[Piecewise[{{x, x <= 1}, {1 + x^2, x > 1}},
  {x, -2, 2}]
```

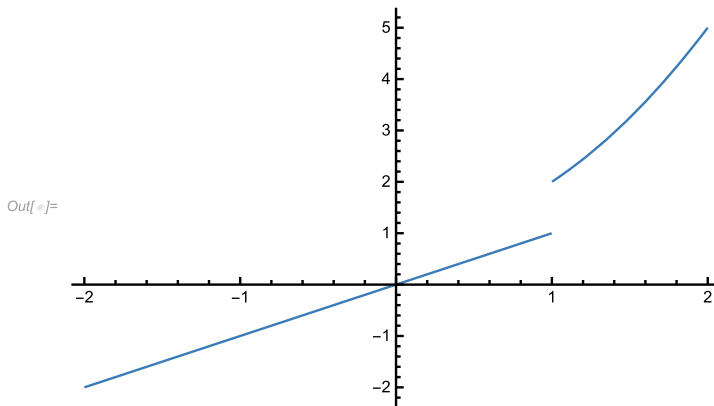


Figure 4.3

This function is obtained by joining two continuous functions whose right and left limits at 1 do not agree. Indeed, we can compute the limits

```
In[.]:= Limit[Piecewise[{{x, x <= 1}, {1 + x^2, x > 1}},
  x -> 1, Direction -> "FromAbove"]
```

```
Out[.]:= 2
```

```
In[.]:= Limit[Piecewise[{{x, x <= 1}, {1 + x^2, x > 1}},
  x -> 1, Direction -> "FromBelow"]
```

```
Out[.]:= 1
```

```
In[.]:= Limit[Piecewise[{{x, x <= 1}, {1 + x^2, x > 1}},
  x -> 1]
```

```
Out[.]:= Indeterminate
```

This is an example of a left continuous function.

We could of course use the function `If` to obtain the same plot as above by using

```
In[.]:= Plot[If[x <= 1, x, 1 + x^2], {x, -2, 2},
  Exclusions -> 1]
```

However **Mathematica**<sup>®</sup> cannot reliably perform mathematical operations such as differentiation and integration on piecewise functions defined by means of `If`. For example in the following expression

```
In[.]:= D[If[x <= 1, x, 1 + x^2], x]
```

```
Out[.]:= If[x <= 1, 1, 2 x]
```

**Mathematica**<sup>®</sup> does not notice that the function is not differentiable at 1, whereas using `Piecewise` it gives the correct answer:

```
In[.]:= D[Piecewise[{{x, x <= 1}, {1 + x^2, x > 1}}, x]
```

```
Out[.]:= {
  1          x < 1
  2 x       x > 1
  Indeterminate True
```



Another important kind of discontinuity is when the one-sided limits do not exist. For example

```
In[ ]:= f[x_] := Piecewise[{{Sin[1/x], x != 0}}, 1]
```

This situation is better seen by using an interactive graphic representation:

```
In[ ]:= Manipulate[Show[Plot[f[x], {x, 1 - d, 1},
  PlotRange -> {{-1, 1}, {-1, 1.2}}], Plot[f[x],
  {x, -1, -1 + d}, PlotRange -> {{-1, 1}, {-1, 1.2}}],
  Epilog -> {PointSize[0.02], Point[{{0, 1}}]},
  {d, 0.1, "d"}, 0.01, 0.99, 0.01]
```

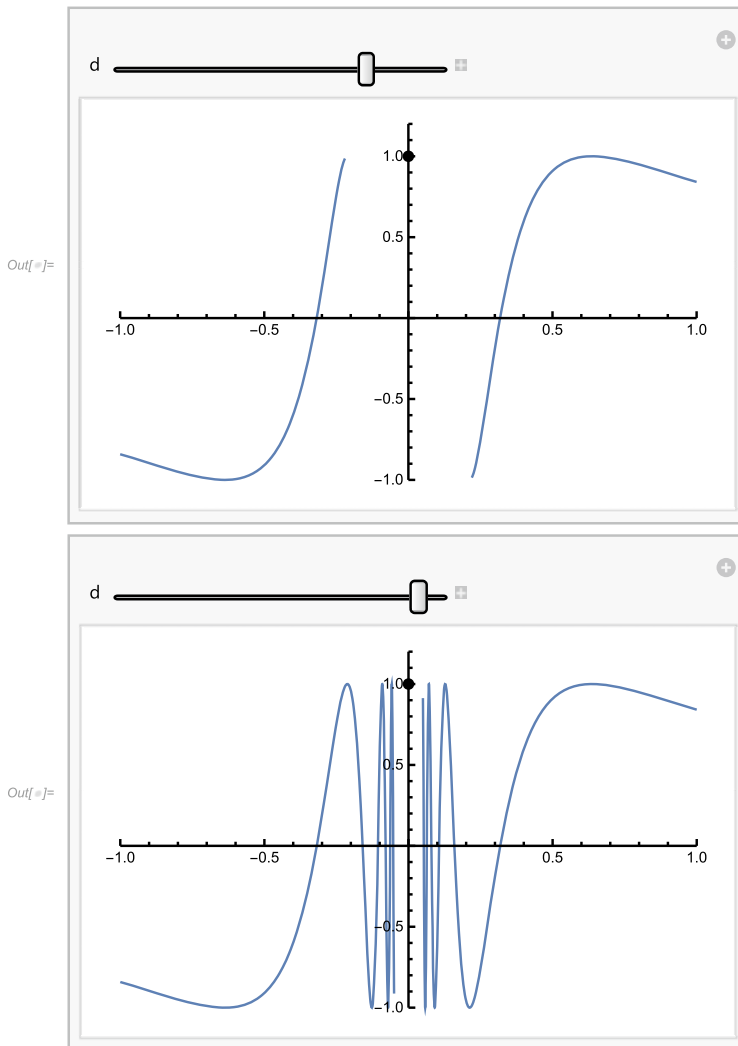


Figure 4.4

In the next example the limit from the left exists but the limit from the right does not. This is another example of the left continuous function.

```
In[.]:= Plot[Piecewise[{{x, x <= 1}, {Sin[1/(x - 1)],
x > 1}}, {x, -2, 2}]
```

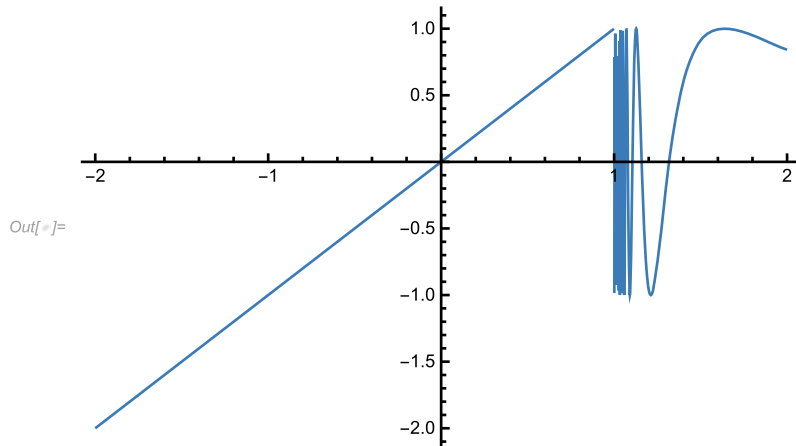


Figure 4.5

We can check that the limit from above does not exist

```
In[.]:= Limit[Piecewise[{{x, x <= 1}, {Sin[1/(x - 1)],
x > 1}}, x -> 1, Direction -> "FromAbove"]
```

```
Out[.]:= Indeterminate
```

#### 4.4.1 Example: the Dirichlet function

One can define in Mathematica<sup>®</sup> more complicated functions, for example the famous Dirichlet function, which is not continuous at any point:

```
In[.]:= dirichlet[x_] := Boole[Simplify[Element[x, Rationals]]]
```

This function is equal to 1 for rational numbers and is equal to 0 otherwise:

```
In[.]:= dirichlet /@ {Sqrt[2], 1/2, Pi}
```

```
Out[.]:= {0, 1, 0}
```

However, there is not much that Mathematica<sup>®</sup> can do with this kind of function. We cannot, for example, draw a graph of it since for plotting of graphs Mathematica<sup>®</sup> uses approximate numbers, and Mathematica<sup>®</sup> does not consider approximate numbers as either rational or irrational:

```
In[.]:= Element[1.1, Rationals]
```

```
Out[.]:= 1.1 ∈ Q
```

We can of course get some idea of the way the graph of such a function looks by using `DiscretePlot`. Here we combine two plots, one over a set of rationals and another one over irrationals.

```
In[ ]:= Show[DiscretePlot[dirichlet[t], {t, -1, 1, 1/10},
  Filling -> None], DiscretePlot[dirichlet[t], {t, -1, 1,
  Sqrt[2]/15}, Filling -> None]]
```

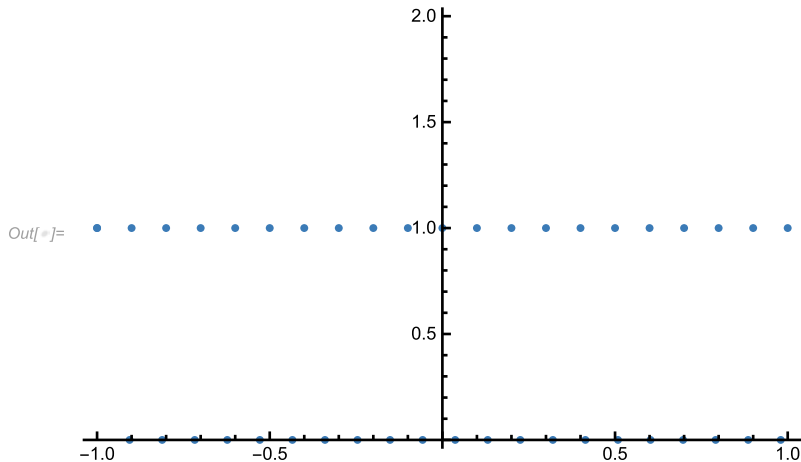


Figure 4.6

It is easy to see that the Dirichlet function is periodic and every rational number is a period.

```
In[ ]:= $Assumptions = {Element[q, Rationals],
  Element[s, Rationals], NotElement[u, Rationals],
  NotElement[v, Rationals]}
```

```
Out[ ]:= {q ∈ ℚ, s ∈ ℚ, u ∉ ℚ, v ∉ ℚ}
```

```
In[ ]:= dirichlet[s + q]
```

```
Out[ ]:= 1
```

```
In[ ]:= dirichlet[v + q]
```

```
Out[ ]:= 0
```

Similarly we can define a function that is continuous only at the point 0:

```
In[ ]:= dd[x_] = Piecewise[{{x, Element[x, Rationals]}}
```

```
Out[ ]:=  $\begin{cases} x & x \in \mathbb{Q} \\ 0 & \text{True} \end{cases}$ 
```

```
In[ ]:= Show[DiscretePlot[dd[t], {t, -1, 1, 1/10},
  Filling -> None], DiscretePlot[dd[t],
  {t, -1, 1, Sqrt[2]/15}, Filling -> None]]
```

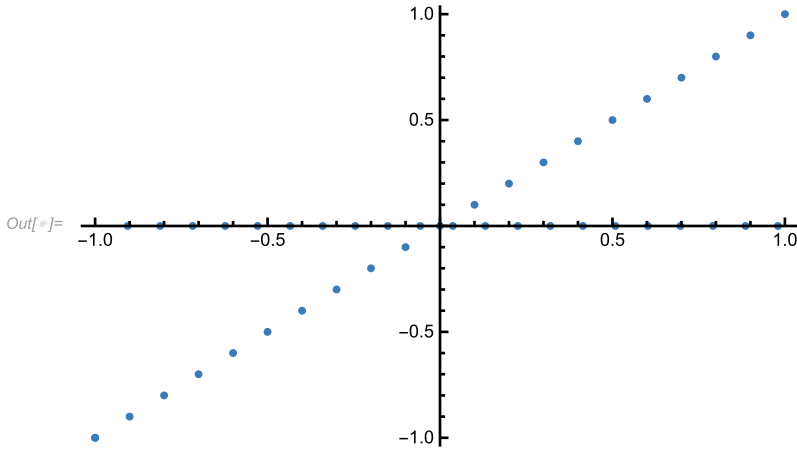


Figure 4.7

## 4.5 The main theorems on continuous functions

In this section we will discuss several important theorems on continuous functions.

**Theorem 11** (The Weierstrass theorem). *Let  $I$  be a closed interval  $[a, b]$ . A continuous function  $f : I \rightarrow \mathbb{R}$  has a maximum and a minimum value (i. e., it attains its supremum and infimum).*

Let  $I$  denote an interval (which could be infinite). We say that a function  $f : I \rightarrow X$  has the Darboux property if for any  $a, b \in I$  and any real number  $p$  between  $f(a)$  and  $f(b)$  there exists a real number  $q$  between  $a$  and  $b$  such that  $f(q) = p$ .

A discontinuous function can have the Darboux property; for example, the function

$$f(x) = \begin{cases} x, & x \leq 1, \\ \sin(1/(x-1)), & x > 1, \end{cases}$$

which appeared in Section 4.4, clearly has the Darboux property although it is discontinuous at 1. However, we have the following theorem (see [14, Theorem 4.14]).

**Theorem 12** (Intermediate value property). *A continuous function  $f : I \rightarrow \mathbb{R}$  has the Darboux property.*

A form of the theorem (easily proved to be equivalent) that is very useful in applications is the following. If  $a$  and  $b$  are two unequal real numbers with  $a < b$  and  $f$  is a continuous function on  $[a, b]$  such that the values of  $f$  at  $a$  and  $b$  have opposite signs, then the equation  $f(x) = 0$  has a solution in the interval  $[a, b]$ .

### 4.5.1 Example

Consider the following problem: find a solution of the equation  $x^3 - 3x = -1$  with precision  $1/100$  (i. e., a number  $x$  such that there is an exact root of the equation  $p$  with  $|x - p| < 1/100$ ).

We consider the function

```
In[.]:= Clear[f]; f[x_] := x^3 - 3*x + 1
```

and look for solutions of  $f(x) = 0$ . **Mathematica**<sup>®</sup> has several built-in functions that can solve this problem with arbitrary precision (see below), but we shall use the Intermediate Value Theorem and a simple **While** loop. Before we start we need to find two real numbers  $a$  and  $b$  such that  $f(a)$  and  $f(b)$  have different signs. Without looking at the graph of  $f$  this needs some guessing, e. g.,

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

```
Out[.]:= 1
```

```
In[.]:= f[1]
```

```
Out[.]:= -1
```

So we know that there is a root between 0 and 1. We can now start at 0 and keep adding  $1/100$  until we get to a number  $p$  such that  $f(p)$  is negative. Then there has to be a root in the interval  $(p - 1/100, p)$ , which will be our solution.

```
In[.]:= a = 0; While[f[a] > 0, a += 1/100]; {a - 1/100, a}
```

```
Out[.]:= {17/50, 7/20}
```

```
In[.]:= N[{17/50, 7/20}]
```

```
Out[.]:= {0.34, 0.35}
```

We check the graph:

```
In[.]:= Plot[f[x], {x, 17/50, 7/20}]
```

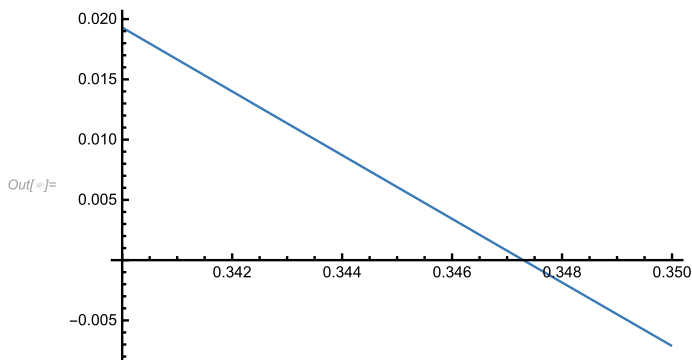


Figure 4.8

Thus any number in the interval  $(17/50, 7/20)$  is a solution to the problem.

There is another natural algorithm we can use. We start as above, by finding two points  $a$  and  $b$  (e. g., 0 and 1) where the function has different signs. We then divide the interval  $[a, b]$  into two equal halves,  $[a, c]$  and  $[c, b]$ . If the signs of  $f$  at  $a$  and  $c$  are different, we choose the interval  $[a, c]$  otherwise we choose  $[c, b]$ . We then subdivide again and continue the process until the width of the interval (denoted below by  $d$ ) is less than  $1/100$ . We then return the midpoint:

```
In[.]:=Module[{a = 0, b = 1, c = 1/2, d = 1},
  While[d >= 1/100, If[Sign[f[a]*f[c]] == -1,
    a = a; b = c; c = (a + b)/2; d = b - a,
    a = c; b = b; c = (a + b)/2; d = b - a]]; N[c, 3]]
```

**Out[.]**:= 0.348

Of course Mathematica<sup>®</sup> has a number of built-in functions that can solve this problem (with the required precision or much higher if desired). As the equation is a polynomial one, we could use the function `NSolve`:

```
In[.]:=NSolve[f[x] == 0, x, WorkingPrecision -> 2.6]
```

**Out[.]**:= {{x -> -1.88}, {x -> 0.347}, {x -> 1.53}}

The function `FindRoot` works even with non-algebraic equations but needs a starting value:

```
In[.]:=FindRoot[f[x] == 0, {x, 0}, WorkingPrecision -> 2.6]
```

**Out[.]**:= {x -> 0.347}

Mathematica<sup>®</sup> can find the solutions of polynomial equations exactly and also with an arbitrary precision:

```
In[.]:=rts = x /. Solve[x^3 - 3*x == -1, x, Reals]
```

```
Out[.]:= {Root[1 - 3*#1 + #1^3 & , 1], Root[1 - 3*#1 + #1^3 & , 2],
  Root[1 - 3*#1 + #1^3 & , 3]}
```

```
In[.]:=N[rts, 2.6]
```

**Out[.]**:= {-1.88, 0.347, 1.53}

Note that we can also find a rational approximation to the exact root above:

```
In[.]:=Rationalize[rts[[2]], 1/100]
```

**Out[.]**:=  $\frac{5}{14}$

## 4.6 Inverse functions and their continuity

We say that a function  $f : A \rightarrow B$  is *injective* or *one-to-one* if  $f(x) = f(y)$  implies  $x = y$ . We know that  $f$  is injective if and only if it has an inverse function  $f^{-1} : f(A) \rightarrow A$  (where  $f(A)$  is the image of  $A$  under  $f$ ).

Let  $I$  be an interval. It follows easily from the intermediate value property theorem that a continuous function  $f : I \rightarrow \mathbb{R}$  is injective if and only if it is either strictly monotone increasing ( $x < y$  implies  $f(x) < f(y)$ ) or strictly monotone decreasing ( $x < y$  implies  $f(x) > f(y)$ ). In this situation one can show that the inverse function is also continuous. This does not have to be true if the domain of  $f$  is not an interval. Consider, for example, the continuous functions  $f : A \rightarrow B$  where  $A = (0, 1) \cup \{2\}$ ,  $B = (0, 1]$ ,  $f(x) = x$  for  $0 < x < 1$  and  $f(2) = 1$ . Then  $f$  has an inverse function  $g = f^{-1}$ , which is defined by  $g(x) = x$ ,  $x \in (0, 1)$  and  $g(1) = 2$ , but it is not a continuous function (it does not have the Darboux property). We can illustrate this example as follows:

```
In[.]:= Graphics[{{PointSize[0.02], Line[{{0, 0}, {1, 0}}]},
  Point[{2, 0}], Line[{{0, 0}, {0, 1}}], Point[{0, 1}],
  Arrow[{{0.5, 0}, {0, 0.5}}], Arrow[{{0.98, 0},
  {0, 0.98}}], Arrow[{{2, 0}, {0, 1}}]}]
```

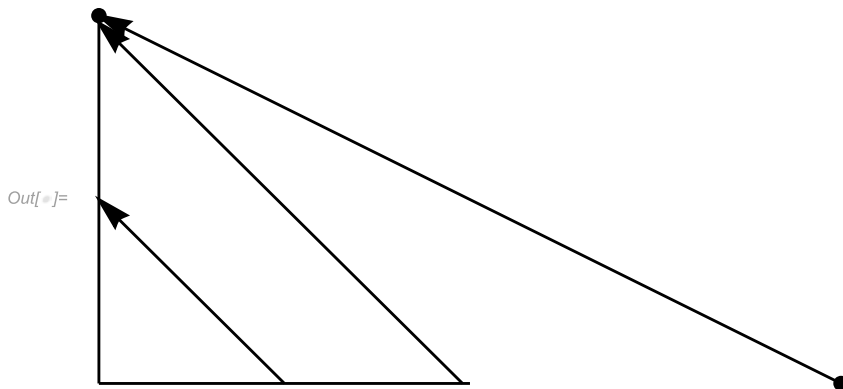


Figure 4.9

Mathematica<sup>®</sup> has a built-in function `InverseFunction` which will return the inverse of certain (not very complicated) functions:

```
In[.]:= InverseFunction[Sin]
Out[.]:= ArcSin

In[.]:= InverseFunction[Function[x, 3*x + 1]]
Out[.]:= Function[x, 1/3 (-1 + x)]
```

Even when Mathematica<sup>®</sup> cannot give an explicit form of the inverse function, one can use it for computation and plotting:

```

In[ ]:= f = Sin[#1] - #1 & ;
In[ ]:= g = InverseFunction[f];
In[ ]:= Plot[{f[x], g[x]}, {x, -2, 2}]

```

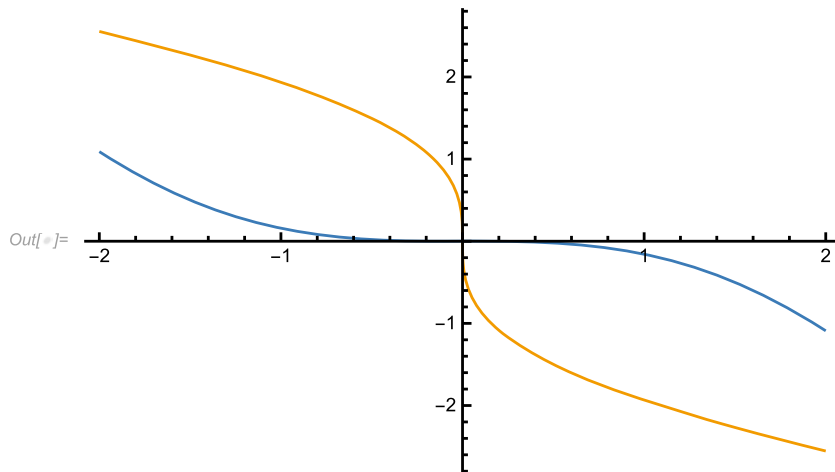


Figure 4.10

## 4.7 Example: recursive sequences and continuity

We can now revisit the topic of recursive sequences. Suppose we are given a sequence defined by a system of recurrence equations,

$$a_1 = x, \quad a_{n+1} = f(a_n),$$

where  $f$  is a continuous function. Then if  $a_n$  is convergent to a limit  $a \in \mathbb{R}$ , and  $a$  has to be a fixed point of  $f$ . This follows immediately from the continuity of  $f$  since

$$f(a) = f\left(\lim_{n \rightarrow \infty} a_n\right) = \lim_{n \rightarrow \infty} f(a_n) = \lim_{n \rightarrow \infty} a_{n+1} = a.$$

Now let us assume that the function  $f$  has two fixed points  $a$  and  $b$  and it is defined and one-to-one (injective) on  $[a, b]$ . Then since  $f$  is monotonic, clearly  $f([a, b]) = [a, b]$ . In other words, if our recursive sequence starts at some  $x$  in between two fixed points, the whole sequence has to remain in between these two fixed points. Moreover, since an injective function has to be monotonic, one of these points has to be the limit, depending on whether the sequence is increasing or decreasing.



Let us illustrate this with an example. Consider the function given by

```
In[.]:= Clear[f]; f[x_] := 3 - 1/x
```

There are only two fixed points:

```
In[.]:= x /. Solve[f[x] == x, x]
Out[.]:= {1/2 (3 - Sqrt[5]), 1/2 (3 + Sqrt[5])}
```

which means these are the only possible limits of the sequence

$$a_1 = x, \quad a_{n+1} = 3 - \frac{1}{a_n}.$$

Let us try to analyze the behavior of this sequence for various starting real numbers  $x$ . First we need to deal with the problem that for certain starting values  $x$ , such as  $1/3$ , a term of the sequence can become 0, which means that the subsequent terms will not be defined. We will say that the sequence explodes in such a case and we need to exclude such points. The set of points at which the sequence explodes satisfies an inverse recurrence relation:

```
In[.]:= Solve[a[n + 1] == 3 - 1/a[n], a[n]]
Out[.]:= {{a[n] -> 1/(3 - a[n + 1])}}
```

So we can define

$$b_1 = 0, \quad b_{n+1} = \frac{1}{3 - b_n}.$$

We can solve this with `RSolve` and find that we get an increasing sequence of points with limit

```
In[.]:= Limit[Simplify[b[n] /. RSolve[{b[n + 1] == 1/(3 - b[n]),
    b[1] == 0}, b[n], n]], n -> Infinity]
Out[.]:= {1/2 (3 - Sqrt[5])}
```

Let us now try to consider what will happen if we start at any other point. We know that the function must send the interval  $[c_1, c_2] = [(3 - \sqrt{5})/2, (3 + \sqrt{5})/2]$  to itself. We only need to know where it is increasing:

```
In[.]:= Reduce[f[x] >= x, x]
Out[.]:= x < 0 || 1/2 (3 - Sqrt[5]) <= x <= 1/2 (3 + Sqrt[5])
```

So the function is increasing in the interval  $[c_1, c_2]$ , hence if we start at any point of  $(c_1, c_2]$ , the limit must be  $c_2$ . Of course if we start at  $c_1$  the sequence is constant. What about the other starting points? If we start to the right of  $c_2$ , then the function will be decreasing, but as it cannot “jump” into the interval or over it (because of continuity

and injectivity), the sequence has to remain to the right of  $c_2$  and converge to  $c_2$  from the right. Moreover, if  $x$  is negative,  $f(x)$  has to be positive and is greater than 3 (the “jump” is possible because  $f$  cannot be continuously defined at 0!), and after that the sequence will converge to  $c_2$  from the right. Finally, if we start between 0 and  $c_1$ , then the sequence will decrease until it becomes negative (because it cannot be bounded below as otherwise it would have to have another limit), and then it will jump to the right side of  $c_2$  and converge to it from above. The conclusion is that the sequence either explodes or converges to  $c_2 = (3 + \sqrt{5})/2$ , except when it starts at  $c_1 = (3 - \sqrt{5})/2$  as it remains there. We suggest the reader to illustrate the behavior of this dynamical system by using Manipulate as in Chapter 2.

## 4.8 Uniform continuity and the Lipschitz property

A function  $f : A \rightarrow \mathbb{R}$  is said to be *uniformly continuous* if for every  $\varepsilon > 0$  there is a  $\delta > 0$  such that for  $x, y \in A$  with  $|x - y| < \delta$  we have  $|f(x) - f(y)| < \varepsilon$ .

Note that when we write the definition in terms of quantifiers, the difference between ordinary continuity and uniform continuity amounts to the difference in the order of quantifiers: for continuity we have

$$\forall \varepsilon, \varepsilon > 0 \forall x, x \in A \exists \delta, \delta > 0 \forall y, y \in A \wedge |x - y| < \delta |f(x) - f(y)| < \varepsilon$$

and for uniform continuity we have

$$\forall \varepsilon, \varepsilon > 0 \exists \delta, \delta > 0 \forall x, x \in A \forall y, y \in A \wedge |x - y| < \delta |f(x) - f(y)| < \varepsilon.$$

Clearly uniform continuity implies ordinary continuity but the converse is not true. We can demonstrate this using Mathematica<sup>®</sup>. Since the condition is written entirely in terms of quantifiers, it should be possible to prove it for polynomial and rational functions by quantifier elimination. Let us compare two functions, the function

```
In[.]:= Clear[f, g]; f[x_] := x^2
```

defined on  $\mathbb{R}$  and the function

```
In[.]:= g[x_] := Sqrt[x]
```

defined on  $\mathbb{R}^+$ . As we know both functions are continuous:

```
In[.]:= Reduce[ForAll[{e, x}, e > 0, Exists[d, d > 0,
  ForAll[{y}, Element[{y}, Reals] &&
  Abs[x - y] < d, Abs[f[x] - f[y]] < e]]]]
```

```
Out[.]:= True
```

```
In[.]:= Reduce[ForAll[{e, x}, e > 0 && x >= 0,
  Exists[d, d > 0, ForAll[{y}, y >= 0 &&
  Abs[x - y] < d, Abs[g[x] - g[y]] < e]]]]
```

```
Out[.]:= True
```

However, the first one is not uniformly continuous while the second one is:

```
In[.]:= Reduce[ForAll[e, e > 0, Exists[d, d > 0,
  ForAll[{x, y}, Element[{x, y}, Reals] &&
  Abs[x - y] < d, Abs[f[x] - f[y]] < e]]]]
```

```
Out[.]:= False
```

```
In[.]:= Reduce[ForAll[e, e > 0, Exists[d, d > 0,
  ForAll[{x, y}, x >= 0 && y >= 0 &&
  Abs[x - y] < d, Abs[g[x] - g[y]] < e]]]]
```

```
Out[.]:= True
```

We have the following theorem [14, Theorem 3.38].

**Theorem 13.** *Every continuous function  $f$  on a bounded closed interval  $[a, b]$  is uniformly continuous therein.*

We can demonstrate this by restricting the function  $x \mapsto x^2$  to the unit interval:

```
In[.]:= Reduce[ForAll[e, e > 0, Exists[d, d > 0,
  ForAll[{x, y}, 0 <= x <= 1 && 0 <= y <= 1 &&
  Abs[x - y] < d, Abs[f[x] - f[y]] < e]]]]
```

```
Out[.]:= True
```

A function  $f : A \rightarrow \mathbb{R}$  satisfies the *Lipschitz condition* if there exists a constant  $c > 0$  such that  $|f(x) - f(y)| \leq c|x - y|$ . Obviously, every function that satisfies the Lipschitz condition is uniformly continuous.

Let us show that the function  $x \mapsto \sqrt{x}$  does not satisfy the Lipschitz condition on  $[0, \infty)$  but does so on  $[1, \infty)$ :

```
In[.]:= Reduce[Exists[c, c > 0, ForAll[{x, y}, x >= 0 && y >= 0,
  Abs[Sqrt[x] - Sqrt[y]] <= c*Abs[x - y]], Reals]
```

```
Out[.]:= False
```

```
In[.]:= Reduce[Exists[c, c > 0, ForAll[{x, y}, x >= 1 && y >= 1,
  Abs[Sqrt[x] - Sqrt[y]] <= c*Abs[x - y]], Reals]
```

```
Out[.]:= True
```

In fact we can actually, instead of deciding whether such a constant  $c$  exists, find its value by removing the function `Exists` from the expression above:

```
In[.]:= Reduce[ForAll[{x, y}, x >= 1 && y >= 1,
  Abs[Sqrt[x] - Sqrt[y]] <= c*Abs[x - y]], Reals]
```

```
Out[.]:= c >= 1/2
```

Note that this time we needed to add the domain Reals to Reduce:

```
In[.]:= Reduce[Exists[c, c > 0, ForAll[{x, y}, x >= 0 &&
    y >= 0, Abs[Sqrt[x] - Sqrt[y]] <= c*Abs[x - y]]]]
...Reduce: Reduce was unable to prove that a radical of
an expression containing only real variables and parameters
is real-valued. If you are interested only in solutions for
which all radicals contained in the input are real-valued,
use Reduce with domain argument Reals.
```

```
Out[.]:= Reduce [∃c, c>0∀{x,y}, x≥0&&y≥0Abs[√x - √y] ≤ c Abs[x - y]]
```

This happens whenever functions that can take non-real values, such as roots, appear in an expression to which Reduce is applied.

The reader should prove all the above results by hand. See also [14] for other examples.

#### 4.8.1 Example

Finally, we consider an example which cannot be solved by using quantifier elimination (because it involves a non-algebraic function).

Consider the following problem: decide whether the function  $\log(x)$  has the Lipschitz property or is uniformly continuous (a) on  $(0, \infty)$  and (b) on  $[a, \infty)$  for  $a > 0$ .

This is the kind of situation where Mathematica<sup>®</sup>'s dynamic graphic capabilities can be very useful. The dynamic graphic below shows that we can find  $\varepsilon > 0$  (in this case  $\varepsilon = 1$ ) and a pair of points  $x, y > 0$  such that  $|x - y| < \delta$  is arbitrarily small while  $|\log(x) - \log(y)| = \varepsilon$ . We can use the slider to make  $\delta$  smaller and see that the distance between  $\log(x)$  and  $\log(y)$  (represented by the vertical green line) does not change (here  $x = \delta$  and  $y = \delta/e$ ). Hence, the function is not uniformly continuous on the interval  $(0, \infty)$ .

```
In[.]:= Manipulate[Show[Plot[Log[x], {x, 0.0001, 1.2},
    PlotRange -> {{0, 1.3}, {-5, 1}}],
    Graphics[{Red, Opacity[0.3], Rectangle[{d/E, 0},
    {d, Log[d/E]}], Opacity[1], Thickness[0.01],
    Blue, Line[{d, 0}, {d, Log[d/E]}], Line[{d/E, 0},
    {d/E, Log[d/E]}], Green, Line[{d, Log[d]},
    {d, Log[d/E]}]}], AxesOrigin -> {0, 0}, PlotRange ->
    {{0, 1.3}, {-5, 1}}, {{d, 1, "d"}, 0.001, 1}]
```

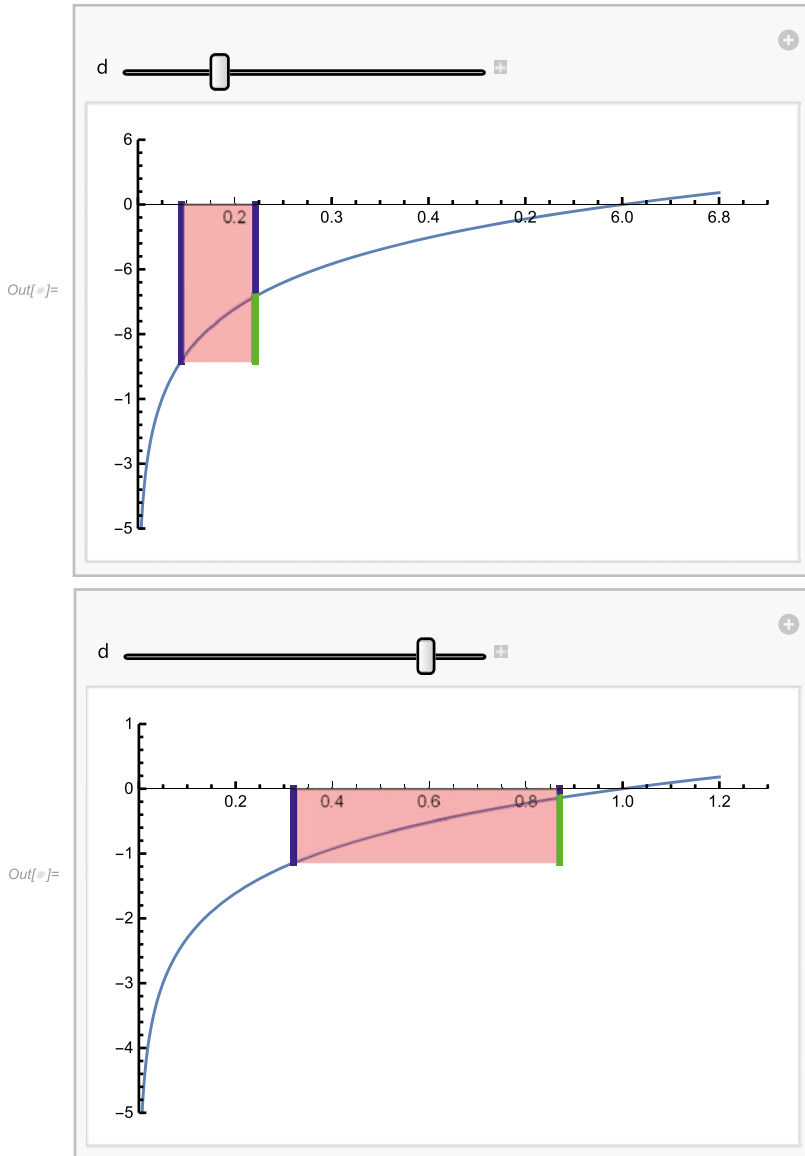


Figure 4.11

We can also see that if  $x, y > a$  for some  $a > 0$ , then the above argument no longer works because we cannot move  $x$  and  $y$  arbitrarily close to 0. In fact, we can show that when restricted to  $[a, \infty)$  the function  $\log$  has the Lipschitz property. Indeed, suppose that  $y > x$ . Then

$$\log\left(\frac{y}{x}\right) = \log\left(\frac{x + (y - x)}{x}\right) = \log\left(1 + \frac{y - x}{x}\right) \leq \log\left(1 + \frac{y - x}{a}\right).$$

We now use the fact that  $\log(z + 1) \leq z$  for  $z \geq -1$ , which Mathematica<sup>®</sup> can prove:

**In[.]**:= Reduce[Log[1 + z] <= z, z]

**Out[.]**:= z > -1

Hence

$$\log\left(\frac{y}{x}\right) \leq \frac{y-x}{a}$$

on  $[a, \infty)$ , which means that  $\log$  has the Lipschitz property with the Lipschitz constant equal to  $1/a$ .

## 5 Differentiation

In this chapter we consider the concept of differentiability and derivative of a function at a point and on an interval. We see that differentiability is a special case of continuity, but of another function called the difference quotient. We then consider various properties of derivatives and applications of differentiation, in particular to finding global and local extrema and to convexity of functions. We also illustrate some aspects of the use of patterns and transformation rules in Mathematica<sup>®</sup>'s programming language by the example of a user-defined derivative.

### 5.1 Difference quotient and derivative of a function

Let  $I$  be an open interval in  $\mathbb{R}$ ,  $a \in I$  and let  $f : I \rightarrow \mathbb{R}$  be a function. Its *difference quotient* is a function  $Q : I \setminus \{a\} \rightarrow \mathbb{R}$  given by

```
In[ ]:= Q[a_, f_][x] := DifferenceQuotient[f[x], {x, a - x}]
In[ ]:= Q[a, f][x]
Out[ ]:=  $\frac{f[a] - f[x]}{a - x}$ 
```

Note that the difference quotient is not defined at the point  $a$ . We are going to consider the following question: can  $Q$  be extended to a function  $\phi : I \rightarrow \mathbb{R}$  such that  $\phi$  is continuous at  $a$ ? If the answer is “yes”, then we say that  $f$  is differentiable at  $a$  and  $\phi(a)$  is its derivative at  $a$ .

In the interactive graphic below we show the graphs of the difference quotients of the functions  $x \mapsto |x|^k$ , where  $k \in \{1/2, 1, 3/2, 2\}$ . We see that the first two functions  $\sqrt{|x|}$  and  $|x|$  are not differentiable at 0 while the remaining functions are all differentiable.

```
In[ ]:= Manipulate[Plot[Evaluate[Q[0, Abs[#1]^k & ][x]],
{x, -1, 1}, Exclusions -> 0, AxesOrigin -> {0, 0}],
{{k, 1/2, "k"}, 1/2, 2, 1/2, Appearance -> "Labeled"},
SaveDefinitions -> True]
```

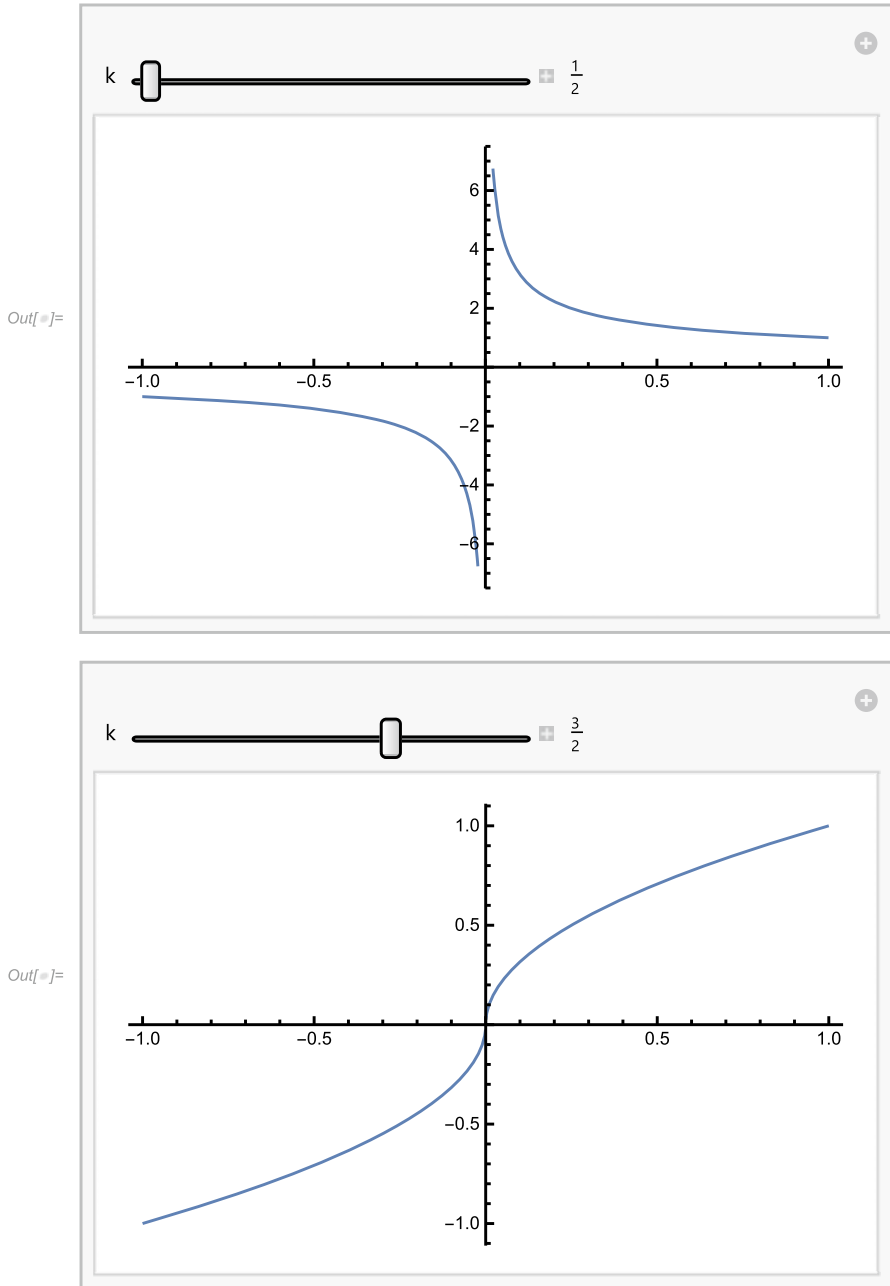


Figure 5.1



The definition can be expressed in the more usual way in terms of limits:  $f$  is *differentiable* at  $a$  if the limit

$$\lim_{x \rightarrow a} Q(a, f)(x) = \lim_{x \rightarrow a} \frac{f(x) - f(a)}{x - a}$$

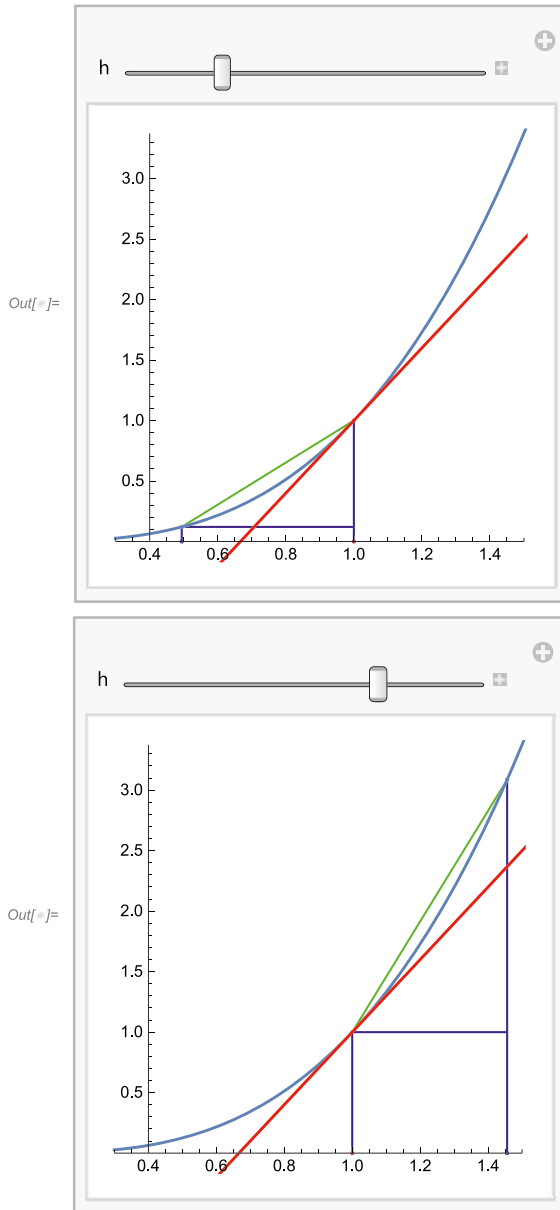
exists, in which case the limit is called the *derivative* of  $f$  at  $a$ . The derivative of a function  $f$  at a point  $a$  is usually denoted as  $f'(a)$  or  $(\frac{df(x)}{dx})|_{x=a}$ . If  $f$  is differentiable on an open interval  $I$ , then the derivative is also a function  $f' : I \rightarrow \mathbb{R}, x \mapsto f'(x)$ .

The second order derivative is defined at a point  $a$  whenever the first order derivative exists in a neighborhood of  $a$  and is differentiable (and hence also continuous) at  $a$ . We then say that  $f$  is twice differentiable at  $a$  and denote its second order derivative by  $f''(a)$  or  $(\frac{d^2f(x)}{dx^2})|_{x=a}$ . In general, we define the  $n$ -th order derivative of a function  $f$  as  $f^{(n)} = (f^{(n-1)})'$ . If  $f$  is  $n$  times differentiable and the  $n$ -th order derivative is continuous, then  $f$  is said to be of class  $C^n$ .

Geometrically the difference quotient represents the secant line of the graph of  $f$  connecting the points  $(x, f(x))$  and  $(a, f(a))$ . As the point  $x$  approaches  $a$ , the secant approaches the tangent to the graph of  $f$  at  $(a, f(a))$ :

```
In[ ]:= f[t_] := t^3
```

```
In[ ]:= Manipulate[Show[Graphics[{Red, PointSize[0.01],
  Point[{1, 0}], Point[{1, f[1]}], Blue,
  Point[{1 + t, f[1 + t]}], Point[{1 + t, 0}],
  Thickness[0.005], Line[{{1, 0}, {1, f[1]}}],
  Line[{{1 + t, f[1 + t]}, {1 + t, 0}}],
  Line[{{1, If[t > 0, f[1], f[1 + t]}],
  {1 + t, If[t > 0, f[1], f[1 + t]}}], Green,
  Line[{{1, f[1]}, {1 + t, f[1 + t]}}]],
  Plot[f[x], {x, 0, 2}], Plot[f[1] +
  Derivative[1][f][1]*(x - 1), {x, -1, 2},
  PlotStyle -> Red], Axes -> True,
  PlotRange -> {{0.3, 1.5}, {0, f[1.5]}},
  AspectRatio -> 1], {{t, 0.5, "h"}, -1, 1},
  SaveDefinitions -> True]
```



**Figure 5.2**

It is easy to see that if a function is differentiable at a point  $a$ , it has to be continuous there. Indeed, for any  $a_n \rightarrow a$  we have

$$\lim_{n \rightarrow \infty} f(a_n) = \lim_{n \rightarrow \infty} \left( \frac{(a_n - a)(f(a_n) - f(a))}{a_n - a} + f(a) \right) = f(a).$$

When the limit  $\lim_{x \rightarrow a^-} (f(x) - f(a)) / (x - a)$  exists, the function  $f$  is said to be *left differentiable* at  $a$  and the limit is called its *left hand derivative*. If the limit  $\lim_{x \rightarrow a^+} (f(x) - f(a)) / (x - a)$  exists, then  $f$  is said to be *right differentiable* at  $a$  and the limit is its *right hand derivative*. A function is *differentiable* if and only if both its left and right hand derivatives exist and are equal.

The following is an example of a function which has both right and left hand derivatives at 1 but they are not equal (hence the function is not differentiable):

```
In[.]:=Plot[Piecewise[{{x, x <= 1}, {x^2, x > 1}}, {x, 0, 2}]
```

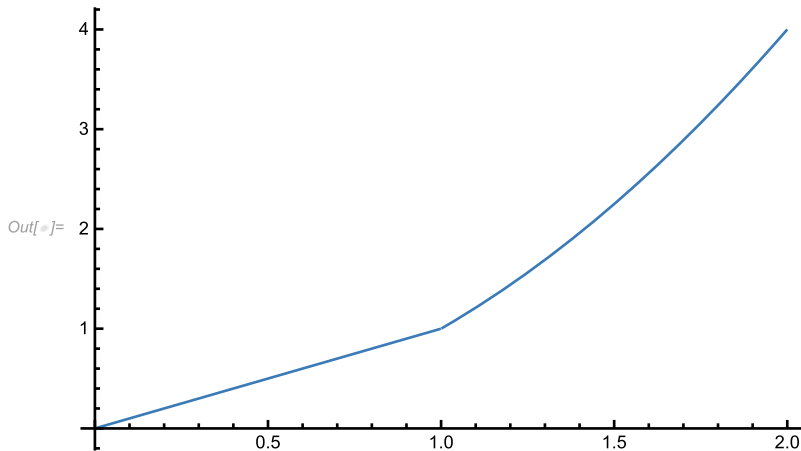


Figure 5.3

Indeed, the right hand derivative is

```
In[.]:=Limit[(Piecewise[{{x, x <= 1}, {x^2, x > 1}}
- 1)/(x - 1), x -> 1, Direction -> "FromAbove"]
```

```
Out[.]:= 2
```

while the left hand derivative is

```
In[.]:=Limit[(Piecewise[{{x, x <= 1}, {x^2, x > 1}}
- 1)/(x - 1), x -> 1, Direction -> "FromBelow"]
```

```
Out[.]:= 1
```

On the other hand, the function whose graph is shown below has neither left nor right hand derivatives:

```
In[.]:=Manipulate[Plot[Piecewise[{{x*Sin[1/x], x != 0}},
{x, -r, r}, PlotRange -> All], {{r, 1, "r"}, 0.01,
1, Appearance -> "Labeled"}, SaveDefinitions -> True]
```

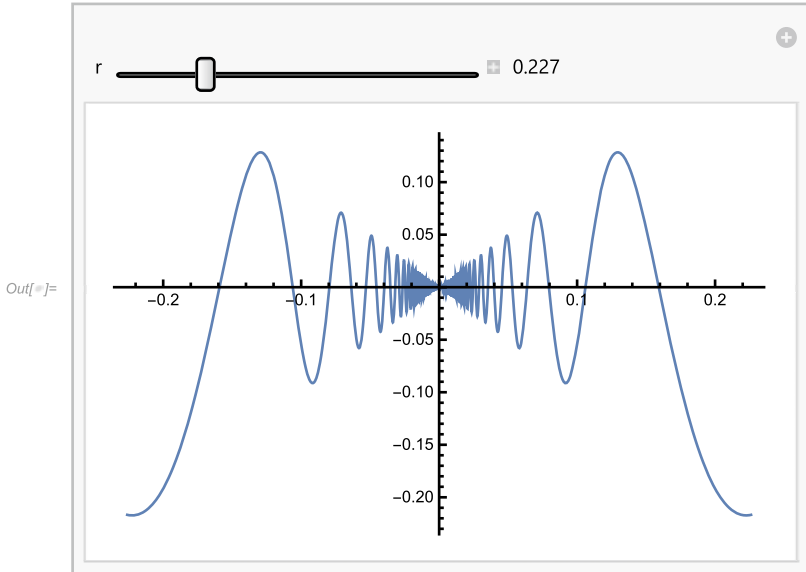


Figure 5.4

When the tangent to the graph at a point is vertical, the derivative at that point exists and is equal to  $\infty$  but the function is said to be non-differentiable there:

```
In[ ]:= Plot[Surd[x, 3], {x, -1, 1}]
```

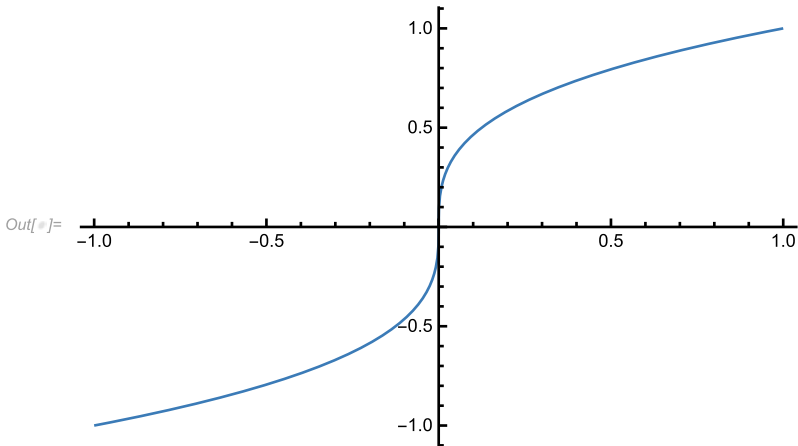


Figure 5.5

Note, however, that from the point of view of differential geometry the curve which represents the graph is regular (or smooth).

The reason why we use the function `Surd` rather than  $x^{1/3}$  (`Power[x, 1/3]`) is that in `Mathematica`®  $x^{1/n}$  is not a real number for negative  $x$  and odd integer  $n$  greater than one. For example,

```
In[.]:= N[(-1)^(1/3)]
Out[.]:= 0.5 + 0.866025 I
```

This is due to the fact that in complex analysis any number has  $n$  complex roots. When  $n$  is odd, then any negative number always has just one real root, but this is not the “principal root”, used in complex analysis. In the case  $n = 3$  the principal root is a complex number with a positive imaginary part. In order to be consistent with this convention `Mathematica`®’s function `Power` always returns this (complex) principal root. Therefore, if one wants to get a real number, one needs to use the function `Surd` (or in this case the function `CubeRoot`), defined specifically for this purpose:

```
In[.]:= Surd[-1, 3]
Out[.]:= -1
```

```
In[.]:= CubeRoot[-1]
Out[.]:= -1
```

Note that in `TraditionalForm` the expression above will look like a cube root  $\sqrt[3]{-1}$ .

In the example below the derivative from the left is  $-\infty$  and the one from the right is  $\infty$ . The tangent is also vertical but the derivative does not exist.

```
In[.]:= Plot[Sqrt[Abs[x - 2]], {x, 0, 4}]
```

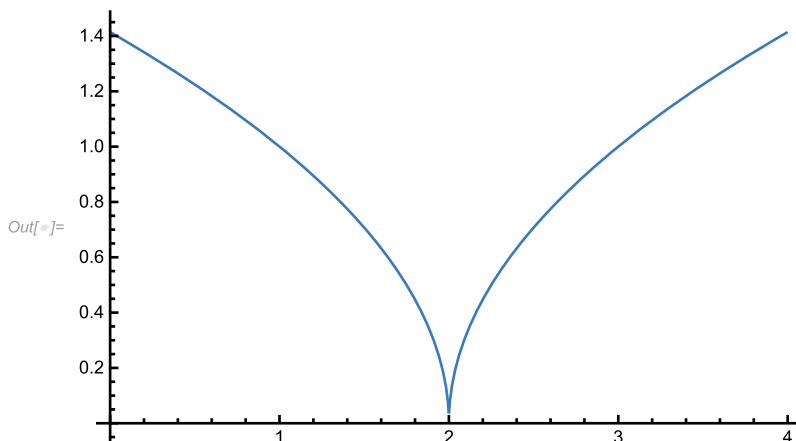


Figure 5.6

Note that in this case the curve is not smooth.

## 5.2 Differentiation in Mathematica®

Mathematica®’s approach to differentiation is a little non-standard, since Mathematica® has two distinct notions of derivative, which we will call “derivative of an expression” and “derivative of a function”. Each can be essentially used for the same purpose but they are conceptually different and each is more convenient in a certain context. It is therefore advisable to learn to use both, which is why we devote the whole section to this topic. In particular, we will explain why instead of writing

$$\mathbf{In[.]} := (x^2)'$$

$$\mathbf{Out[.]} := (x^2)'$$

we have to write either

$$\mathbf{In[.]} := D[x^2, x]$$

$$\mathbf{Out[.]} := 2 x$$

or

$$\mathbf{In[.]} := (\text{Function}[x, x^2])'$$

$$\mathbf{Out[.]} := \text{Function}[x, 2 x]$$

and instead of

$$\mathbf{In[.]} := \text{Sin}[x]'$$

$$\mathbf{Out[.]} := \text{Sin}[x]'$$

we must write

$$\mathbf{In[.]} := \text{Sin}'[x]$$

$$\mathbf{Out[.]} := \text{Cos}[x]$$

### 5.2.1 Differentiation of expressions using D

Recall that in Mathematica® “everything is an expression” and an expression always has the form  $F[x_1, x_2, \dots, x_n]$ , where  $F$  is called the head of the expression and each  $x_i$  is an argument. Both the head and the arguments can themselves be either atoms or expressions of the above form. Here we will only consider certain special kinds of expressions which are related to mathematical functions. One can think of them as arising by applying a mathematical function to a certain number of variables. For example  $x^2 + 2x + 1$ ,  $2x + a$ ,  $\text{Sin}[x \text{Exp}[xy^2]] + 3$  are such expressions. It is important to keep in mind the distinction between expressions and the corresponding functions (of one variable); for example, the function  $\text{Function}[x, x^2 + 2x + 1]$  corresponds to the expression  $x^2 + 2x + 1$ . In this section we will only consider functions of one variable, so if an expression contains two symbols, like  $2x + a$ , we think of  $a$  as a constant and take as the corresponding function  $\text{Function}[x, 2x + a]$  (and not the function of two variables  $\text{Function}[\{x, a\}, 2x + a]$ ).

Expressions are differentiated in Mathematica® using the symbol `D`. The derivative of an expression `expr` with respect to a variable `x` is written as `D[expr, x]`:

**In[.]**:= `D[x*Sin[x], x]`

**Out[.]**:= `x Cos[x] + Sin[x]`

Note that `D` treats every symbol in an expression that does not explicitly depend on `x` as a constant:

**In[.]**:= `D[a*x^2 + b + 2, x]`

**Out[.]**:= `2 a x`

Otherwise if, for instance, we want `b` to depend on `x`, we need to write either

**In[.]**:= `D[a*x^2 + b[x] + 2, x]`

**Out[.]**:= `2 a x + b'[x]`

or

**In[.]**:= `D[a*x^2 + b + 2, x, NonConstants -> b]`

**Out[.]**:= `2 a x + D[b, x, NonConstants -> {b}]`

There is another approach (which we mention only briefly as we will not use it) that uses the “total differential” `Dt` instead of `D`. `Dt` makes the opposite assumption to that of `D`, i. e., that everything is non-constant unless declared to be a constant by means of the option `Constants`:

**In[.]**:= `Dt[a*x, x]`

**Out[.]**:= `a + x Dt[a, x]`

**In[.]**:= `Dt[a*x, x, Constants -> a]`

**Out[.]**:= `a`

**In[.]**:= `Dt[a*x^2 + b, x, Constants -> {a, b}]`

**Out[.]**:= `2 a x`

If we want `a` to be treated as a constant globally, we can give it the `Attribute Constant`. The symbol `a` will then be treated by `Dt` as a constant until the kernel is quit or the `Attribute` is cleared:

**In[.]**:= `SetAttributes[a, Constant]`

**In[.]**:= `Dt[a x, x]`

**Out[.]**:= `a`

**In[.]**:= `ClearAttributes[a, Constant]`

If by using `D` we want to compute the value of the derivative of a function `f` at a point `a` we have to use a replacement rule, i. e., the following

**In[.]**:= `D[f[x], x] /. x -> a`

**Out[.]**:= `f'[a]`

computes the derivative  $f'(a)$ . Note that the substitution of  $a$  for  $x$  has to be done after differentiation, otherwise we will get 0. Of course the name of the variable can be any allowed name, i. e., the expressions

```
In[.]:= D[x^3, x] /. x -> 2
Out[.]:= 12
```

```
In[.]:= D[var^3, var] /. var -> 2
Out[.]:= 12
```

both return the same answer. However, we have to be sure that the variable  $x$  does not have an assigned value or we can use a scoping construct such as `Block` or `Module`, e. g.,

```
In[.]:= x = 1; D[x^3, x]
...General: 1 is not a valid variable.
Out[.]:=  $\partial_1 1$ 
```

```
In[.]:= Block[{x}, D[x^3, x] /. x -> 2]
Out[.]:= 12
In[.]:= Clear[x]
```

In general when differentiating with respect to  $x$  an expression of the form  $F[x]$ , where  $F$  is an atomic expression, `Mathematica`<sup>®</sup> treats the head of the expression as the name of a function and if it knows the derivative of  $F$ , for example  $G$ , then it returns  $G[x]$ :

```
In[.]:= D[Sin[x], x]
Out[.]:= Cos[x]
```

Otherwise it returns  $F'[x]$  (or in `FullForm` the output is `Derivative[1][F][x]`). The function `Derivative` will be discussed in Section 5.2.2.

```
In[.]:= D[Abs[x], x]
Out[.]:= Abs'[x]
```

```
In[.]:= % // FullForm
Out[.]//FullForm= Derivative[1][Abs][x]
```

Although `Abs` is a built-in function, `Mathematica`<sup>®</sup> does not have any value assigned as its derivative because in the complex plane `Abs` is not differentiable at any point, but in the real case we can give a formula which is valid for every point except 0. So we can define our own derivative that will suit our purpose:

```
In[.]:= Derivative[1][Abs] = Piecewise[{{1, #1 > 0},
{-1, #1 < 0}}, Indeterminate] &
Out[.]:=  $\begin{cases} 1 & \#1 > 0 \\ -1 & \#1 < 0 \ \& \\ \text{Indeterminate} & \text{True} \end{cases}$ 
```



Hence,

$$\begin{aligned} \mathbf{In[.]} &:= D[\text{Abs}[x], x] \\ \mathbf{Out[.]} &:= \begin{cases} 1 & x > 0 \\ -1 & x < 0 \\ \text{Indeterminate} & \text{True} \end{cases} \end{aligned}$$

### 5.2.2 Differentiation of functions using Derivative

`Derivative` is used to differentiate functions. Mathematica® allows one to define functions in two ways: by means of patterns and rules and as pure functions. We can use `Derivative` in both cases.

We first define a function by means of patterns:

```
In[.]:= cube[x_] := x^3
```

Mathematica® remembers the following information about this function:

```
In[.]:= ?cube
Out[.]:= Global`cube
        cube[x_] := x^3

In[.]:= DownValues[cube]
Out[.]:= {HoldPattern[cube[x_]] :> x^3}
```

We can now differentiate the function `cube` by

```
In[.]:= Derivative[1][cube]
Out[.]:= 3 #1^2 &
```

The result is a pure function that can be immediately applied to arguments. There is a quicker way to achieve the same result:

```
In[.]:= cube'
Out[.]:= 3 #1^2 &
```

Note that we cannot use `D` in this case:

```
In[.]:= D[cube, x]
Out[.]:= 0
```

However, we can form an expression `cube[x]` which we can differentiate:

```
In[.]:= D[cube[x], x]
Out[.]:= 3 x^2
```

Calculating derivatives at points using `Derivative` is very simple:

```
In[.]:= cube'[2]
Out[.]:= 12
```

Exactly the same approach works for pure functions, e. g.,

```
In[.]:= Function[x, x^3]'
Out[.]:= Function[x, 3 x^2]

In[.]:= #^3 & '[2]
Out[.]:= 12
```

The same method also works with built-in functions, for example:

```
In[.]:= {Sin', Cos'}
Out[.]:= {Cos[#1] &, -Sin[#1] &}

In[.]:= Map[Derivative[1], {Sin, Cos}]
Out[.]:= {Cos[#1] &, -Sin[#1] &}
```

Note that the derivative of a constant function is zero:

```
In[.]:= Derivative[1][1 &]
Out[.]:= 0 &
```

Using `Derivative` has a number of advantages over using `D`; for example, it is much simpler to plot the graph of a derivative:

```
In[.]:= Plot[{cube[x], cube'[x]}, {x, -1, 1}]
```

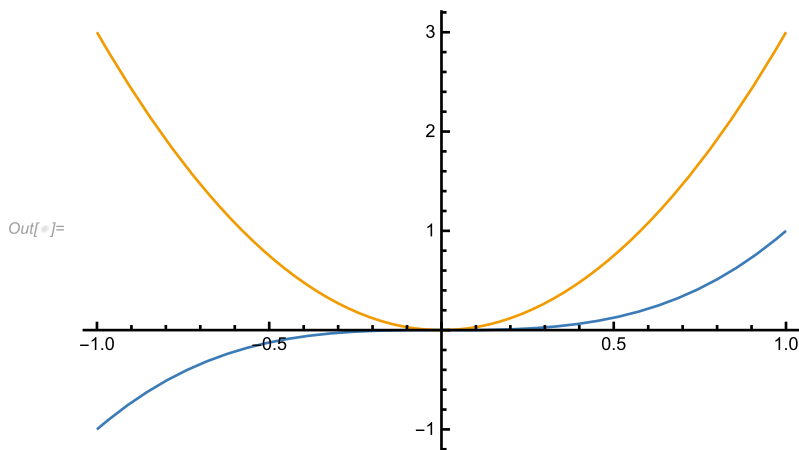


Figure 5.7

To achieve the same result by means of `D` we need to use a more complicated code:

```
In[.]:= Plot[Evaluate[{cube[x], D[cube[x], x]}], {x, -1, 1}]
```

In Mathematica<sup>®</sup> the  $n$ -th derivative of an expression  $f[x]$  is obtained by `D[f[x], {x, n}]` and of a function  $f$  as `Derivative[n][f]`. The last can also be entered for small  $n$  by using `'` several times (for instance for  $n = 2$  we write  $f''$ ).

### 5.2.3 Algebraic rules of differentiation

The basic properties of derivatives follow from properties of limits and allow us to turn differentiation into an algorithm. More precisely, these properties show that once we know the derivatives of a certain family of “basic” functions there is an algebraic algorithm that computes the derivatives of their sums, products, inverses and compositions. Of course, the derivatives of these basic functions have to be computed using analytic means, that is, by computing suitable limits of difference quotient functions. The basic rules [14, Theorem 3.48] can easily be verified for the functions `D` or `Derivative`:

(i) the derivative of a constant is zero

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[1, x] \\ \mathbf{Out}[\cdot] &:= 0 \end{aligned}$$

(ii) the sum rule

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[f[x] + g[x], x] \\ \mathbf{Out}[\cdot] &:= f'[x] + g'[x] \end{aligned}$$

(iii) the Leibniz rule (the product rule)

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[f[x] g[x], x] \\ \mathbf{Out}[\cdot] &:= g[x] f'[x] + f[x] g'[x] \end{aligned}$$

(iv) the quotient rule

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[f[x]/g[x], x] \text{ // Together} \\ \mathbf{Out}[\cdot] &:= \frac{g[x] f'[x] - f[x] g'[x]}{g[x]^2} \end{aligned}$$

The “chain rule” is the rule for computing the derivative of the composition of two functions [14, Theorem 3.51]:

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[f[g[x]], x] \\ \mathbf{Out}[\cdot] &:= f'[g[x]] g'[x] \end{aligned}$$

or

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Derivative}[1][\text{Composition}[f, g]] \\ \mathbf{Out}[\cdot] &:= f'[g[\#1]] g'[\#1] \& \end{aligned}$$

From the chain rule we can derive the very important rule for differentiation of inverse functions:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{InverseFunction}[f]' \\ \mathbf{Out}[\cdot] &:= \frac{1}{f' [f^{(-1)}[\#1]]} \& \end{aligned}$$

or

$$\begin{aligned} \mathbf{In}[\cdot] &:= D[\text{InverseFunction}[f][x], x] \\ \mathbf{Out}[\cdot] &:= \frac{1}{f' [f^{(-1)}[x]]} \end{aligned}$$

### 5.2.4 Example: user-defined derivative

A good exercise in Mathematica<sup>®</sup>'s pattern and rule-based programming is to write one's own version of `D` (or of `Derivative`):

```
In[.]:= d[Sin[x_], x_] := Cos[x]
      d[Cos[x_], x_] := -Sin[x]
      d[Exp[x_], x_] := Exp[x]
      d[Log[x_], x_] := 1/x
      d[f_, x_] /; FreeQ[f, x] := 0
      d[x_, x_] := 1
      d[(f_) + (g_), x_] := d[f, x] + d[g, x]
      d[(f_)*(g_), x_] := d[f, x]*g + f*d[g, x]
      d[(y_)^(a_), x_] := y^a*d[a*Log[y], x]
      d[(f_)[u_], x_] := Block[{v}, (d[f[v], v] /.
      v -> u)*d[u, x]]
```

The first four rules define how `d` operates on “basic functions”. We have included only the functions `Sin`, `Cos`, `Exp` and `Log`. Next we have the rule that says that the derivative of a constant (i. e., an expression that does not explicitly involve  $x$ ) is 0 and that the derivative of the identity function  $x \mapsto x$  is 1. Next comes the rule for differentiating sums and products (the Leibniz rule). Next we have a rule for differentiating powers. In fact, our rule for differentiating powers is based on an easy to prove formula known as “logarithmic differentiation”:

$$\frac{df(x)}{dx} = f(x) \frac{d \log(|f(x)|)}{dx}$$

(the formula is valid even without the absolute value if we accept the existence of logarithms of negative numbers, which we shall not go into here). The last rule is the chain rule.

Note that Mathematica<sup>®</sup> cannot “deduce” the rule for differentiating integer powers from the rule for differentiating products, like we do in ordinary mathematics, because Mathematica<sup>®</sup>'s rules apply to the `FullForm` of the expressions, and the `FullForm` of, for example,  $x^3$  is

```
In[.]:= FullForm[x^3]
Out[.]//FullForm= Power[x, 3]
```

and not `Times[x, x, x]`, which is the form of  $x \times x$  before it gets evaluated:

```
In[.]:= FullForm[Hold[x x x]]
Out[.]//FullForm= Hold[Times[x, x, x]]
```

The reader can check that the derivative  $d$  we have just defined gives the same answers as the built-in function  $D$ , e. g.,

**In[.]:** =  $d[\text{Sin}[x] \text{Log}[x \text{Cos}[x^2]], x]$  // Simplify

**Out[.]:** =  $\text{Cos}[x] \text{Log}[x \text{Cos}[x^2]] + \text{Sin}[x] \left( \frac{1}{x} - 2x \text{Tan}[x^2] \right)$

**In[.]:** =  $D[\text{Sin}[x] \text{Log}[x \text{Cos}[x^2]], x]$  // Simplify

**Out[.]:** =  $\text{Cos}[x] \text{Log}[x \text{Cos}[x^2]] + \text{Sin}[x] \left( \frac{1}{x} - 2x \text{Tan}[x^2] \right)$

**In[.]:** =  $d[\text{Sin}[x]/x, x]$

**Out[.]:** =  $\frac{\text{Cos}[x]}{x} - \frac{\text{Sin}[x]}{x^2}$

**In[.]:** =  $D[\text{Sin}[x]/x, x]$

**Out[.]:** =  $\frac{\text{Cos}[x]}{x} - \frac{\text{Sin}[x]}{x^2}$

### 5.3 Main properties of differentiable functions

Let  $f : I \rightarrow \mathbb{R}$ , where  $I$  is an interval, and  $a \in I$ . We say that  $f$  has a *local maximum* (respectively *local minimum*) at  $a$  if there is some  $\delta > 0$  such that for all  $x \in I$  with  $|x - a| < \delta$  we have  $f(x) \leq f(a)$  (respectively  $f(x) \geq f(a)$ ). If strict inequalities hold we say that  $f$  has a *strict local maximum* (respectively *strict local minimum*) at  $a$ .

The word *extremum* refers to both maxima and minima. A global extremum is always a local one but, of course, the converse is not true. In fact, a differentiable function on a non-closed interval need not attain its global extrema (see the graph in Section 5.3.1).

The following simple fact is extremely useful.

**Theorem 14** (Fermat's Interior Extremum Theorem). *Let  $I$  be an open interval,  $a \in I$  and let  $f : I \rightarrow \mathbb{R}$  be differentiable at  $a$ . If  $f$  has a local extremum at  $a$ , then  $f'(a) = 0$ .*

A point  $a$  such that  $f'(a) = 0$  is called a *critical point* of  $f$  and  $f(a)$  is called a *critical value*.

Suppose now that  $I$  is a closed interval and  $f$  is differentiable. Then by the Weierstrass theorem (see Theorem 11 in Chapter 4) the function  $f$  has to attain its supremum and infimum (i. e., it has to have a maximum and a minimum value). Thus the maxima must occur either at critical points or at the endpoints of the interval. In such cases we can find the global maximum and minimum by solving the equation  $f'(x) = 0$ , i. e., by finding the critical points and then comparing the critical values of  $f$  with its values at the endpoints. In general a differentiable function need not have a finite number of critical points. However, complex analytic functions (and in particular real ones) have a finite number of them, which follows from what is known as the Identity Theorem.

If the number of critical points is finite, then we can find the global maximum and minimum by an algorithm (provided we have an algorithm for solving equations).

Another important theorem that we need is *Mean Value Theorem* [14, Theorem 4.17].

**Theorem 15 (Mean Value Theorem).** *If  $f : [a, b] \rightarrow \mathbb{R}$ , where  $a < b$ , is differentiable on  $(a, b)$  and is continuous on  $[a, b]$ , then there exists at least one point  $c \in [a, b]$  such that*

$$\frac{f(b) - f(a)}{b - a} = f'(c).$$

This immediately implies that if  $f'(c) > 0$  on an interval  $I$ , then  $f$  is strictly increasing on  $I$  and if  $f'(c) < 0$ , then  $f$  is strictly decreasing. The same holds with  $>$  ( $<$ ) replaced by  $\geq$  ( $\leq$ ) and the word “strictly” removed. However, if  $f$  is strictly increasing on  $I$ , we can only conclude that  $f'(x) \geq 0$  on  $I$  (consider for example the function  $f(x) = x^3$  on  $[0, 1]$ , which is strictly increasing, but  $f'(0) = 0$ ).

Recall that a *Lipschitz function*  $f : I = [a, b] \rightarrow \mathbb{R}$  was defined as one for which there exists a constant  $M$  such that  $|f(x) - f(y)| \leq M|x - y|$  for all  $x, y \in I$ . A Lipschitz function is always continuous and even uniformly continuous but not necessarily differentiable. Suppose, however, that a function  $f$  is differentiable in  $(a, b)$ . Then the Lipschitz condition is equivalent to the derivative of the function being bounded on  $I$ . Indeed, suppose that  $f$  is Lipschitz with constant  $M$ . Let  $c \in (a, b)$ . Then for every  $x \in I \setminus \{c\}$

$$\frac{|f(x) - f(c)|}{|x - c|} \leq M.$$

Taking limits we obtain  $f'(c) \leq M$ , hence the derivative of  $f$  is bounded on  $I$ . Conversely, suppose that for all  $x \in (a, b)$  we have  $|f'(x)| \leq C$ . Let  $x, y \in (a, b)$ . Then by the Mean Value Theorem we have  $(f(x) - f(y))/(x - y) = f'(\xi)$  for some  $\xi \in (x, y) \subset (a, b)$ . Hence  $|f(x) - f(y)|/|x - y| = |f'(\xi)| \leq C$  and  $f$  is Lipschitz with constant  $C$ . This makes it much easier to check if a differentiable function is Lipschitz. For example, consider the function  $f(x) = \sin(\log(x))$  for  $x \geq 1$ . Since the function is differentiable, we need only to compute its derivative to show that it is Lipschitz:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{D}[\text{Sin}[\text{Log}[x]], x] \\ \mathbf{Out}[\cdot] &:= \frac{\text{Cos}[\text{Log}[x]]}{x} \end{aligned}$$

Clearly, on  $[1, \infty)$  we have  $|\cos(\log(x))|/|x| \leq 1$ , hence the function is Lipschitz.

Recall (see Section 4.6 in Chapter 4) that a continuous function  $f : I \rightarrow \mathbb{R}$  defined on an interval has an inverse function (defined on its image) if and only if it is strictly monotone. The latter condition can be checked by computing the derivative, since we know that if  $f'(x) > 0$  on  $I$ , then  $f$  is strictly monotone. If  $f'(x) \geq 0$  on  $I$ , then  $f$  need not be strictly monotone ( $f$  can be constant on a non-trivial subinterval), and, hence,  $f$  may not have an inverse function. However, if in addition  $f$  is analytic and

not constant, then  $f'(x) \geq 0$  implies that  $f$  has an inverse function (for example, the inverse function to  $f(x) = x^3$  is  $g(x) = x^{1/3}$  on  $\mathbb{R}$ ).

```
In[.]:= InverseFunction[#^3 &]
...InverseFunction: Inverse functions are being used. Values
may be lost for multivalued inverses.
```

```
Out[.]:= #11/3 &
```

### 5.3.1 Example: global and local extrema

Next we consider a case that will demonstrate Mathematica<sup>®</sup>'s ability to solve problems that are very hard or impossible to solve by hand.

Let us find the maximum and minimum values of the function  $f(x) = \sin(\cos(5x)) - e^{(x-1)^2}$  on the interval  $[0, 2]$ . We can plot the function to see the approximate answers:

```
In[.]:= f[x_] := Sin[Cos[5 x]] - E^(x - 1)^2
In[.]:= Plot[f[x], {x, 0, 2}]
```

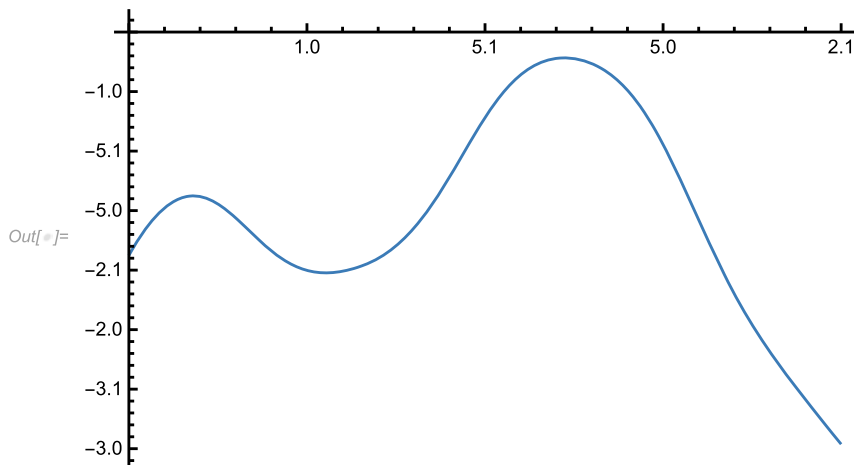


Figure 5.8

Mathematica<sup>®</sup>'s functions `Maximize` and `Minimize` can tell us the global maximum and minimum. Let us first compute the minimum:

```
In[.]:= Minimize[{f[x], 0 <= x <= 2}, x]
Out[.]:= {-E + Sin[Cos[10]], {x -> 2}}
```

The first element is the minimum value, the second one is a list  $\{x \rightarrow a\}$ , where  $a$  is a point where the minimum is attained. Note that Mathematica<sup>®</sup> returns only one

such point, even though the function may attain minima at other points too. For example,

```
In[.]:= Minimize[(x - 1)^2 (x - 2)^2, x]
Out[.]:= {0, {x -> 1}}
```

but the function is also zero at point 2.

Returning to our example, we see that the minimum value is attained at the end of the interval, where  $x = 2$  (as we can also see from the graph), and is equal to

```
In[.]:= f[2]
Out[.]:= -E + Sin[Cos[10]]
```

The answer given by `Maximize` will be more complicated, since `Mathematica`<sup>®</sup> needs to express the root in an exact form. There is no general standard form of expressing roots of such equations so `Mathematica`<sup>®</sup> does it using “root objects” (see Chapter 1):

```
In[.]:= Maximize[{Sin[Cos[5 x]] - E^(x - 1)^2, 0 <= x <= 2}, x]
Out[.]:= {-E(-1+Root[{-2 E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E(-1+#1)2 #1 &, 1.22259830515136658892}]2)
+ Sin[Cos[5 Root[
{-2 E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E(-1+#1)2 #1 &,
1.22259830515136658892}]]],
{x -> Root[{-2 E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E(-1+#1)2 #1 &,
1.22259830515136658892}]}}
```

```
In[.]:= N[%]
Out[.]:= {-0.217221, {x -> 1.2226}}
```

The critical points of  $f$  can be found by solving the equation  $f'(x) = 0$  in the interval  $0 < x < 2$ . This can be done either by using `Solve` or `Reduce`:

```
In[.]:= crits = x /. Solve[f'[x] == 0 && 0 < x < 2, x, Reals]
Out[.]:= {Root[{-2 E-1+#12 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E-1+#12
#1 &, 0.180932451667257231810}],
Root[{-2 E-1+#12 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E-1+#12
#1 &, 0.55358202938476197032}],
Root[{-2 E-1+#12 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2 E-1+#12
#1 &, 1.22259830515136658892}]}
```

```
In[.]:= N[crits]
Out[.]:= {0.180932, 0.553582, 1.2226}
```

We see that there are three critical points. We can now compute the values of  $f$  at the critical points and at the endpoints. These points are our candidates for the maximum



and the minimum, so we make a list of them:

```
In[.]:= candidates = (Join[{0}, crits, {2}]);
In[.]:= f /@ candidates // N
Out[.]:= {-1.87681, -1.37658, -2.02275, -0.217221, -3.4623}
```

We see that the maximum is attained at the fourth point and the minimum is at the last (the right endpoint 2).

Let us now consider the question of determining the nature of the critical points (that is, whether they are local maxima, local minima or neither). For this we will study the sign of the derivative of  $f$  in the interval  $(0, 2)$ . Clearly, if a function is increasing (decreasing) to the left of a critical point and decreasing (increasing) to the right, the point is a local maximum (minimum). So we can determine the nature of the critical points by finding where the function is increasing and where it is decreasing. We can use Reduce to solve the inequality

```
In[.]:= Reduce[f'[x] > 0 && 0 < x < 2, x]
Out[.]:= 0 < x < Root{
  [-2E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2E(-1+#1)2 #1 &,
  0.180932451667257231810]}||
  Root{[-2E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2E(-1+#1)2 #1 &,
  0.55358202938476197032]} < x <
  Root{[-2E(-1+#1)2 + 5 Cos[Cos[5 #1]] Sin[5 #1] + 2E(-1+#1)2 #1 &,
  1.22259830515136658892]}
```

and we can immediately determine the nature of the critical points. However, there are also more elementary methods to demonstrate this. The first is to observe that the derivative

```
In[.]:= f'[x]
Out[.]:= -2E(-1+x)2(-1 + x) - 5Cos[Cos[5 x]] Sin[5 x]
```

is a continuous function. This implies (by the Darboux property) that the sign of the derivative can change only after passing through a critical point. This means that to determine the intervals of monotonicity we only need to check the signs of the derivative at points lying between the points (for example at midpoints). Below the function Partition divides our list of “candidates” into sublists of length 2 with offset 1. Applying the function Mean gives us a list of the midpoints and then by applying the function Sign we find the sign of the derivative at them:

```
In[.]:= Sign[f'[#]] & /@ (Mean /@ (Partition[candidates, 2, 1]))
Out[.]:= {1, -1, 1, -1}
```

From this we see that the first critical point is a local maximum, the second a local minimum and the third again a local maximum.

Note that we made the assumption that  $f'$  is continuous. Is this always true? No, as we will show in Section 5.3.2. But it turns out that continuity of the derivative is not required since the derivative of any function always has the Darboux property, even when it is not continuous.<sup>5</sup>

The second approach to determine the nature of critical points makes use of the second order derivative. In the next chapter we will show (using Taylor's formula) that if the second order derivative at a critical point is positive, then the point is a local minimum and if it is negative, a local maximum. Let us check this in our case:

```
In[.]:= Sign[f''[#]] & /@ crits
Out[.]:= {-1, 1, -1}
```

We could try to compute the global maxima and minima on the whole  $\mathbb{R}$ . We first compute the limits as  $x \rightarrow \infty(-\infty)$ , which are both  $-\infty$ . This tells us that the function is not bounded from below. It also follows that there exists some interval  $[-r, r]$  such that the maximum on this interval is equal to the maximum on  $\mathbb{R}$  (in other words, the function is bounded above and its supremum is attained on  $\mathbb{R}$ ). However, it is not in general possible to determine the value of  $r$  which is sufficient for this purpose. Moreover, as  $r$  increases finding the maximum value takes longer and longer.

### 5.3.2 Example

In the following example we will show that it is possible to have a function with a local minimum such that there is no interval to the left where the function is decreasing and no interval to the right where it is increasing. Moreover, we will show that the derivative of this function is not continuous. Also we will be able to see that the Darboux property for the derivative holds.

We will define a function which depends on a parameter  $c \geq 1$ . We could define it as a function of two variables,  $x$  and  $c$ , but at this point we prefer to think of it as a function of one variable with a parameter.

```
In[.]:= g[c_][x_] := Piecewise[{{(Sin[1/x] + c)*x^2,
  x != 0}, {c*x^2, x == 0}}]

In[.]:= Manipulate[Plot[g[c][x], {x, -r, r}, PlotRange ->
  All], {{c, 1, "c"}, 1, 2, Appearance -> "Labeled"},
  {{r, 1, "r"}, 0.01, 1, Appearance -> "Labeled"},
  SaveDefinitions -> True]
```

<sup>5</sup> [https://en.wikipedia.org/wiki/Darboux%27s\\_theorem\\_\(analysis\)](https://en.wikipedia.org/wiki/Darboux%27s_theorem_(analysis)).

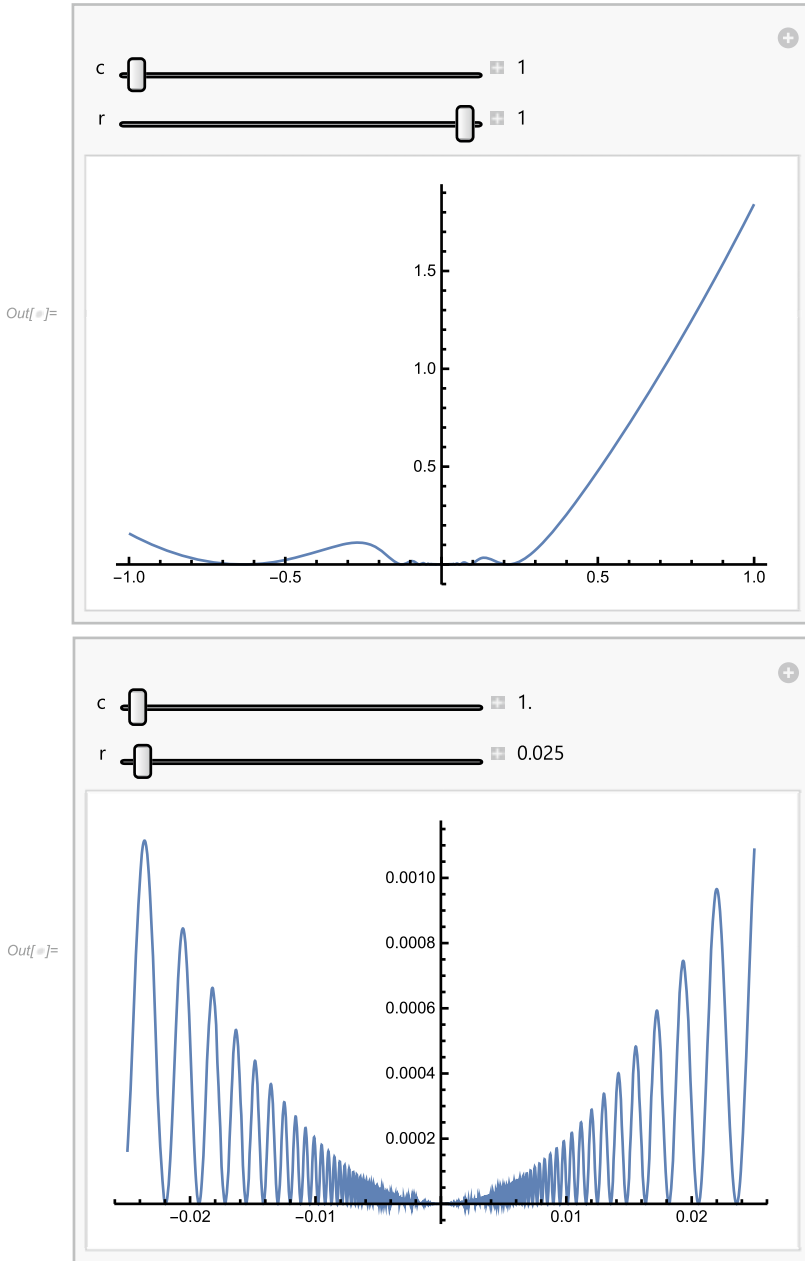


Figure 5.9

The function  $g[c]$  has a local minimum at 0 (since  $|\sin(1/x)| \leq 1$  and  $c \geq 1$ ). For  $c > 1$  the local minimum at 0 is strict. For  $c = 1$  there are points arbitrarily close to 0 in which the function has the value 0.

Let us show that for every  $r > 0$  the restrictions of  $g[c]$  to  $[0, r]$  and  $[-r, 0]$  are not monotone. The derivative of  $g[c]$  is given by

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{FullSimplify}[g[c]'[x]] \\ \mathbf{Out}[\cdot] &:= \begin{cases} -\text{Cos}\left[\frac{1}{x}\right] + 2x \left(c + \text{Sin}\left[\frac{1}{x}\right]\right) & x \neq 0 \\ 0 & \text{True} \end{cases} \end{aligned}$$

Let us consider points  $x_n = 1/(2\pi n)$ , where  $n$  is an integer:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Simplify}[g[c]'[1/(2 \text{Pi } n)], \text{Assumptions} \rightarrow \\ &\quad \text{Element}[n, \text{Integers}]] \\ \mathbf{Out}[\cdot] &:= -1 + \frac{c}{n\pi} \end{aligned}$$

For large enough  $n$  these values are negative, while the points  $x_n$  are positive and as small as we like. Now consider the points  $y_n = 1/(\pi(2n + 1))$ , where  $n$  is an integer.

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Simplify}[g[c]'[1/(\text{Pi } (2n + 1))], \text{Assumptions} \rightarrow \\ &\quad \text{Element}[n, \text{Integers}]] \\ \mathbf{Out}[\cdot] &:= 1 + \frac{2c}{\pi + 2n\pi} \end{aligned}$$

This time by choosing large enough  $n$  we can make  $y_n$  arbitrarily close to 0 while  $2c/(2\pi n + \pi) + 1 > 0$ . The sequences  $-x_n$  and  $-y_n$  have analogous properties on the negative axis.

Let us now show that  $g[c]$  is differentiable at 0 but its derivative is not continuous there. To see that the function  $g[c]$  is differentiable, we consider the difference quotient defined earlier in this chapter:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Q}[0, g[c]][x] \\ \mathbf{Out}[\cdot] &:= \begin{cases} x \left(c + \text{Sin}\left[\frac{1}{x}\right]\right) & x \neq 0 \\ 0 & \text{True} \end{cases} \end{aligned}$$

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Limit}[\%, x \rightarrow 0] \\ \mathbf{Out}[\cdot] &:= 0 \end{aligned}$$

Hence the derivative of  $g[c]$  at 0 is defined and is 0. In order for the derivative to be continuous its limit as  $x \rightarrow 0$  has to exist and has to be equal to the value of the derivative at 0. But the limit does not exist.

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Limit}[-\text{Cos}[1/x] + 2*x*(c + \text{Sin}[1/x]), x \rightarrow 0] \\ \mathbf{Out}[\cdot] &:= \text{Interval}[\{-1, 1\}] \end{aligned}$$

Mathematica® 11 somewhat surprisingly returns an interval rather than Indeterminate, although the limit of  $\text{Cos}[1/x]$  is Indeterminate and the limit of the second term

is zero. In any case the limit does not exist and the derivative is not continuous at zero. However, it is clear from the graph (for  $c = 2$ ) that the derivative has the Darboux property:

```
In[.]:=Plot[x*(2 + Sin[1/x]) - Cos[1/x], {x, -1, 1},
  Exclusions -> {x == 0}]
```

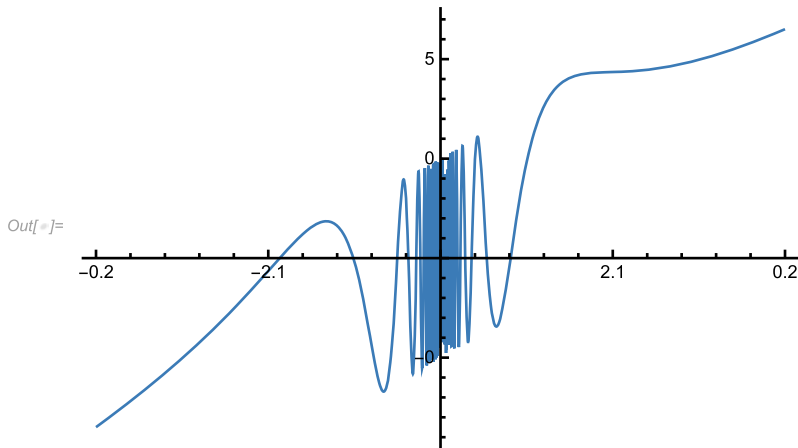


Figure 5.10

### 5.3.3 Example: the inverse function of the hyperbolic sine

Let us consider the following example. We will consider the hyperbolic trigonometric functions, which in Mathematica<sup>®</sup> are defined as Sinh and Cosh. Since in this example we do not want to use Mathematica<sup>®</sup>'s built-in knowledge about these functions we will define our own versions:

```
In[.]:=sinh[x_] := (Exp[x] - Exp[-x])/2
```

```
In[.]:=cosh[x_] := (Exp[x] + Exp[-x])/2
```

Unfortunately in this example the function InverseFunction does not work correctly in Mathematica<sup>®</sup> 11.3:

```
In[.]:=InverseFunction[sinh][x]
```

```
...InverseFunction: Inverse functions are being used. Values
may be lost for multivalued inverses.
```

```
Out[.]:=Log[x - Sqrt[1 + x^2]]
```

As we will see below the function has two inverses over the complex numbers and Mathematica<sup>®</sup> chooses the one which is incorrect over the real numbers. We can clearly see that this function does not take real values. It is a pity that the current

version of Mathematica<sup>®</sup> does not allow to specify the domain over which InverseFunction should be considered. We will derive the correct answer below.

We will now show that the function  $\sinh$  is invertible and compute the derivatives of its inverse. Our first approach is simply to find the inverse function. To show that the function  $\sinh$  is invertible we need to show that the equation  $\sinh(x) = y$  has at most one solution. In fact we will show that it has exactly one solution for every real  $y$ :

```
In[.]:= x /. Solve[y == sinh[x], x, Reals]
Out[.]:= {Log[y + Sqrt[1 + y^2]]}
```

This equation is easy to solve by hand (just use the substitution  $e^x = t$  and solve the resulting quadratic equation). This means that the inverse function of  $\sinh$  is  $x \mapsto \log(x + \sqrt{x^2 + 1})$ . We can now find its derivative directly:

```
In[.]:= Simplify[D[Log[Sqrt[x^2 + 1] + x], x]]
Out[.]:=  $\frac{1}{\sqrt{1+x^2}}$ 
```

The weakness of this approach is that it requires explicitly solving the equation  $f(x) = y$  for  $x$ . In general this can be very hard to do or even impossible in an explicit form. However, we do not need to solve the equation to prove the existence of an inverse or even to find a formula for it. Since the function is defined on an interval, we only need to show that it is strictly monotone:

```
In[.]:= D[sinh[x], x]
Out[.]:=  $\frac{1}{2}(E^{-x} + E^x)$ 
```

Note also that the image of  $\sinh$  is the whole of  $\mathbb{R}$ , since

```
In[.]:= FunctionRange[sinh[x], x, y]
Out[.]:= True
```

The answer `True` after applying the function `FunctionRange` means that every real number belongs to the range. Since the derivative is positive everywhere, the function  $\sinh$  has an inverse function defined on  $\mathbb{R}$ . We can find the derivative using the formula

```
In[.]:= Derivative[1][InverseFunction[f]][x]
Out[.]:=  $\frac{1}{f'[f^{(-1)}[x]]}$ 
```

Since the derivative of  $\sinh$  is clearly  $\cosh$ , this can be written as

$$\frac{1}{\cosh(\sinh^{(-1)}(x))}.$$

This is indeed a solution of the problem, but as it is not given in a convenient form, we now make use of the (easy to prove) identity  $\cosh(x)^2 - \sinh(x)^2 = 1$  and obtain

$$\frac{1}{\cosh(\sinh^{(-1)}(x))} = \frac{1}{\sqrt{\sinh^2(\sinh^{-1}(x)) + 1}} = \frac{1}{\sqrt{1 + x^2}}.$$

Since  $\cosh$  takes only positive values we take the positive square root. We can now check that this is exactly the answer we get by differentiating Mathematica<sup>®</sup>'s built-in function `ArcSinh`:

```
In[ ]:= D[ArcSinh[x], x]
Out[ ]:=  $\frac{1}{\sqrt{1+x^2}}$ 
```

## 5.4 Convex functions

**Definition 4.** A function  $f : I \rightarrow \mathbb{R}$  is said to be *convex* if for every  $\lambda \in (0, 1)$  and every  $x, y \in I$  we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

If the above holds with  $\leq$  replaced by  $\geq$ , then the function  $f$  is called *concave*.

Geometrically this corresponds to the fact, as illustrated below, that the blue point on the secant connecting two points with coordinates  $(x, f(x))$  and  $(y, f(y))$  lies above the red point on the graph of the function  $f$ :

```
In[ ]:= Manipulate[Module[{f = (#1 - 1)^2 + 5 & },
  Show[Plot[f[x], {x, -1, 3}, AxesOrigin -> {0, 0}],
  Graphics[{Line[{{p[[1]], 0}, {p[[1]], f[p[[1]]}}],
  Line[{{q[[1]], 0}, {q[[1]], f[q[[1]]}}],
  Line[{{t*p[[1]] + (1 - t)*q[[1]], 0},
  {t*p[[1]] + (1 - t)*q[[1]], f[t*p[[1]] +
  (1 - t)*q[[1]]}], Line[{{p[[1]], f[p[[1]]},
  {q[[1]], f[q[[1]]}}], Red, PointSize[0.02],
  Point[{t*p[[1]] + (1 - t)*q[[1]],
  f[t*p[[1]] + (1 - t)*q[[1]]}], Blue,
  Point[{t*p[[1]] + (1 - t)*q[[1]], t*f[p[[1]] +
  (1 - t)*f[q[[1]]}]]], {{p, {0, 0}}, {-1, 0},
  {1.9, 0}, Locator}, {{q, {2, 0}}, {2, 0}, {3, 0},
  Locator}, {{t, 0.5, "t"}, 0, 1}, SaveDefinitions
-> True]
```

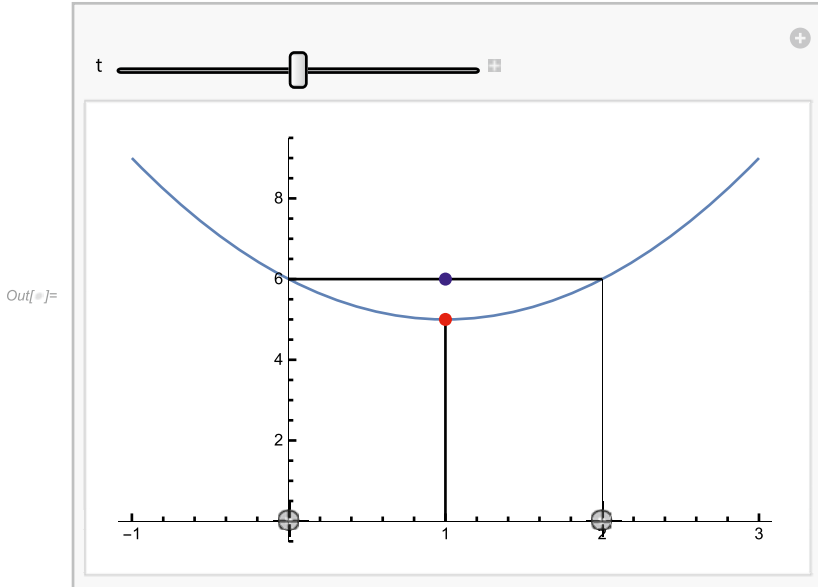


Figure 5.11

There is an equivalent formulation of convexity, which also has a geometric interpretation. Consider the following difference quotient function (of two variables):

$$\tilde{Q}(f)(x, y) = \frac{f(x) - f(y)}{x - y},$$

where  $x, y \in I$ . If we fix one of the variables, say  $y$ , we obtain a function of one variable (whose domain is  $I \setminus \{y\}$ ). One can show that  $f$  is convex if and only if the difference quotient function is increasing when viewed as a function of one variable.

The following interactive illustration shows that the slope of any secant of the graph of a convex function is an increasing function:

```
In[ ]:= Manipulate[Module[{f = (#1 - 1)^2 + 5 & ,
  g, P, Q}, g = Plot[f[x], {x, -1, 5}];
  P = {p[[1]], f[p[[1]]]}; Q = {q[[1]], f[q[[1]]]};
  Show[g, Graphics[{Text["P", P], Text["Q", Q],
  Red, Line[{P, Q}]}], Axes -> True, Frame ->
  False, PlotRange -> All, AxesOrigin -> {0, 0},
  Ticks -> None]], {{p, {0, 0}}, {-1, 0}, {1.9, 0},
  Locator}, {{q, {2, 0}}, {2, 0}, {5, 0}, Locator},
  SaveDefinitions -> True]
```



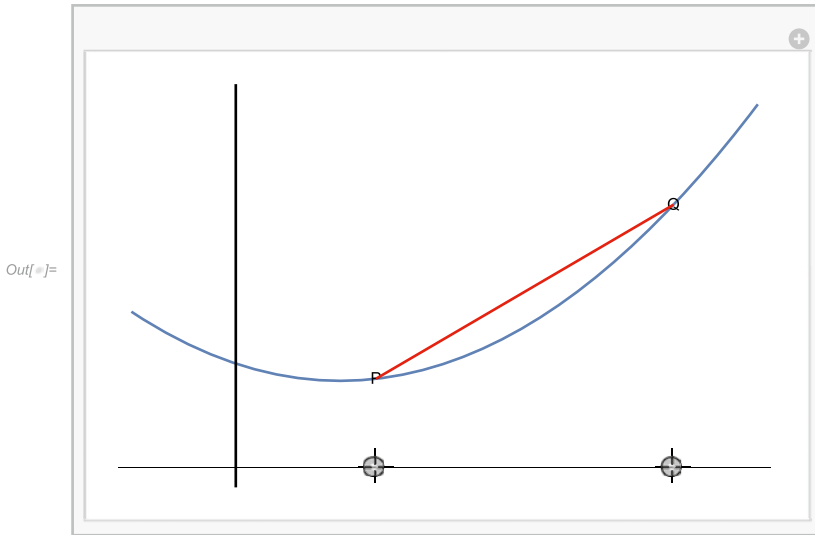


Figure 5.12

A convex function needs not be differentiable or even continuous. For example it is easy to check that the following function is convex on  $[-1, 1]$ :

```
In[ ]:= gg[x_] := Piecewise[{{2, x == -1},
  {Abs[x], -1 < x < 0}, {x^2, Inequality[0, LessEqual,
  x, Less, 1]}, {2, x == 1}}]
```

```
In[ ]:= Plot[gg[x], {x, -1, 1}, Prolog ->
  {Point[{{-1, 3/2}, {1, 3/2}}]}, PlotRange ->
  {{-1, 1.1}, {0, 2}}]
```

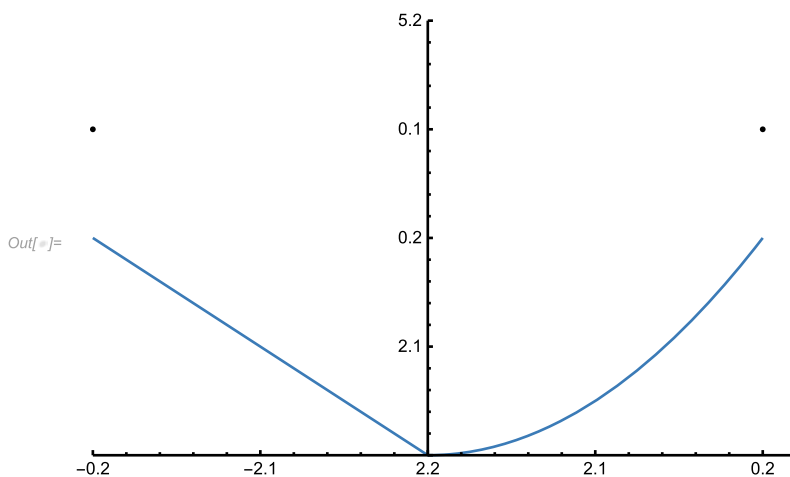


Figure 5.13

This function is not differentiable at 0 (but is both left and right differentiable) and it is not continuous at the endpoints of the interval.

Using the fact that the difference quotient function of a convex function is increasing one can show that a convex function on an interval has both left and right derivatives at every interior point of the interval (because as we move the left point of the secant keeping the right point fixed, the slope of the secant gives us an increasing and bounded function, hence, the left derivative exists; the analogous argument holds for the right derivative). These two one-sided derivatives need not be equal but their existence implies that the function is continuous at all points in the interior of the interval (left [right] differentiability implies left [right] continuity which together imply continuity).

If  $f$  is differentiable (in the interior of its domain), then one can characterize its convexity in terms of the first order derivative. A geometric characterization is that the tangent to the graph of a convex function (at points over the interior of the domain) lies below the graph. Naturally, the tangents to the graph of a concave function lie above it. The function shown below is convex on certain subintervals of its domain (the convex part of the function is shown in red) and concave at others. The points where the behavior changes are endpoints of the intervals of convexity and concavity and the tangents at such point lie above the graph on one side and below on the other.

```
In[.]:= Manipulate[Module[{f = Sin[6*#1^3] & , u, v},
  u = Minimize[{f[x], -1 <= x <= 1}, x][[1]];
  v = Maximize[{f[x], -1 <= x <= 1}, x][[1]];
  Show[Plot[f[x], {x, -1, 1}, ColorFunction ->
  Function[x, If[Derivative[2][f][x] > 0, Red, Blue]],
  ColorFunctionScaling -> False, Epilog ->
  {PointSize[0.02], Point[{a, 0}]}, PlotRange ->
  {{-1, 1}, (4/3)*{u, v}}, Plot[f[a] +
  Derivative[1][f][a]*(x - a), {x, -1, 1},
  PlotRange -> {{-1, 1}, (4/3)*{u, v}}]],
  {{a, 0, "a"}, -1, 1, Appearance -> "Labeled"},
  SaveDefinitions -> True]
```

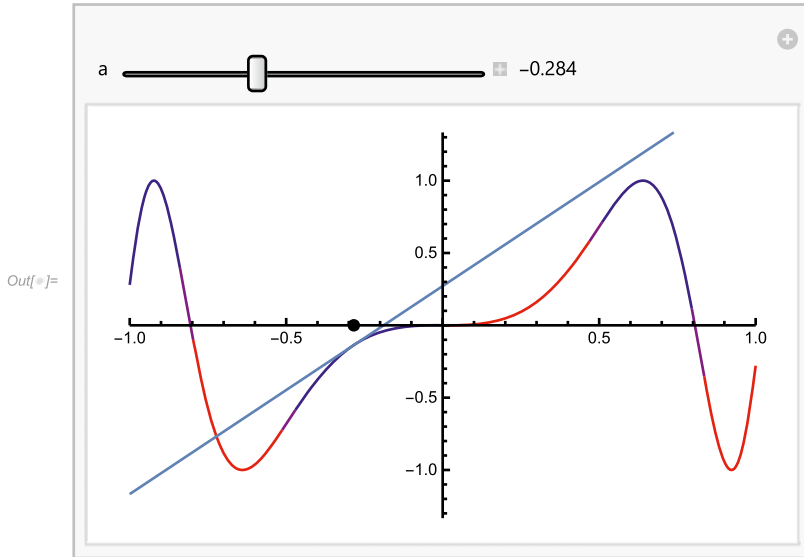


Figure 5.14

Another characterization of convexity of a differentiable function is that its derivative is increasing (the slopes of tangents are increasing). The derivative of a function is not necessarily differentiable itself, but when it is, the condition that the derivative is increasing can be replaced by the condition that the second order derivative is non-negative (positive for strict convexity). This is generally the most convenient condition and was used in the graph above to color the convex and concave parts of the graph red and blue (by means of the `ColorFunction` option of `Plot`). The option `ColorFunctionScaling` is used to stop Mathematica<sup>®</sup> from scaling the argument in `ColorFunction` to lie between 0 and 1.

### 5.4.1 Jensen's inequality

One of the most common applications of convexity is in proving inequalities. The first definition of convexity above (Definition 4) leads by an easy inductive argument to one of the most useful inequalities, *Jensen's inequality*.

**Theorem 16 (Jensen's inequality).** *Let  $f : I \rightarrow \mathbb{R}$  be a convex function,  $x_1, x_2, \dots, x_n \in I$  and let  $t_1, t_2, \dots, t_n \in [0, 1]$  be such that  $\sum_{i=1}^n t_i = 1$ . Then*

$$f\left(\sum_{i=1}^n t_i x_i\right) \leq \sum_{i=1}^n t_i f(x_i).$$

**5.4.1.1 Example**

Let us consider the following inequality:

$$1 + \sqrt[3]{e^{2a}} \sqrt[3]{e^b} \leq \sqrt[3]{(1 + e^a)^2} \sqrt[3]{1 + e^b}.$$

To prove it we first rewrite it in the form

$$1 + e^{2a/3} e^{b/3} \leq (1 + e^a)^{2/3} (1 + e^b)^{1/3}.$$

Let us take the natural logarithm of both sides (we can do it, since both sides are positive and the function log is increasing):

$$\log(1 + e^{2a/3} e^{b/3}) \leq \frac{2}{3} \log(1 + e^a) + \frac{1}{3} \log(1 + e^b).$$

Since  $2/3 + 1/3 = 1$  and the function  $x \mapsto \log(e^x + 1)$  has the second order derivative,

**In[.]**:= D[Log[1 + E^x], {x, 2}] // Simplify

**Out[.]**:=  $\frac{E^x}{(1 + E^x)^2}$

which is positive, the function is convex, and we can use Jensen's inequality:

**In[.]**:= F[t1 x1 + t2 x2] <= t1 F[x1] + t2 F[x2]/.

{x1 -> a, x2 -> b, t1 -> 2/3, t2 -> 1/3,

F -> (Log[1 + E^#] &)}

**Out[.]**:=  $\text{Log}\left[1 + E^{\frac{2a}{3} + \frac{b}{3}}\right] \leq \frac{2}{3} \text{Log}[1 + E^a] + \frac{1}{3} \text{Log}[1 + E^b]$

The right hand side is  $\log((e^a + 1)^{2/3} (e^b + 1)^{1/3})$ , hence, applying exp to both sides gives the desired inequality.

## 6 Sequences and series of functions

In this chapter we will consider the problem of approximating functions by polynomials and representing functions by power series. We will discuss the Taylor series and Mathematica<sup>®</sup>'s function Series. Next we consider uniform and almost uniform convergence of function sequences and series and conditions for continuity and differentiability of their limits and sums.

### 6.1 Power series continued

We will now return to the subject of power series, which we already introduced in Section 3.10.

A *power series* can be viewed as a generalization of a polynomial. Recall that polynomials are just lists of numbers  $(a_0, \dots, a_d)$  together with rules for adding and multiplying any two such lists. It is convenient to write polynomials in the form  $a_0 + a_1x + a_2x^2 + \dots + a_nx^n$ , where  $x$  is called an indeterminate or a variable. Formal power series are defined in exactly the same way, except that we consider infinite rather than just finite sequences. Such a series can be written in the form  $a_0 + a_1x + a_2x^2 + \dots + a_nx^n + \dots$ , where  $x$  is again a variable. Two such series can also be added and multiplied (the multiplication being given by the Cauchy product).

One important difference between polynomials and formal power series is that a polynomial always defines a function given by substituting numbers for the variable  $x$ . However, in the case of formal power series the situation is more complicated for although we can “substitute” a number for  $x$ , the number series thus obtained may not be convergent (it is always convergent when we substitute 0). To obtain a function we need to find the set of points where the series is convergent.

Also recall from Section 3.10 that we consider power series of the form  $\sum_{n=0}^{\infty} a_n(x - x_0)^n$ , where  $x_0$  is called the center of the series. It is not hard to prove that a power series defines a function that is continuous in the interior of its region of convergence. A result by Abel known as *Abel's continuity/limit theorem* [14, Theorem 9.51] states that the function is also left or right continuous at the endpoints of the region of convergence (provided the region of convergence contains those endpoints).

Any polynomial can be rewritten as a polynomial with any given center. Let us demonstrate how this can be done with Mathematica<sup>®</sup>. Consider the polynomial

```
In[.] := poly[x_] := x^3 + 3*x^2 - 2*x + 5
```

We want to rewrite it as a polynomial with center at  $x_0 = 1$ :

```
In[.] := poly1[x] = Expand[poly[x] /. x -> y + 1] /. y -> x - 1  
Out[.] := 7 + 7(-1 + x) + 6(-1 + x)^2 + (-1 + x)^3
```

<https://doi.org/10.1515/9783110590142-006>

```
In[.]:= Simplify[poly[x] == poly1[x]]
Out[.]:= True
```

Later on in this chapter we will see how to use the Taylor expansion for the same purpose. For power series the situation is more complicated, since every series with center  $x_0$  is convergent at  $x = x_0$ , but a series may not be convergent at every point (hence it will not be possible to rewrite it with the center at a point which lies outside of its region of convergence).

### 6.1.1 Example

A powerful feature of Mathematica<sup>®</sup> is its ability to express sums of certain power series in terms of built-in analytic functions. Consider the function

$$f(x) = \sum_{n=1}^{\infty} \frac{(x-1)^n}{\sqrt{n+1}}.$$

Let us first find the region of convergence of the power series and then sketch the graph of the function which is defined by this power series. The region of convergence of the power series is  $[0, 2)$ :

```
In[.]:= SumConvergence[(x - 1)^n/Sqrt[n + 1], n]
Out[.]:= Abs[-1 + x] < 1 || x == 0
```

We can plot the region of convergence as follows:

```
In[.]:= reg = Graphics[{Disk[{0, 0}, 0.04], Thickness[0.01],
  Line[{{0, 0}, {1.95, 0}}], Red, Disk[{1, 0}, 0.04],
  Black, Circle[{2, 0}, 0.04]}]
```


**Out[.]**:= 

Figure 6.1

Mathematica<sup>®</sup> can find an exact formula for this series in terms of the special function PolyLog:

```
In[.]:= f[x_] = Sum[(x - 1)^n/Sqrt[n + 1], {n, 0, Infinity}]
Out[.]:=  $\frac{\text{PolyLog}[\frac{1}{2}, -1 + x]}{-1 + x}$ 
```

Let us plot the function over the region of convergence:

```
In[.]:= Show[{Plot[f[x], {x, -2, 2}, AxesOrigin ->
  {0, 0}], reg}]
```

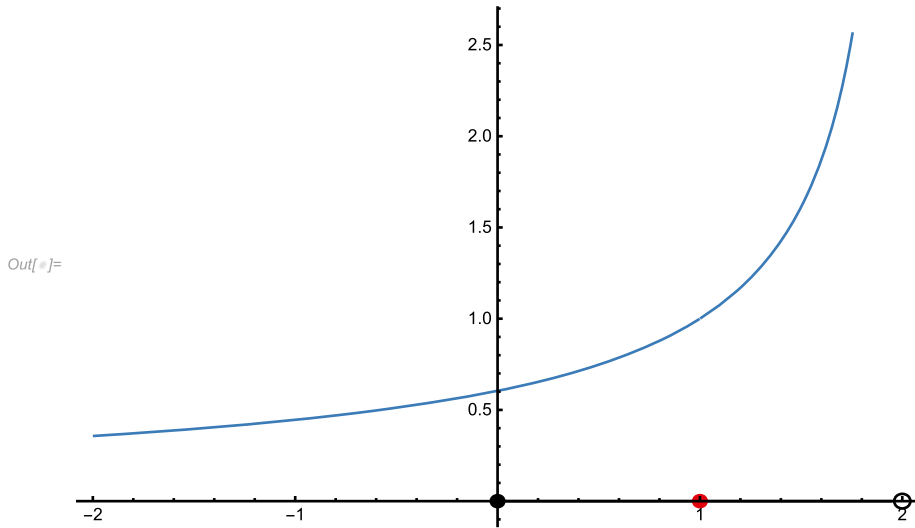


Figure 6.2

Note that the function itself is defined and differentiable on the entire negative real axis, although the series is not convergent there. It is actually also defined on the positive axis, but it assumes complex values there, e. g.,

```
In[.]:= N[ComplexExpand[f[3]]]
Out[.]:= -0.805031 - 1.06447 I
```

### 6.1.2 Example

Let us consider the series  $\sum_{n=0}^{\infty} ((-1)^n + 2)^n x^n$ . This is an example of the series that Mathematica<sup>®</sup> cannot deal with in this form and needs human help. Note that applying SumConvergence does not work:

```
In[.]:= SumConvergence[(2 + (-1)^n)^n x^n, n]
Out[.]:= SumConvergence[(2 + (-1)^n)^n x^n, n]
```

Not surprisingly, Mathematica<sup>®</sup> cannot also find the sum:

```
In[.]:= Sum[(-1)^n + 2)^n x^n, {n, 0, Infinity}]
Out[.]:=  $\sum_{n=0}^{\infty} (2 + (-1)^n)^n x^n$ 
```

Noting that the coefficient of  $x^n$  is 1 for  $n$  odd and  $3^n$  for  $n$  even, we can write this series as the sum of two series  $\sum_{n=0}^{\infty} x^{2n+1}$  and  $\sum_{n=0}^{\infty} 3^{2n} x^{2n}$  that are easy to deal with.

The region of convergence of the original series will then be the intersection of the regions of convergence of these two series.

```
In[.]:= SumConvergence[3^(2*n)*x^(2*n), n]
Out[.]:= Abs[x] < 1/3 || 3x == 1 || x == -1/3
```

We have to be careful here, since this answer in Mathematica® 11.3 is clearly wrong, as the series is not convergent when  $x = 1/3$ .

```
In[.]:= SumConvergence[x^(2*n + 1), n]
Out[.]:= Abs[x] < 1
```

Hence the region of convergence is  $[-1/3, 1/3)$  and the sum is

```
In[.]:= f[x_] = Sum[3^(2*n)*x^(2*n), {n, 0, Infinity}] +
Sum[x^(2*n + 1), {n, 0, Infinity}]
Out[.]:= 1/(1 - 9x^2) + x/(1 - x^2)
```

The graph of the function over its region of convergence is

```
In[.]:= Plot[f[x], {x, -3^(-1), 1/3}]
```

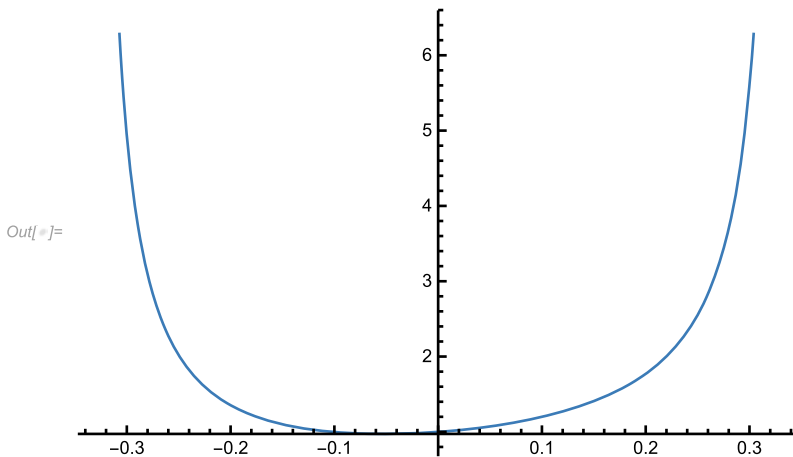


Figure 6.3

Note that again that the function found by Mathematica® is defined on a much larger region:

```
In[.]:= Plot[f[x], {x, -1, 1}, Exclusions -> {-3^(-1), 1/3}]
```



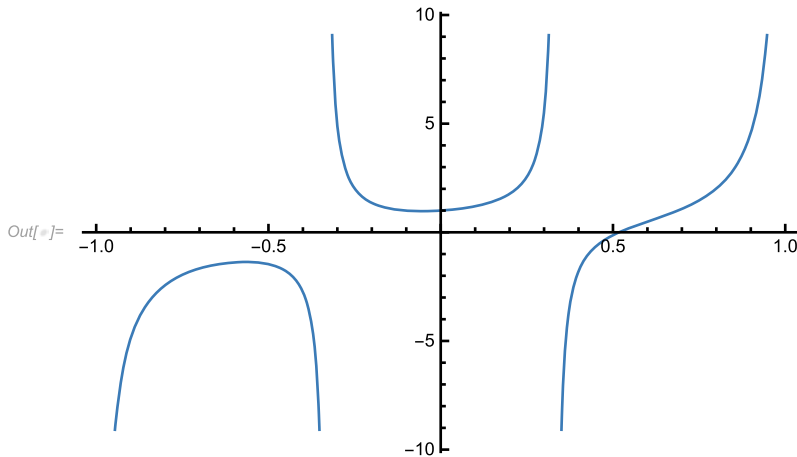


Figure 6.4

As we will see in Section 6.2, although the series that we were given was only defined in the interval  $[-1/3, 1/3]$ , the function  $f$  can also be expressed as a power series outside of this interval, but the series will be a different one with a different center.

Note also that we could find the radius of convergence of the original series by using the *Cauchy–Hadamard Theorem*, which says that the radius of convergence  $R$  of a power series  $\sum_{n=0}^{\infty} a_n(x - x_0)^n$  is given by

$$\frac{1}{R} = \limsup_{n \rightarrow \infty} (|a_n|^{1/n}).$$

In Mathematica<sup>®</sup> 11.3  $\limsup$  is computed by the function `MaxLimit`:

```
In[ ]:= MaxLimit[Abs[2 + (-1)^n], n -> Infinity]
Out[ ]:= 3
```

Thus the radius of convergence is  $R = 1/3$ .

## 6.2 Taylor polynomials and Taylor series

Let  $f : D \rightarrow \mathbb{R}$  be a function which is  $n$  times differentiable at  $x_0 \in D$ . We define the  $n$ -th *Taylor polynomial* of  $f$  with center at  $x_0$  by  $T_n(f) : D \rightarrow \mathbb{R}$ , where

$$T_n(f)(x) = \sum_{k=0}^n \frac{f^{(k)}(x_0)(x - x_0)^k}{k!}.$$

We view  $T_n(f)$  as a degree  $n$  polynomial approximation to  $f$  near  $x_0$ . Of course the Taylor polynomial depends on both  $n$  and  $x_0$  and sometimes it is denoted by  $T_n(f, x_0)$ , but for simplicity of notation we will omit the center of expansion.

If the function  $f$  is smooth (differentiable infinitely many times), then one can also form its *Taylor series*  $\sum_{k=0}^{\infty} f^{(k)}(x_0)(x - x_0)^k/k!$ , but it need not converge everywhere where the original function is infinitely differentiable. It may even happen that the Taylor series diverges everywhere except the center [3, Section 24].

The first Taylor polynomial is

$$T_1(f)(x) = f(x_0) + f'(x_0)(x - x_0).$$

Since  $f$  is differentiable, we have

$$f(x) - T_1(f)(x) = \left( \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0) \right) (x - x_0).$$

Let

$$r(x) = \frac{f(x) - f(x_0)}{x - x_0} - f'(x_0).$$

By the definition of derivative this can be extended to a function defined on  $D$  such that  $\lim_{x \rightarrow x_0} r(x) = 0$ . Hence we have  $f(x) = T_1(f)(x) + R(x)$ , where  $R(x) = (x - x_0)r(x)$  is some function on  $D$  with the property that

$$\lim_{x \rightarrow x_0} \frac{R(x)}{x - x_0} = 0.$$

This is generalized to an arbitrary function of class  $C^n$  in the following theorem.

**Theorem 17** (The Peano Remainder Terms Theorem [1, Section 7.9]). *If  $f : D \rightarrow \mathbb{R}$  is  $n$  times differentiable at  $x_0 \in D$  and  $D$  contains an open interval  $I$  such that  $x_0 \in I$  (in such a situation we say that  $D$  is a neighborhood of  $x_0$ ), then*

$$f(x) = T_n(f)(x) + R_n(x),$$

where

$$\lim_{x \rightarrow x_0} \frac{R_n(x)}{(x - x_0)^n} = 0.$$

We can also write  $R_n(x) = (x - x_0)^n r_n(x)$ , where  $r_n$  has the property  $\lim_{x \rightarrow x_0} r_n(x) = 0$ .

Any function  $h$  that has the property  $\lim_{x \rightarrow x_0} h(x)/g(x) = 0$  is called an  $o(g(x))$  function. Hence the remainder  $R_n$  is an  $o((x - x_0)^n)$  function. Note that there are a number of useful functions related to asymptotics implemented in **Mathematica**<sup>® 6</sup>.

We can easily show that a representation of  $f$  in the form of the sum of a polynomial of degree  $n$  and an  $o((x - x_0)^n)$  function is unique (we will refer to this representation as *the Peano representation of the function*).

We can use the Peano Remainder Terms Theorem to prove an important result about local maxima and minima (which we mentioned in Chapter 5).

<sup>6</sup> <https://reference.wolfram.com/language/guide/Asymptotics.html>

**Theorem 18.** Let  $f : (a, b) \rightarrow \mathbb{R}$  be  $n$  times differentiable at  $c \in (a, b)$ , where  $n \geq 2$ . Assume that  $f^{(k)}(c) = 0$  for  $k = 1, 2, \dots, n-1$  and  $\alpha = f^{(n)}(c) \neq 0$ . We have

- (i) if  $n$  is even and  $\alpha > 0$  ( $\alpha < 0$ ), then  $f$  has a strict local minimum (maximum) at  $c$ ;
- (ii) if  $n$  is odd, then  $f$  does not have a local extremum at  $c$ .

In particular, if  $c$  is a critical point of  $f$ , i. e.,  $f'(c) = 0$ , and if  $f''(c) > 0$  ( $f''(c) < 0$ ), then  $f$  has a local minimum (maximum) at  $c$ .

The proof follows immediately from the Peano Remainder Terms Theorem. We write

$$f(x) - f(c) = \frac{\alpha(x-c)^n}{n!} + (x-c)^n r_n(x), \quad \frac{f(x) - f(c)}{(x-c)^n} = \frac{\alpha}{n!} + r_n(x).$$

If  $n$  is even, then the term  $(x-c)^n$  is always positive. Since  $r_n(x) \rightarrow 0$  as  $x \rightarrow c$ , if  $\alpha > 0$ , then  $f(x) > f(c)$  in some neighborhood of  $c$ . Hence, there is a strict minimum at  $c$ . Similarly, if  $\alpha < 0$ , then  $f(x) < f(c)$  in some neighborhood of  $c$ , hence  $c$  is a local maximum. If  $n$  is odd, then  $f(x) - f(c)$  changes sign when  $x$  passes through  $c$ , hence there is neither a local maximum nor a local minimum at  $c$ .

Given a function  $f$  and a point  $a$ , we can find the following expansion consisting of the Taylor polynomial of degree  $n$  with center at  $a$  and the remainder in Mathematica<sup>®</sup> as follows:

**In[.]**:= Clear[f]

**In[.]**:= Series[f[x], {x, a, 2}]

**Out[.]**:= f[a] + f'[a](x - a) +  $\frac{1}{2}$  f''[a](x - a)<sup>2</sup> + 0[x - a]<sup>3</sup>

Note that the remainder is denoted in Mathematica<sup>®</sup> by  $0[x - a]^{n+1}$ , where  $n = 2$  in our case. Mathematica<sup>®</sup>'s 0 notation does not quite correspond to the mathematical standard "little  $o$ " notation. It essentially means that the remainder is of the form  $(x - a)^3 h(x)$ , where  $h(x)$  is some analytic function (that is, one that itself is defined by a power series near  $a$ ).

There is a convenient short form for Series:

**In[.]**:= f[x] + 0[x, a]^3

**Out[.]**:= f[a] + f'[a](x - a) +  $\frac{1}{2}$  f''[a](x - a)<sup>2</sup> + 0[x - a]<sup>3</sup>

When  $a$  is 0, we can omit it:

**In[.]**:= f[x] + 0[x]^3

**Out[.]**:= f[0] + f'[0]x +  $\frac{1}{2}$  f''[0]x<sup>2</sup> + 0[x]<sup>3</sup>

Note that Mathematica<sup>®</sup> by default assumes that the symbolic functions are analytic and can be expanded as power series:

**In[.]**:= Series[g[x]\*Cos[x], {x, 0, 2}]

**Out[.]**:= g[0] + g'[0]x +  $\frac{1}{2}$  (-g[0] + g''[0])x<sup>2</sup> + 0[x]<sup>3</sup>

It is also possible to tell Mathematica<sup>®</sup> that the symbolic function is not analytic:

```
In[.] := Series[g[x]*Cos[x], {x, 0, 2}, Analytic -> False]
Out[.] := g[x]  $\left(1 - \frac{x^2}{2} + O[x]^3\right)$ 
```

but we cannot tell Mathematica<sup>®</sup> that  $g$  is of class  $C^n$ .

Taylor series with center at 0 are known as *Maclaurin series*. For the function  $\sin$  we have:

```
In[.] := Sin[x] + O[x]^6
Out[.] := x -  $\frac{x^3}{6} + \frac{x^5}{120} + O[x]^6$ 
```

In this case we can obtain the general coefficient of the Taylor series:

```
In[.] := SeriesCoefficient[Sin[x], {x, 0, n}]
Out[.] :=  $\begin{cases} \frac{I^n (-1 + (-1)^n)}{2^n n!} & n \geq 0 \\ 0 & \text{True} \end{cases}$ 
```

Note that although the coefficients are real numbers, Mathematica<sup>®</sup> uses the complex  $i$  to obtain a general formula.

The function `Series` returns a special kind of object and its Head is `SeriesData`:

```
In[.] := FullForm[f[x] + O[x]^2]
Out[.] // FullForm = SeriesData[x, 0, List[f[0],
Derivative[1][f][0]], 0, 2, 1]
```

`SeriesData` returned by the function `Series` does not represent an infinite series but only a formula essentially equivalent to the Peano representation of a function. Such an object always contains information about a finite number of series coefficients. For example, if

```
In[.] := ss = Sum[x^n/n!, {n, 0, 4}] + O[x]^5
Out[.] := 1 + x +  $\frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + O[x]^5$ 
```

then

```
In[.] := SeriesCoefficient[ss, {x, 0, 4}]
Out[.] :=  $\frac{1}{24}$ 
```

However,

```
In[.] := SeriesCoefficient[ss, {x, 0, 5}]
Out[.] := SeriesCoefficient  $\left[1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + O[x]^5, \{x, 0, 5\}\right]$ 
```

To obtain the  $n$ -th Taylor polynomial we use the function `Normal` (this is a multi-purpose function used in different context which returns what is called “the normal form” of various expressions):

**In[.]**:= Normal[Series[f[x], {x, a, 2}]]

**Out[.]**:=  $f[a] + (-a + x)f'[a] + \frac{1}{2}(-a + x)^2 f''[a]$

As we mentioned earlier, to extract a coefficient in a series expansion we need to use the function SeriesCoefficient:

**In[.]**:= SeriesCoefficient[f[x] + 0[x]^4, {x, 0, 3}]

**Out[.]**:=  $\frac{1}{6} f^{(3)}[0]$

This is the same answer we can get by asking for a suitable coefficient of the Taylor polynomial:

**In[.]**:= Coefficient[Normal[f[x] + 0[x]^4], x^3]

**Out[.]**:=  $\frac{1}{6} f^{(3)}[0]$

SeriesCoefficient works both on power series (objects with Head SeriesData) and on analytic functions. It will always compute the coefficient of the  $n$ -th term for a numerical  $n$ , but in many cases it will not be able to return a general formula for a symbolic  $n$ :

**In[.]**:= SeriesCoefficient[Sin[Cos[x] - 1], {x, 0, n}]

**Out[.]**:= SeriesCoefficient[-Sin[1 - Cos[x]], {x, 0, n}]

However, for  $n = 20$  we have

**In[.]**:= SeriesCoefficient[Sin[Cos[x] - 1], {x, 0, 20}]

**Out[.]**:=  $-\frac{60657859289}{2432902008176640000}$

Mathematica<sup>®</sup> tries to express a convergent series in terms of built-in analytic functions, for example,

**In[.]**:= Sum[x^n/(2\*n + 1)!, {n, 0, Infinity}]

**Out[.]**:=  $\frac{\text{Sinh}[\sqrt{x}]}{\sqrt{x}}$

Only in the case of such kind of expressions the function SeriesCoefficient will return a coefficient of an arbitrary degree (and in some cases also a symbolic one). If Mathematica<sup>®</sup> cannot express a series in terms of known analytic functions, it is unable to return the coefficients, even when they are obvious, for example,

**In[.]**:= SeriesCoefficient[Sum[x^n/(n^2)!, {n, 0, Infinity}], {x, 0, 10}]

**Out[.]**:= SeriesCoefficient $\left[\sum_{n=0}^{\infty} \frac{x^n}{n^2!}, \{x, 0, 10\}\right]$

In such cases the Peano representation gives the desired result:

**In[.]**:= SeriesCoefficient[Sum[x^n/(n^2)!, {n, 0, 4}] + 0[x]^5, {x, 0, 4}]

**Out[.]**:=  $\frac{1}{20922789888000}$

Now, suppose  $f$  itself is a polynomial, e. g.,

$$\mathbf{In}[\cdot] := f[x_] := 1 - 2x + 3x^2 + x^3$$

What will be its Taylor polynomials with center at 0? The answer follows immediately from the uniqueness of the Peano representation mentioned above:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Table}[\text{Normal}[\text{Evaluate}[f[x]] + 0[x]^n], \{n, 1, 4\}] \\ \mathbf{Out}[\cdot] &:= \{1, 1 - 2x, 1 - 2x + 3x^2, 1 - 2x + 3x^2 + x^3\} \end{aligned}$$

The third and higher Taylor polynomials are equal to the polynomial  $f(x)$  itself. The lower ones are just the truncations of the polynomial  $f(x)$  after the corresponding degree. Now let us consider the Taylor polynomials of  $f$  with centers at points other than 0, e. g., at 1. Again by the uniqueness theorem, the third polynomial will be just the same polynomial but rearranged so that the center is at 1:

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Normal}[f[x] + 0[x, 1]^4] \\ \mathbf{Out}[\cdot] &:= 3 + 7(-1 + x) + 6(-1 + x)^2 + (-1 + x)^3 \end{aligned}$$

$$\begin{aligned} \mathbf{In}[\cdot] &:= \text{Expand}[\%] \\ \mathbf{Out}[\cdot] &:= 1 - 2x + 3x^2 + x^3 \end{aligned}$$

Note that this also gives us another method of rewriting a polynomial as a polynomial centered around any given number  $a$ : we simply use the Taylor polynomial of the same degree centered at  $a$ .

As we know, series can be added and can also be multiplied by means of the Cauchy product. We can also perform these operations on their Peano representations. For example

$$\begin{aligned} \mathbf{In}[\cdot] &:= g1 = \text{Cos}[x] + 0[x]^10; g2 = \text{Cos}[x] + 0[x]^3; \\ \mathbf{In}[\cdot] &:= g1 * g2 \\ \mathbf{Out}[\cdot] &:= 1 - x^2 + 0[x]^3 \end{aligned}$$

For any series which begins with a non-zero constant term we can find its multiplicative inverse by writing either

$$\begin{aligned} \mathbf{In}[\cdot] &:= 1 / (\text{Cos}[x] + 0[x]^10) \\ \mathbf{Out}[\cdot] &:= 1 + \frac{x^2}{2} + 0[x]^4 \end{aligned}$$

or

$$\begin{aligned} \mathbf{In}[\cdot] &:= 1 / \text{Cos}[x] + 0[x]^4 \\ \mathbf{Out}[\cdot] &:= 1 + \frac{x^2}{2} + 0[x]^4 \end{aligned}$$

Functions which are not analytic at some point can sometimes be expanded in a series centered at that point but this series will not in general be a power series. For example, we can expand

$$\mathbf{In[.]} := 1/\text{Sin}[x] + 0[x]^3$$

$$\mathbf{Out[.]} := \frac{1}{x} + \frac{x}{6} + 0[x]^3$$

This kind of series is called a *Laurent series* and it plays an important role in complex analysis but it does not have the properties of power series that will be important for us.

Series (or, more precisely, their Peano representations) can also be composed, just like functions, but only if the center of the first series is the value of the second at its own center. Thus, since  $\cos(0) = 1$  we can evaluate

$$\mathbf{In[.]} := \text{ComposeSeries}[\text{Sin}[x] + 0[x, 1]^4, \text{Cos}[x] + 0[x]^3]$$

$$\mathbf{Out[.]} := \text{Sin}[1] - \frac{1}{2} \text{Cos}[1] x^2 + 0[x]^3$$

However, the following composition

$$\mathbf{In[.]} := \text{ComposeSeries}[\text{Sin}[x] + 0[x]^10, \text{Cos}[x] + 0[x]^10]$$

is not defined.

The function `InverseSeries` corresponds to the function `InverseFunction`; in other words, it gives a series for the inverse of the function represented by the original series:

$$\mathbf{In[.]} := \text{InverseSeries}[\text{Series}[\text{Sin}[x], \{x, 0, 3\}]]$$

$$\mathbf{Out[.]} := x + \frac{x^3}{6} + 0[x]^4$$

In general the function `InverseSeries` may not return a power series but a more general type of series (the *Puiseux series*):

$$\mathbf{In[.]} := \text{InverseSeries}[1 + x^2 + 0[x]^10]$$

$$\mathbf{Out[.]} := \sqrt{x-1} + 0[x-1]^{9/2}$$

Note that in `Mathematica`<sup>®</sup> 11.3 there is a bug which causes many examples with `InverseSeries` not to be properly evaluated, for instance expressions `InverseSeries[1 - x^2 + 0[x]^10]` or `InverseSeries[Cos[x] + 0[x]^10]`, which work fine in `Mathematica`<sup>®</sup> 10.4.

The main practical application of these operations on series is that they allow us to find the Peano representation of functions, which are compositions of functions whose Peano representations (or Taylor series) are known, without the need to compute derivatives.

### 6.2.1 Example

Let us find the limit

$$\lim_{x \rightarrow \infty} \left( x - x^2 \log \left( \frac{1}{x} + 1 \right) \right).$$

This problem can be easily solved by using the l'Hospital rule, but we will show how to use the Peano Remainder Terms Theorem. First we convert the problem to one where

the limit is taken as the variable goes to 0 (in order to avoid series expansions at infinity). So let us take  $y = 1/x$ . The problem reduces to finding

$$\lim_{y \rightarrow 0} \frac{y - \log(y + 1)}{y^2}.$$

Since

$$\mathbf{In[.]} := \text{Log}[1 + y] + O[y]^3$$

$$\mathbf{Out[.]} := y - \frac{y^2}{2} + O[y]^3$$

we have  $\log(1 + y) = y - y^2/2 + r(y)y^2$ , where  $\lim_{y \rightarrow 0} r(y) = 0$ . Hence,

$$\lim_{y \rightarrow 0} \frac{y - \log(y + 1)}{y^2} = \lim_{y \rightarrow 0} \frac{y^2/2 - r(y)y^2}{y^2} = \lim_{y \rightarrow 0} \left( \frac{1}{2} - r(y) \right) = \frac{1}{2}.$$

Instead of using the substitution  $y = 1/x$  we could have used the Taylor expansion at infinity:

$$\mathbf{In[.]} := x - x^2 * \text{Log}[1/x + 1] + O[x, \text{Infinity}]$$

$$\mathbf{Out[.]} := \frac{1}{2} + O\left[\frac{1}{x}\right]^1$$

This tells us that the function  $f$  can be written in the form  $1/2 + \tilde{r}(x)$ , where  $\lim_{x \rightarrow \infty} \tilde{r}(x) = 0$ . Hence we get the same answer as above.

### 6.2.2 Example

Let us study for which  $a > 0$  the series

$$\sum_{n=1}^{\infty} \left( \frac{1}{\sqrt{n}} - \tan^{-1}\left(\frac{1}{\sqrt{n}}\right) \right)^a$$

is convergent.

The function `SumConvergence` cannot deal with this problem in `Mathematica`® 10.4. However in version 11.3 something surprising happens. The first evaluation of the following expression returns the original input but a subsequent evaluation returns the correct solution (provided we do not quit the Kernel in between):

$$\mathbf{In[.]} := \text{SumConvergence}[(1/\text{Sqrt}[n] - \text{ArcTan}[1/\text{Sqrt}[n]])^a,$$

$$n, \text{Assumptions} \rightarrow \{a > 0\}]$$

$$\mathbf{Out[.]} := 3 a > 2$$

This is probably due to some internal time constraint on the evaluation and the fact that `Mathematica`® often saves intermediate computation results and reuses them in subsequent evaluations. This would explain why on the second evaluation `Mathematica`® has sufficient time to arrive at the solution. We have not been able to test whether



performing this computation on a very fast computer would return the answer on the first attempt.

Let us now show how to deal with this problem with the help of the Taylor expansion. First, note that we are dealing with a sum of positive terms:

**In[.]**:= Reduce[ArcTan[x] < x, x]

**Out[.]**:= x > 0

So now let us consider the following Taylor expansion with center at 0 of the function ArcTan:

**In[.]**:= ArcTan[x] + O[x]^5

**Out[.]**:= x -  $\frac{x^3}{3}$  + O[x]^5

Hence if  $f(n) = 1/\sqrt{n} - \tan^{-1}(1/\sqrt{n})$ , then  $f(n) = 1/(3n^{3/2}) + R_3(n)$ , where  $\lim_{n \rightarrow \infty} R_3(n)/(1/n^{3/2}) = 0$ . Hence using the limit comparison test we see that  $f(n)$  is similar to  $1/n^{3/2}$  and  $f(n)^a$  to  $1/n^{3a/2}$ . Hence the series converges if and only if  $a > 2/3$ .

### 6.2.3 Approximating functions by Taylor polynomials

The Peano Remainder Terms Theorem can be viewed as the “local” version of Taylor’s theorem, which tells us that the Taylor polynomial of  $f$  with center  $x_0$  approximates the function  $f$  better and better the closer we are to  $x_0$  (see [11, Chapter 7]). However, this approximation may be good only at  $x_0$ . Let us consider the function

**In[.]**:= h[x\_] := Piecewise[{{0, x == 0}}, Exp[-(x^2)^(-1)]]

**In[.]**:= Plot[h[x], {x, -1, 1}]

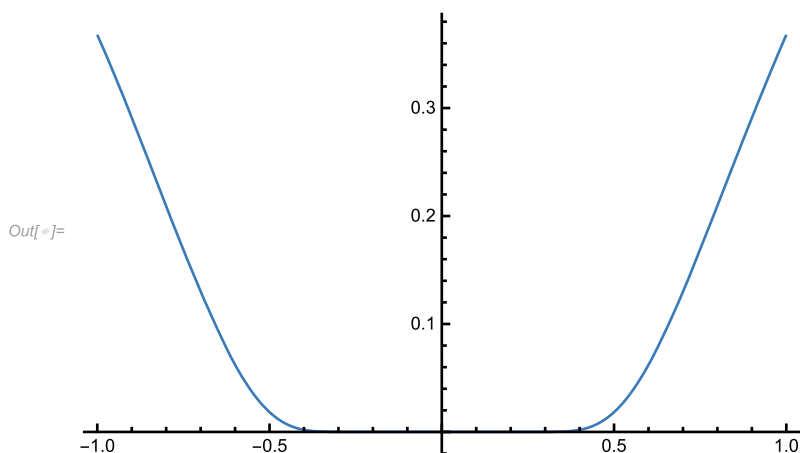


Figure 6.5

This function is differentiable everywhere (including 0) infinitely many times. One can easily show that all of its derivatives at 0 are zero, e. g.,

**In[.]**:= Derivative[20][h][0]

**Out[.]**:= 0

Moreover,

**In[.]**:= Limit[h[x]/x^n, x -> 0, Assumptions -> n >= 0]

**Out[.]**:= 0

Thus the Taylor formula for  $h$  takes the trivial form  $h(x) = 0 + h(x)$ , where 0 is the  $n$ -th Taylor polynomial at zero for every  $n$  and  $h$  itself is the remainder. In this case the Taylor polynomials with center at zero give us no information about the behavior of the function except at zero. Note that  $h$  is an example of an infinitely differentiable function whose Taylor series does not converge to it.

The Peano Remainder Terms Theorem is purely local: it does not help us at all to measure how well a Taylor polynomial approximates the function  $f$  in some neighborhood of  $x$ . Still, it can be very useful for solving certain problems.

To obtain global information about the approximation of a function by its Taylor polynomials we need a different form for the remainder. There are many of them but the best-known one is Lagrange's form. It requires stronger assumptions on the function  $f$ .

**Theorem 19** (The Lagrange Remainder Terms Theorem [14, Theorem 8.44]). *Let  $f$  be an  $(n + 1)$  times differentiable function in the interval  $(a, b)$  and suppose that the  $n$ -th derivative of  $f$  is continuous at  $a$  and  $b$ . Then there exists some  $c \in (a, b)$  such that*

$$f(x) = T_n(f)(x) + \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1}.$$

Note that for  $n = 0$  this gives the Mean Value Theorem. In general we know nothing about the value of  $c$  except that it lies in the interval  $(a, b)$ . However, this information is often sufficient to obtain a global estimate of the quality of the Taylor approximation. A very useful consequence is the following statement.

**Corollary 1.** *Let  $f$  be as in Theorem 19 and suppose that  $|f^{(n+1)}(x)| \leq M$  for all  $x \in (a, b)$ . Then*

$$|f(x) - T_n(f)(x)| \leq \frac{M(b-a)^{n+1}}{(n+1)!}$$

*on the whole interval  $(a, b)$ .*

Note that the right hand side tends to 0 as  $n \rightarrow \infty$ . If  $f$  is differentiable infinitely many times and if the result above holds for every positive integer  $n$ , then the Taylor polynomials are the partial sums of series that converges to  $f$  on  $(a, b)$ . For example,

if  $f(x) = \sin(x)$ , then  $f^{(n)}(x)$  is bounded on the entire real line for all  $n$  and we see that  $f$  is the sum of its Taylor series. Since  $f$  was defined as the sum of a convergent power series in Section 3.10 and we know that such a representation must be unique, the Taylor series and the power series centered at 0 which defines the function  $\sin$  must be the same. Exactly the same is true for the function  $f(x) = \exp(x)$ . Even though its derivatives are not bounded globally (as in the case of  $\sin$  or  $\cos$ ) they are still bounded by the same constant for every  $n$  on  $(a, b)$ . The constant now depends on the interval, but in any case, the remainder must tend to 0 as  $n \rightarrow \infty$  at any point on the real line. More generally, we shall see later on that if a function is defined by a convergent power series centered at  $x_0$  in an interval, then it is infinitely many times differentiable and the series must coincide with the Taylor series at  $x_0$ .

The Lagrange form of the remainder contains more information than the Peano one (although it needs stronger assumptions on  $f$ ) and can therefore be used for the same purposes and for some other purposes for which the Peano form is not suitable.

#### 6.2.4 Example: rational approximation of $\sqrt{e}$

Let us find a rational approximation to  $\sqrt{e}$  with error less than 0.001.

This is easy to do in Mathematica<sup>®</sup>. In fact, the function `Rationalize` will solve exactly our problem:

```
In[ ]:= Rationalize[Sqrt[E], 0.001]
Out[ ]:=  $\frac{61}{37}$ 

In[ ]:= Abs[% - E^(1/2)] < 0.001
Out[ ]:= True
```

Of course there are many solutions to this problem and Mathematica<sup>®</sup> chooses the one with the least possible denominator.

Let us now see how this can be done with the help of Taylor polynomials and the Lagrange form of the remainder. Using the Lagrange Remainder Terms Theorem we have

$$e^{1/2} = 1 + \frac{1}{2} + \frac{1}{2!} \left(\frac{1}{2}\right)^2 + \dots + e^{c_n} \frac{1}{(n+1)!} \left(\frac{1}{2}\right)^{n+1},$$

where  $c_n$  is some number between 0 and  $1/2$ , hence it is less than  $1/2$ . Thus the remainder is bounded above by  $2(1/2)^{n+1}/(n+1)!$  (since  $\sqrt{e} < 2$ ).

We now use the `While` loop to find the first  $n$  for which this remainder is less than 0.001:

```
In[ ]:= n = 1; While[(2*(1/2)^(n + 1))/(n + 1)! >=
0.001, n++]; n
Out[ ]:= 4
```

Hence the answer we obtain is

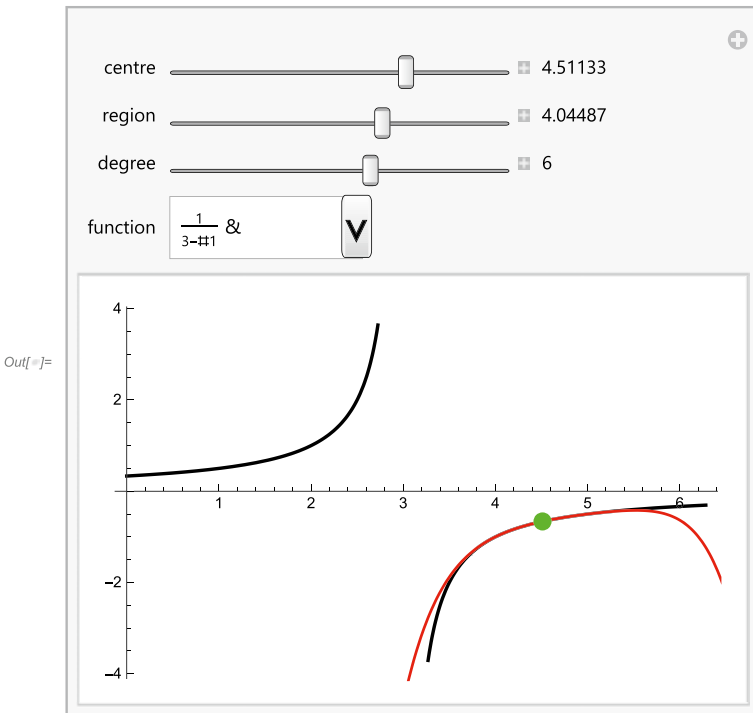
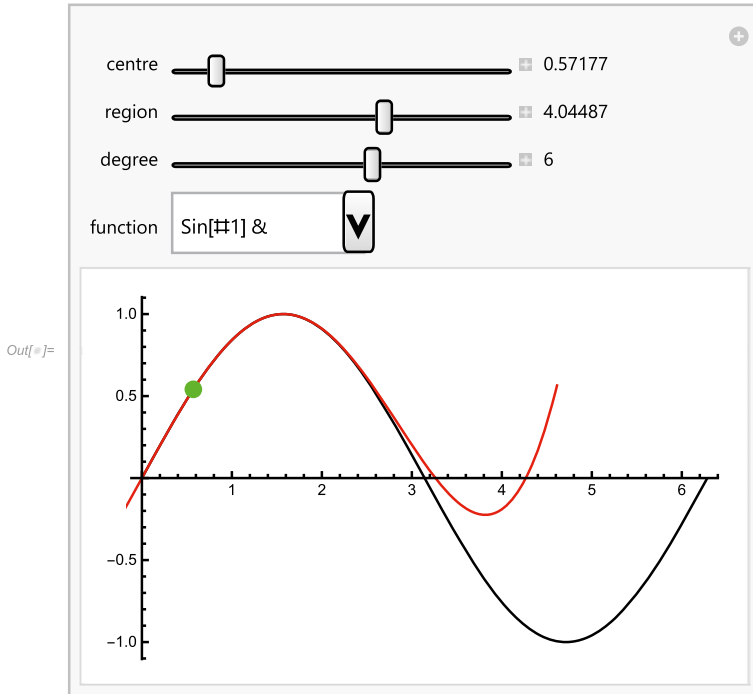
```
In[.]:= Sum[(1/2)^n/n!, {n, 0, 4}]
Out[.]:=  $\frac{211}{128}$ 

In[.]:= Abs[% - E^(1/2)] < 0.001
Out[.]:= True
```

### 6.2.5 Example: illustration of approximation of functions with Taylor polynomials

In the interactive illustration below we display the graph of a function and its Taylor polynomials of varying degrees (from 0 to 10). We can manipulate three parameters: the degree of the Taylor polynomial, its center and the region over which we consider the approximation. With the default function  $\sin$ , we observe that as we increase the degree of the polynomial, the Taylor polynomial approximates the function over a larger and larger region, but the approximation is better near the center of expansion. Our second example is the function  $x \mapsto 1/(3-x)$ , which is not defined at  $x = 3$  and has two branches. We see that to approximate the branch to the left of the point 3, we need to choose a center to the left of 3, and to approximate the branch to the right, we need to choose a center to the right of 3. The last function is  $x \mapsto \log(x+1)$ . For this function the Taylor series with center at 0 is convergent only in the interval  $(-1, 1]$  (we only show the graph over the positive axis). To approximate it elsewhere a different center has to be chosen.

```
In[.]:= T[n_, a_: 0][f_] := Normal[f[x] + O[x, a]^(n + 1)]
In[.]:= Manipulate[Show[{Plot[f[x], {x, 0, 2*Pi},
  PlotStyle -> {Thick, Black}, Exclusions ->
  If[Head[f[[1]]] === Power, {3}, None]],
  Plot[Evaluate[T[n, a][f]], {x, a - r, a + r},
  PlotStyle -> Red], Graphics[{Green, PointSize[0.03],
  Point[{a, f[a]}]}]], {{a, 0, "center"}, 0, 2*Pi,
  Appearance -> "Labeled"}, {{r, 1, "region"}, 0.1,
  2*Pi, Appearance -> "Labeled"}, {{n, 0, "degree"},
  0, 10, 1, Appearance -> "Labeled"}, {{f, Sin[#1] & ,
  "function"}, {Sin[#1] & , 1/(3 - #1) & ,
  Log[1 + #1] & }, PopupMenu], SaveDefinitions -> True]
```



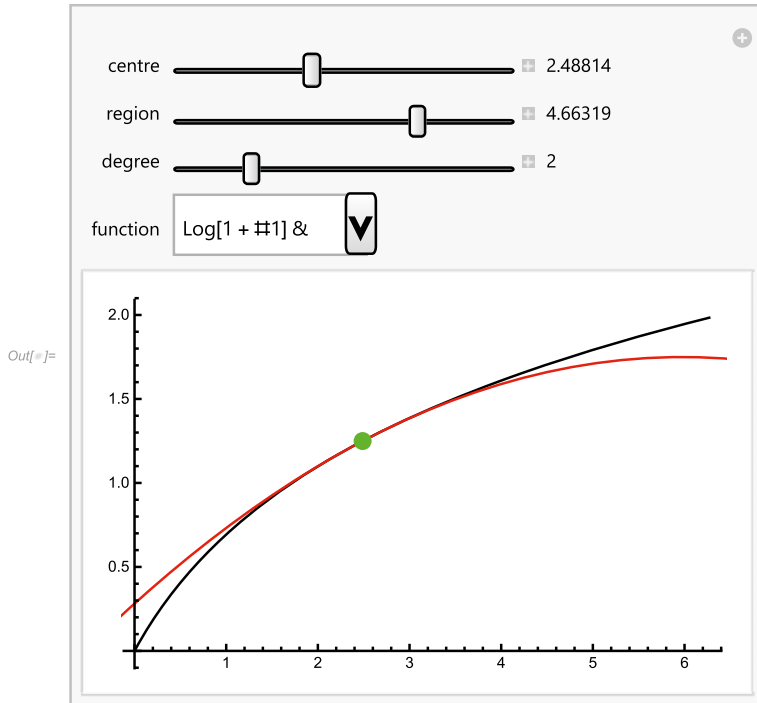


Figure 6.6

### 6.3 Convergence of sequences and series of functions

We started this book with the axioms of the real numbers, which are of course the most basic objects of real analysis. However, we have already pointed out that most (though not all) concepts that we have defined, such as sequences, series and their limits, can also be defined when the real numbers are replaced by complex numbers and most of the theorems have their complex analogues. In fact, much of what we have done so far can be done in a more abstract setting, in which sequences, series, limits, derivatives, etc., are considered in the context of Banach spaces (see [7, Chapter V]). Here we will not do this but only observe that most of the definitions of these concepts and most of the proofs for both real and complex numbers depend on the fact that both real and complex numbers are vector spaces and that they have a notion of “distance”, which is given by  $|x - y|$ , where  $|\cdot|$  denotes the absolute value for real numbers and the modulus for complex ones. We will now show that much of the theory of sequences, series and their convergence remains valid if numbers are replaced by functions  $f : X \rightarrow \mathbb{R}$ , where  $X$  will be a subset of  $\mathbb{R}$ . We first need to introduce the concept of “distance” between two functions. If  $X$  is compact and the functions are continuous, then the natural concept of distance between two such functions  $f$  and  $g$

is the maximum distance between their values at the same points in  $X$ . For example, consider the functions  $f(x) = \cos(x)$  and  $g(x) = x - x^3/3$  on the interval  $[-1, 1]$ :

```
In[ ]:= Plot[{Cos[x], x - x^3/3}, {x, -1, 1}, PlotStyle ->
  {Red, Green}, Prolog -> With[{a = -0.636},
  Line[{a, Cos[a]}, {a, a - a^3/3}]]]
```

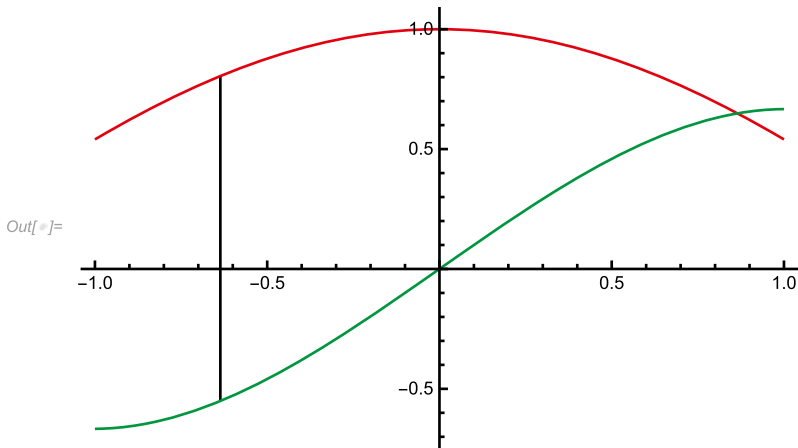


Figure 6.7

What is the distance between these functions? According to our definition it is the largest distance between two points on the red and green curves with the same  $x$ -coordinate. It can be computed as follows:

```
In[ ]:= N[Maximize[{Abs[Cos[x] - x + x^3/3], -1 <= x <= 1}, x]]
Out[ ]:= {1.35473, {x->-0.636733}}
```

Since functions on  $X$  form a vector space, we only need to define the distance between a function and the constant function with value 0. This is called the (supremum) norm of  $f$  and is denoted by  $\|f\|$ :

$$\|f\| = \sup_X |f(x)|.$$

The distance between two functions  $f$  and  $g$  is then defined by

$$\text{dist}(f, g) = \|f - g\| = \sup_X |f(x) - g(x)|.$$

Note, however, that this will only be a real number if the supremum is attained, i. e., is a maximum. This will always be the case when the functions are continuous and are defined on a compact set. Since distance should always be finite, we will not use this word except when dealing with continuous functions defined on compact sets. However, the norm of a function is defined without these restrictions. The *norm* has the following properties:

- (i)  $\|f\| \geq 0$ ;
- (ii)  $\|f\| = 0$  if and only if  $f = 0$  (the zero function);
- (iii) if  $\lambda \in \mathbb{R}$ , then  $\|\lambda f\| = |\lambda| \|f\|$ ;
- (iv)  $\|f + g\| \leq \|f\| + \|g\|$  (the triangle inequality).

Note that the triangle inequality for the norm implies the “usual” triangle inequality for the distance:

$$\text{dist}(f, g) \leq \text{dist}(f, h) + \text{dist}(h, g).$$

This is true even if we allow distance to take the value  $\infty$ .

Now let us consider a sequence  $\{f_n\}$  of real-valued functions on some interval (not necessarily compact)  $X \subset \mathbb{R}$ . We say that such a sequence is *pointwise convergent* if for each  $x \in X$  the sequence  $\{f_n(x)\}$  is convergent. In this case we can define a function  $f : X \rightarrow \mathbb{R}$  by  $f(x) = \lim_{n \rightarrow \infty} f_n(x)$  and we say that the sequence is pointwise convergent to  $f$ . As we will soon see, pointwise convergence is not a good notion of convergence for functions as it does not have useful properties. This is because although the values of the functions  $f_n$  at a given  $x$  approach the value of  $f$ , the functions  $f_n$  do not become closer to  $f$  as  $n \rightarrow \infty$ . The correct notion of convergence for functions is uniform convergence, defined below, which means that  $\text{dist}(f, f_n) \rightarrow 0$  as  $n \rightarrow \infty$ .

Both pointwise convergence and uniform convergence can be expressed in terms of quantifiers: for *pointwise convergence* we have

$$\forall \varepsilon, \varepsilon > 0 \quad \forall x, x \in X \quad \exists N, N \in \mathbb{N} \quad \forall n > N \quad |f_n(x) - f(x)| < \varepsilon$$

and for *uniform convergence* we have

$$\forall \varepsilon, \varepsilon > 0 \quad \exists N, N \in \mathbb{N} \quad \forall x, x \in X \quad \forall n > N \quad |f_n(x) - f(x)| < \varepsilon.$$

As we can see, just as in the case of ordinary and uniform continuity, the difference between pointwise convergence and uniform convergence can be expressed in terms of the difference in the order of quantifiers. The reader may be interested to know whether there is an example to show uniform convergence using quantifier elimination, just as we did in the case of uniform continuity. Unfortunately this cannot be done because any such statement would involve quantification over the integers and there is no such algorithm (quantifier elimination algorithms exist only over real numbers).

Uniform convergence implies pointwise convergence. The main use of pointwise convergence is that it provides the necessary condition for uniform convergence and in finding the function to which the given sequence could be uniformly convergent. Unfortunately, as we will soon see, uniform convergence on non-compact sets is rather rare; for example, we will see that polynomial sequences do not converge uniformly on  $\mathbb{R}$ . Fortunately, almost all the important properties of uniform convergence are satisfied for a weaker notion of convergence called “*almost uniform convergence*”. We say



that a sequence of functions  $\{f_n\}$  converges almost uniformly to  $f$  on  $X$  if the restrictions of  $f$  to all closed intervals contained in  $X$  converge uniformly.

### 6.3.1 Examples: pointwise, uniform and almost uniform convergence of function sequences

- (i) Consider the sequence of functions  $\{f_n\}$  with  $f_n = x^n$  defined on the closed interval  $[0, 1]$ . Computing  $\lim_{n \rightarrow \infty} x^n$  we get that the limit is the discontinuous function  $f(x) = 0$  for  $x < 1$  and  $f(1) = 1$ , as we see in the following interactive illustration:

```
In[.]:= Manipulate[Plot[Evaluate[{{x^n, Piecewise[
  {{0, x < 1}}, 1]}], {x, 0, 1}, PlotStyle ->
  {Black, Orange}, PlotRange -> {0, 1.1},
  Epilog -> {Orange, PointSize[0.02],
  Point[{1, 1]}], Axes -> False],
  {n, 1, 50, 1, Appearance -> "Labeled"}]
```

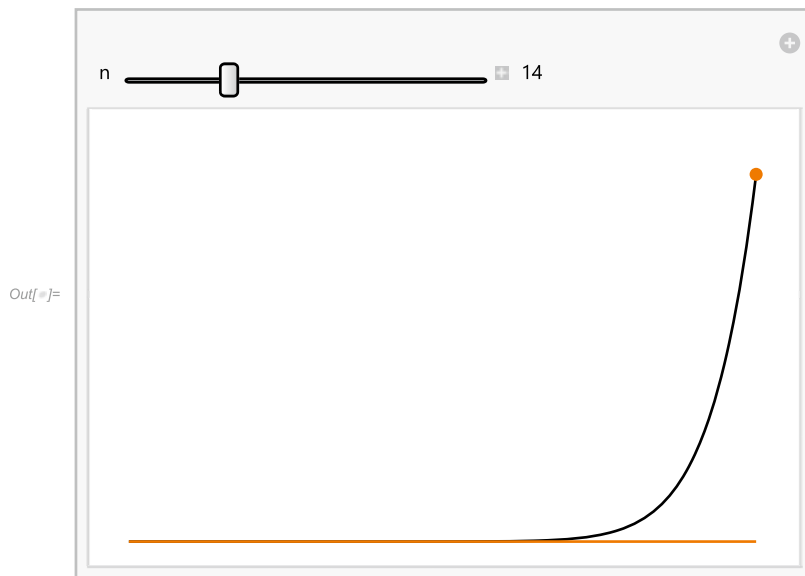


Figure 6.8

On the other hand, for each fixed  $n$ , the distance between  $f_n$  and  $f$  is clearly 1, hence the sequence does not converge uniformly. It also does not converge almost uniformly, since it does not converge on the entire interval  $[0, 1]$ , which is closed. Note that the limit function is discontinuous although all the functions in the sequence are continuous.

If we restrict the domain to the half closed interval  $[0, 1)$  the answer will change. The limit function is now 0 (a continuous functions). The sequence is still not uniformly convergent but it is almost uniformly convergent, since it is clearly uniformly convergent on every closed subinterval of  $[0, 1)$ .

- (ii) Consider the sequence  $\{f_n\}$  with  $f_n(x) = x^n - x^{n+1}$  on the interval  $[0, 1]$ , illustrated below:

```
In[ ]:= Manipulate[Plot[x^n - x^(n + 1), {x, 0, 1},
  PlotRange -> {0, 0.25}], {n, 1, 50, 1,
  Appearance -> "Labeled"}]
```

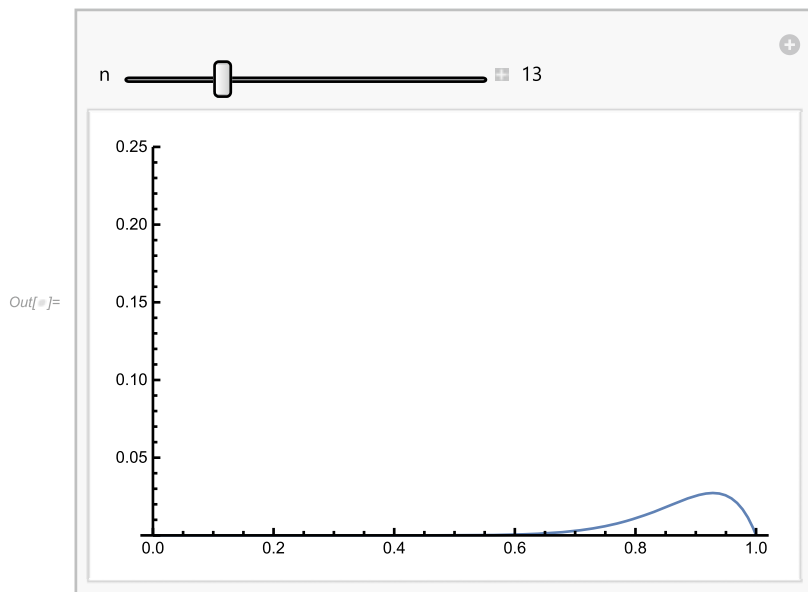


Figure 6.9

The limit function is the constant function 0. To check uniform convergence we compute the distance between the  $n$ -th term of the sequence and the limit function, which is just the supremum norm of the  $n$ -th term. We first find the critical points:

```
In[ ]:= f[n_][x_] := x^n - x^(n + 1)
```

```
In[ ]:= x[n_] = x /. First[Solve[D[f[n][x], x] == 0, x]]
```

```
Out[ ]:=  $\frac{n}{1 + n}$ 
```

For each  $n$  we have only one critical point  $x_n$  and since the function is 0 at both ends,  $f_n$  assumes the maximum value at this point. We see that

$$\|f_n\| = |f_n(x_n)| = \left(\frac{n}{n+1}\right)^n - \left(\frac{n}{n+1}\right)^{n+1}.$$

Since

```
In[.]:= Limit[f[n][x[n]], n -> Infinity]
```

```
Out[.]:= 0
```

the sequence  $\{f_n\}$  is uniformly convergent.

- (iii) For the sequence of functions  $\{f_n\}$  with  $f_n(x) = x^n - x^{2n}$  the situation is somewhat different. The limit function is again the zero function but a calculation analogous to the one in (ii) gives the value at the moving maximum equal to  $1/4$ , which is independent of  $n$ . That means that again the convergence is not uniform on  $[0, 1]$  but is almost uniform on  $[0, 1)$  since the maximum can be moved to the right of any closed subinterval:

```
In[.]:= Manipulate[Plot[x^n - x^(2*n), {x, 0, 1},
  PlotRange -> {0, 0.25}], {n, 1, 30, 1,
  Appearance -> "Labeled"}]
```

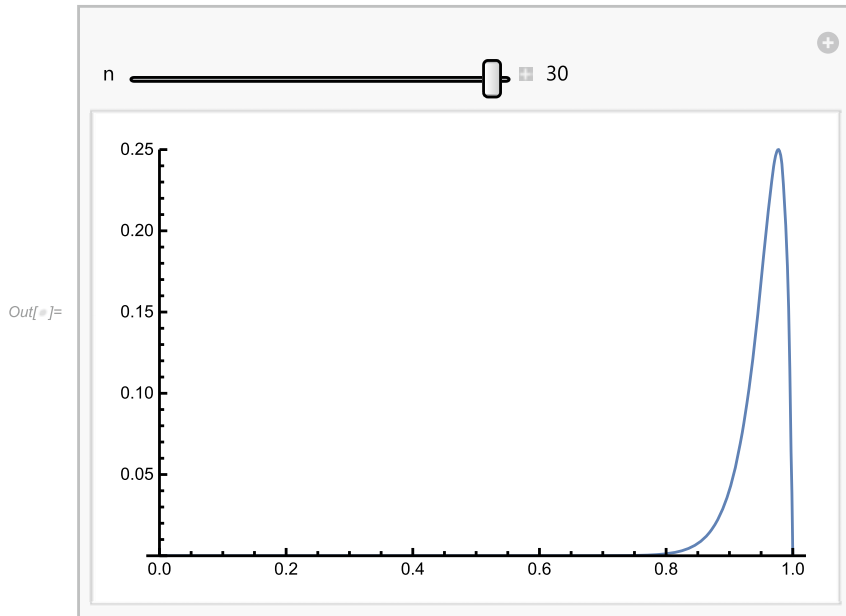


Figure 6.10

- (iv) For the sequence  $\{f_n\}$  with  $f_n(x) = nx/(1 + n + x)$  on  $[0, \infty)$  we have

```
In[.]:= Limit[(n*x)/(1 + n + x), n -> Infinity,
  Assumptions -> x > 0]
```

```
Out[.]:= x
```

So the sequence converges pointwise to the function  $f(x) = x$ . Is the convergence uniform? The distance between  $f_n$  and  $f$  is given by

**In[.]**:= Maximize[{x - (n\*x)/(1 + n + x), x >= 0}, x]

**Out[.]**:= {∞, {x -> Indeterminate}}

Hence the sequence does not converge uniformly. We can see that if we restrict ourselves to any closed interval (by moving the two locator points in the interactive plot below), then the distance between the two restrictions of the functions in the sequence and the limiting function can be made arbitrarily small:

**In[.]**:= Manipulate[Plot[{(n\*x)/(1 + n + x), x},  
 {x, 0, 5}, PlotRange -> {{0, 5}, {0, 5}},  
 Prolog -> {Red, Line[{p, q}], Line[  
 {{First[p], (n\*First[p])/(1 + n + First[p])},  
 {First[p], First[p]}}], Line[{{First[q],  
 (n\*First[q])/(1 + n + First[q])},  
 {First[q], First[q]}}]}], {n, 1, 100, 1,  
 Appearance -> "Labeled"}, {{p, {2, 0}}, {0, 0},  
 {5, 0}, Locator}, {{q, {3, 0}}, {0, 0}, {5, 0},  
 Locator}]

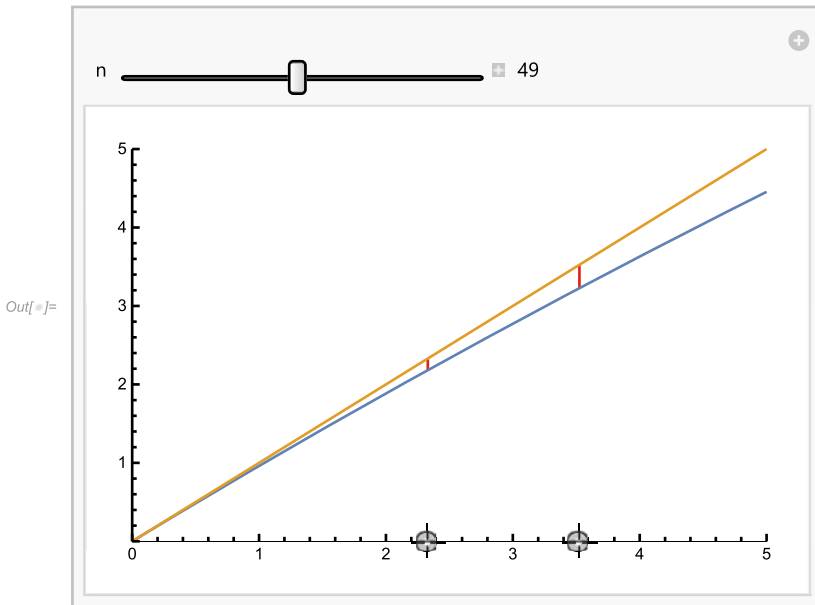


Figure 6.11

### 6.3.2 Continuity and differentiability of limits and sums

We saw in the first example above (see Section 6.3.1 (i)) that a sequence of continuous functions can converge pointwise to a discontinuous one. One of the most important properties of uniform convergence is the fact that the limit of a uniformly convergent sequence of continuous functions is continuous. This phenomenon is best understood in terms of interchanging limit operations: if  $\{f_n\}$  is uniformly convergent on  $X$ , then for every  $a \in X$  we have the following statement.

**Theorem 20** ([14, Theorem 9.12]). *The limit of a uniformly convergent sequence of continuous functions on  $X$  is continuous on  $X$ . That is, for each  $a \in X$*

$$\lim_{x \rightarrow a} \left( \lim_{n \rightarrow \infty} f_n(x) \right) = \lim_{n \rightarrow \infty} \left( \lim_{x \rightarrow a} f_n(x) \right).$$

In general, of course, limits cannot be interchanged (one example is the sequence  $f_n(x) = x^n$  on  $[0, 1]$  at the point 1).

It is obvious that the conclusion of the theorem holds also for sequences of functions which are only almost uniformly convergent, since continuity is a “local property”; in other words, to decide whether a function is continuous or not at some point  $a$  we only need to know its values in some neighborhood of  $a$ , which can always be taken to be a closed interval. Note that this observation immediately tells us that the sequence in Section 6.3.1 (i) cannot be almost uniformly convergent on  $[0, 1]$ , because the limit function is not continuous.

The situation with differentiability is more complicated. Let us consider the family of differentiable functions  $\{f_n\}$  with  $f_n(x) = \sqrt{x^2 + 1/n}$ , where  $x \in \mathbb{R}$ . The sequence is pointwise convergent to

```
In[ ]:= Limit[Sqrt[x^2 + 1/n], n -> Infinity,
  Assumptions -> Element[x, Reals]]
```

```
Out[ ]:= Abs[x]
```

which is not differentiable at 0. Evaluating expressions

```
In[ ]:= FullSimplify[Maximize[{Sqrt[x^2 + 1/n] - x,
  x >= 0}, x], Assumptions -> {n > 0}];
```

```
In[ ]:= FullSimplify[Maximize[{Sqrt[x^2 + 1/n] + x,
  x <= 0}, x], Assumptions -> {n > 0}];
```

we see that the maximum is  $1/\sqrt{n}$ , which tends to 0 as  $n \rightarrow \infty$ . Hence, a uniformly convergent sequence of differentiable functions may have a limit that is not differentiable.

```
In[ ]:= Manipulate[Plot[{Sqrt[x^2 + 1/n], Abs[x]},
  {x, -1, 1}, PlotRange -> {{-1, 1}, {0, 2}},
  AxesOrigin -> {0, 0}, AspectRatio -> Automatic],
  {n, 1, 1000, 1, Appearance -> "Labeled"}]
```

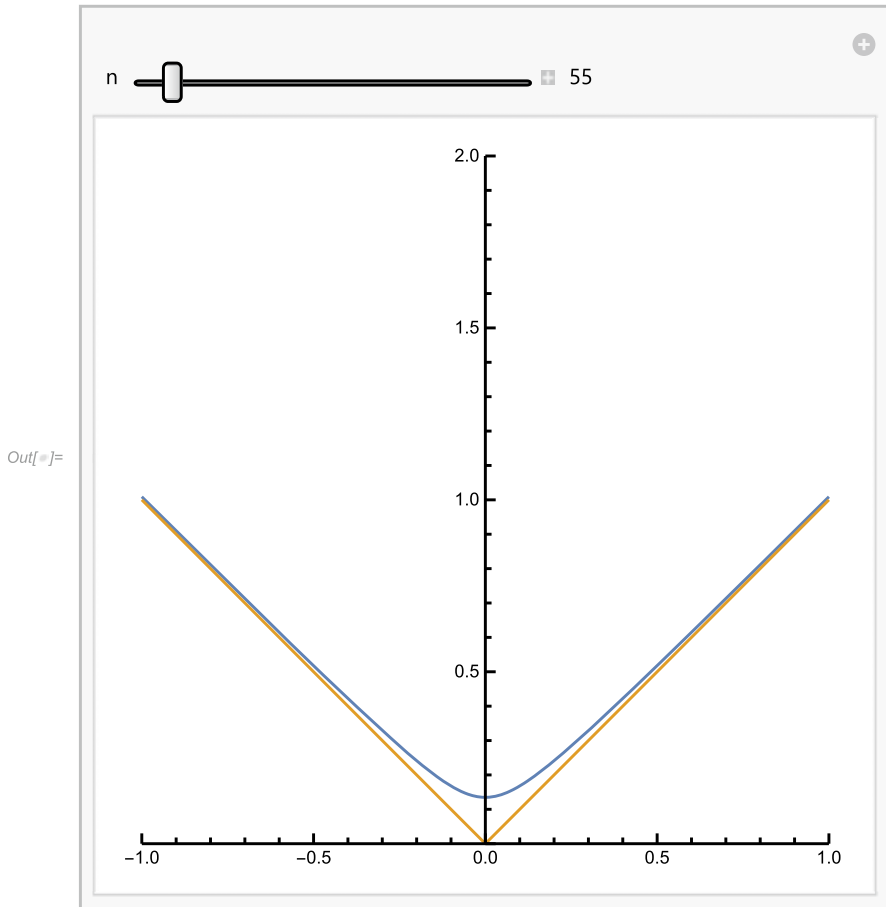


Figure 6.12

Even when a sequence of differentiable functions  $\{f_n\}$  converges uniformly to a differentiable function  $f$ , it could happen that at some point  $a$  the sequence of derivatives  $f'_n(a)$  does not converge to  $f'(a)$ . This happens for the sequence  $\{f_n\}$  with  $f_n(x) = x/(nx^2 + 1)$  on  $\mathbb{R}$ , which converges uniformly to 0, but  $f'_n(0) = 1$  (see [14, p.407]):

```
In[ ]:= Manipulate[Plot[x/(n*x^2 + 1), {x, -1, 1},
  PlotRange -> {{-1, 1}, {-0.5, 0.5}},
  {n, 1, 1000, 1, Appearance -> "Labeled"}]
```

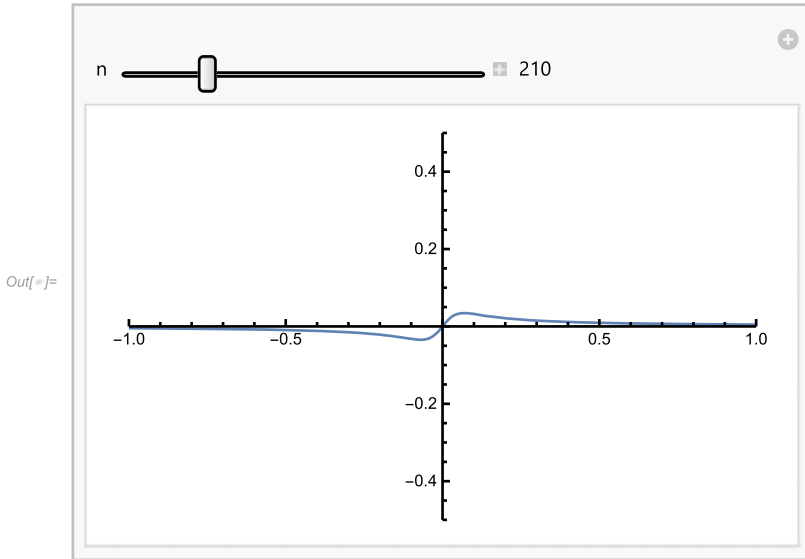


Figure 6.13

A sufficient condition for the limit of a sequence of differentiable functions to be differentiable is given by the following theorem.

**Theorem 21** (Differentiation of a sequence of functions [14, Theorem 9.40]). *Suppose that the sequence of functions  $\{f_n\}$  is such that*

- (i)  $f_n$  is of class  $C^1$  on  $[a, b]$ ;
- (ii) there is some point  $x_0 \in [a, b]$  such that the sequence  $\{f_n(x_0)\}$  converges;
- (iii) the sequence of derivatives  $\{f'_n\}$  converges uniformly on  $[a, b]$  to  $g$ .

Then  $\{f_n\}$  converges uniformly to some  $f$  on  $[a, b]$  such that  $f'(x) = g(x)$  on  $[a, b]$ .

In particular, this means that if a sequence of functions of class  $C^1$  on  $X$  is convergent on  $X$  and its sequence of derivatives is almost uniformly convergent, then the limit of the sequence is a differentiable function whose derivative is the limit of the derivatives of the functions in the given sequence.

The theory of convergence of sequences of functions is quite analogous to the theory of convergence of number sequences. Most results which can be formulated for number sequences also hold for function sequences with the absolute value  $|\cdot|$  replaced by the norm  $\|\cdot\|$ . For example, the *Cauchy criterion* takes the following form. A sequence of functions  $\{f_n\}$  is uniformly convergent if and only if for every  $\varepsilon > 0$  there is an integer  $N$  such that for every pair of integers  $n, m$  larger than  $N$ ,  $\|f_n - f_m\| < \varepsilon$ .

Series of functions are defined in the same way as they are defined for numbers: they are simply sequences of partial sums (in which the sums are formed using the

usual addition of functions). Thus we can speak of pointwise, uniform and almost uniform convergence of series.

The above theorems about sequences of functions imply corresponding theorems for series. For example, the sum of an almost uniformly convergent series of continuous functions is continuous. Also, if  $\{f_n\}$  is a sequence of functions of class  $C^1$  on  $[a, b]$  such that for some  $x_0 \in [a, b]$  the series  $\sum_{n=0}^{\infty} f_n(x_0)$  is convergent and the series  $\sum_{n=0}^{\infty} f'_n$  is uniformly convergent on  $[a, b]$ , then  $\sum_{n=0}^{\infty} f_n$  is uniformly convergent on  $[a, b]$  to a differentiable function and its derivative is  $\sum_{n=0}^{\infty} f'_n$  (see [14, Corollary 9.41]). For this reason, in the case of function series we are almost always interested in determining whether it is (almost) uniformly convergent. Although there is no algorithm that can be used, just as in the case of numbers series, there are a number of tests. Some of them reduce the problem to a problem involving sequences or series of numbers. Next we present some basic tests.

Let  $\{f_n\}$  be a sequence of functions defined on  $X$ .

- (1.) A necessary condition for (almost) uniform convergence of  $\sum_{n=0}^{\infty} f_n$  is that the sequence  $\{f_n\}$  converges (almost) uniformly to the zero function, which is equivalent to the condition that the sequence of numbers  $\{\|f_n\|\}$  converges to 0.
- (2.) A sufficient condition for (almost) uniform convergence of  $\sum_{n=0}^{\infty} f_n$  is that the series of positive numbers  $\sum_{n=0}^{\infty} \|f_n\|$  converges, which follows from the Cauchy criterion. Clearly in this case the series  $\sum_{n=0}^{\infty} f_n$  also converges absolutely.
- (3.) The most difficult cases are when the necessary condition  $\|f_n\| \rightarrow 0$  is satisfied but the series  $\sum_{n=0}^{\infty} \|f_n\|$  diverges. In such situations we need to use other methods. One of them is the Dirichlet test.

**Theorem 22** (Dirichlet's test for uniform convergence [14, Theorem 9.29]). *Suppose that  $\{b_n\}$  is a sequence of (non-negative) functions on  $X$  such that  $b_n(x) \geq b_{n+1}(x)$  and  $b_n$  tends to zero uniformly on  $X$ . If  $\{a_n\}$  is a sequence of functions such that  $|s_n(x)| \leq M$  for all  $n$  and  $x \in X$ , where  $s_n(x) = \sum_{k=1}^n a_k(x)$ , then  $\sum_{k=1}^{\infty} a_k(x)b_k(x)$  converges uniformly on  $X$ .*

The problem with the above sufficient condition (2.) is the need to compute the suprema of  $f_n$  over  $X$ , which, as we know, can sometimes be a difficult problem. Often, it is easier to find an upper bound for the values of a function than to find the least upper bound (supremum). This makes the following test useful when trying to prove uniform convergence of function series.

**Theorem 23** (Weierstrass M-test [14, Theorem 9.25]). *Let  $\{M_n\}$  be a sequence of non-negative real numbers and let  $\{f_n\}$  be a sequence of functions defined on  $X$ , such that  $|f_n(x)| \leq M_n$  for all  $x \in X$  and each  $n \in \mathbb{N}$ . If the series  $\sum_{k=1}^{\infty} M_k$  converges, then the series  $\sum_{k=1}^{\infty} f_k$  converges uniformly (and absolutely) on  $X$ .*



Finally we can state a very important result about power series, which follows from the Weierstrass M-test.

**Theorem 24** ([14, Theorem 9.26]). *Let  $R > 0$  be the radius of convergence of the power series  $\sum_{k=0}^{\infty} a_k(x - x_0)^k$  and let  $0 < r < R$ . Then  $\sum_{k=0}^{\infty} a_k(x - x_0)^k$  converges uniformly on  $[x_0 - r, x_0 + r]$ . In other words, any power series is almost uniformly (and absolutely) convergent in the interior of its region of convergence.*

It follows from this theorem that a power series  $\sum_{k=0}^{\infty} a_k(x - x_0)^k$  is infinitely many times differentiable in the interior of its region of convergence and its derivative can be computed by “term-by-term differentiation”, i. e., it is equal to the power series  $\sum_{k=1}^{\infty} k a_k(x - x_0)^{k-1}$ . From this in turn it follows that if a function  $f$  can be expressed as a convergent power series, then this series must be its Taylor series. Indeed, if  $f(x) = \sum_{k=0}^{\infty} a_k(x - x_0)^k$ , then by differentiating repeatedly term-by-term and substituting  $x = x_0$  we obtain  $a_n = f^{(n)}(x_0)/n!$ , which is the  $n$ -th coefficient of the Taylor series with center  $x_0$ .

We have just seen that a power series determines a continuous function on its region of convergence. This function is differentiable infinitely many times, and all of its derivatives are also given by power series with the same radius of convergence. Note that the derivative of the power series may not be convergent at an endpoint, even if the original power series itself is convergent there:

**In[.]**:= SumConvergence[x^n\*((-1)^n/n), n]

**Out[.]**:= SumConvergence[x^n\*((-1)^n/n), n] || x == 1

but for the derivative we have

**In[.]**:= SumConvergence[x^n\*(-1)^n, n]

**Out[.]**:= Abs[x] < 1

### 6.3.3 Examples: pointwise, uniform and almost uniform convergence of function series

- (i) Let us investigate pointwise, uniform and almost uniform convergence of the series  $\sum_{n=1}^{\infty} x^2/(n^4 + x^4)$  for  $x \in \mathbb{R}$ .

We easily see that the sequence  $\{f_n\}$ , where  $f_n = x^2/(n^4 + x^4)$ , uniformly converges to 0:

**In[.]**:= Manipulate[Plot[x^2/(n^4 + x^4), {x, -10, 10},  
PlotRange -> {0, 1}], {n, 1, 10, 1}]

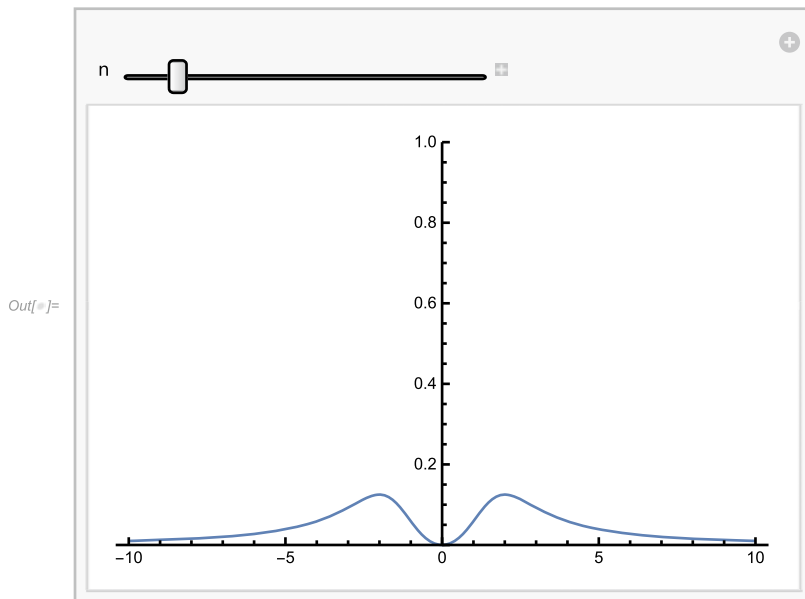


Figure 6.14

In this case we can actually easily find the norms of the functions  $f_n$ :

```
In[ ]:= FullSimplify[Maximize[x^2/(n^4 + x^4), x],
Assumptions -> Element[n, Integers] && n > 0]
Out[ ]:= {1/(2 n^2), {x -> -n}}
```

Since the series  $\sum_{n=1}^{\infty} 1/(2n^2)$  converges, our series is uniformly convergent. Instead of computing the norms, we could have observed that the inequality

$$\frac{x^2}{n^4 + x^4} \leq \frac{1}{2n^2}$$

holds for all  $x \in \mathbb{R}$ , hence the result follows from the Weierstrass M-test.

(ii) Let us investigate the convergence of the series  $f(x) = \sum_{n=0}^{\infty} x e^{-nx}$  on  $(0, \infty)$ .

For a fixed  $x$ , the series  $\sum_{n=0}^{\infty} x e^{-nx}$  is just the geometric series with constant ratio  $r = e^{-x}$  with  $0 < r < 1$ , hence the series is pointwise convergent. Next let us find the norms of the functions  $f_n(x) = x e^{-nx}$ :

```
In[ ]:= x /. Solve[D[x/E^(n*x), x] == 0, x, Reals][[1, 1]]
Out[ ]:= 1/n
```

Hence the maximum of  $f_n$  is attained at  $x = 1/n$  and is equal to  $1/(en)$ . Thus, the necessary condition for uniform convergence is satisfied but the series of norms diverges. Hence we have to use a different approach.

In this case the easiest thing to do is to note that we can actually compute the sum of the series  $s$ , its partial sums  $s_n$  and hence the remainders  $r_n = s - s_n$  because they are all sums of geometric series. Thus

$$r_n = s - s_n = \sum_{k=n+1}^{\infty} x e^{-kx} = \frac{x e^{-nx}}{e^x - 1}.$$

So the question of the uniform convergence of our series reduces to the question whether  $\{g_n\}$  with  $g_n = x e^{-nx} / (e^x - 1)$  converges uniformly to 0 on  $(0, \infty)$ . This is clearly not true on the whole  $(0, \infty)$ , but it is true on any compact subset in  $(0, \infty)$  either by looking at the following illustration:

```
In[.]:= Manipulate[Plot[x/(E^(n*x))*(E^x - 1)), {x, 0, 10},
  PlotRange -> {0, 1}], {n, 1, 10, 1}]
```

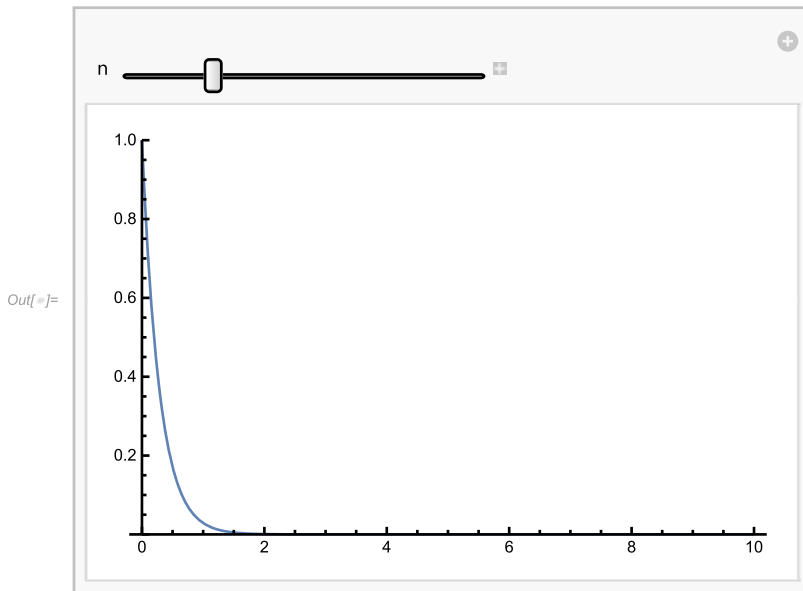


Figure 6.15

or by computing

```
In[.]:= Limit[x/(E^(n*x))*(E^x - 1)), x -> 0]
Out[.]:= 1
```

Hence the series is not uniformly convergent but is almost uniformly convergent.

### 6.3.4 Example

Let us show that the function  $f(x) = \cos(\sqrt{x})$  is differentiable infinitely many times at 0 and let us find its fifth derivative.

Strictly speaking we should speak of right derivatives or derivatives from above, since the formula is not defined for negative  $x$ . Observe, however, that for non-negative  $x$  the function  $f(x)$  is given by the series  $\sum_{n=0}^{\infty} (-1)^n x^n / (2n)!$  and the series is defined and convergent for all  $x$ . But, as we said earlier, this means that it must be the Taylor series with center at 0 of the function  $g(x) = \sum_{n=0}^{\infty} (-1)^n x^n / (2n)!$ , where  $g$  is an extension of  $f$  to the whole real axis. Hence its fifth derivative at 0 will be

$$\mathbf{In[.]} := (5! * (-1)^5) / 10!$$

$$\mathbf{Out[.]} := -\frac{1}{30240}$$

This can be verified either by a direct computation of derivatives (note that we need to use `Limit` here as direct substitution will not work):

$$\mathbf{In[.]} := \text{Limit}[D[\text{Cos}[\text{Sqrt}[x]], \{x, 5\}], x \rightarrow 0]$$

$$\mathbf{Out[.]} := -\frac{1}{30240}$$

or by computing the fifth coefficient of the Taylor series:

$$\mathbf{In[.]} := 5! * \text{SeriesCoefficient}[\text{Cos}[\text{Sqrt}[x]], \{x, 0, 5\}]$$

$$\mathbf{Out[.]} := -\frac{1}{30240}$$

Note that we can actually plot the graph of the function over the negative and positive axis:

$$\mathbf{In[.]} := \text{Plot}[\text{Cos}[\text{Sqrt}[x]], \{x, -10, 100\}]$$

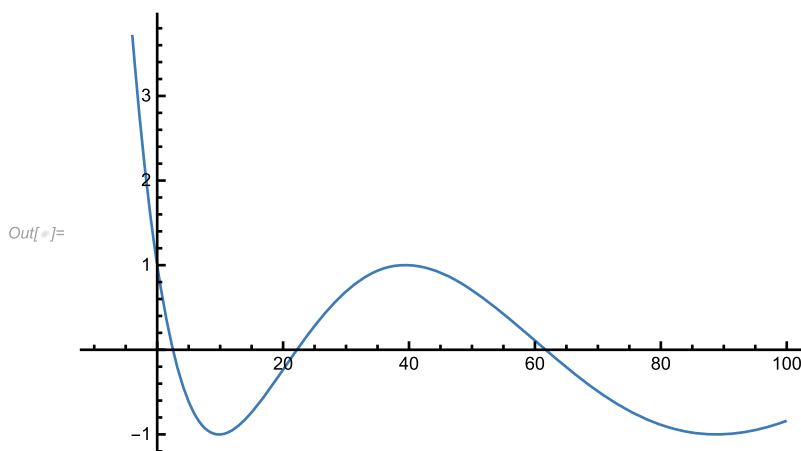


Figure 6.16

The reason why this works is that the function  $\cos$  is defined on the whole complex plane and takes real values for pure imaginary arguments, as can be seen by evaluating

**In[.]**:= ComplexExpand[Cos[I\*x]]

**Out[.]**:= Cosh[x]

where the function  $\cosh$  was defined in Section 5.3.3.



# 7 Integration

In this chapter we first define an indefinite integral (antiderivative) of a function and discuss the Risch algorithm which is used by Mathematica<sup>®</sup> to compute it. Then we define the Riemann integral, state the fundamental theorem of calculus and consider some applications.

## 7.1 Indefinite integrals

Let  $f : I \rightarrow \mathbb{R}$  be a function, where  $I \subset \mathbb{R}$  is an interval. We say that a differentiable function  $F : I \rightarrow \mathbb{R}$  is an *indefinite integral* of  $f$  on  $I$  (or an *antiderivative* or a *primitive function*) if  $F'(x) = f(x)$  for all  $x \in I$ . This naturally leads to two questions. One is: what kind of functions have antiderivatives? Clearly, not all. For example, we know already that the derivative of any function must have the Darboux property, hence a function that does not have this property, for example, a function that has a “simple jump” discontinuity, cannot have a primitive function.

Later we shall see that any continuous function has an antiderivative (the fundamental theorem of calculus). As we have already seen (example in Section 5.3.2) a function may have an antiderivative even if it is not continuous. Moreover, if a function has an antiderivative, it has infinitely many of them, since if we add any constant to an antiderivative of a function we will get another antiderivative. Note that in practice this means that two antiderivatives of the same function can look quite different: for example both  $\sin^2 x$  and  $-\cos(2x)/2$  are antiderivatives of  $\sin(2x)$ . That causes a linguistic difficulty: should one speak of the antiderivative or an antiderivative of a function? We shall usually use the definite article when we refer to the whole class of antiderivatives and to the indefinite one when we want to refer to a particular antiderivative.

The antiderivative of a function  $f$  is usually denoted by  $\int f(x) dx$ . In Mathematica<sup>®</sup> one computes antiderivatives by using the function `Integrate`, e. g.,

```
In[.]:= Integrate[x*Cos[x], x]
```

```
Out[.]:= Cos[x] + x Sin[x]
```

In many books on calculus and analysis one writes  $x \sin x + \cos x + c$ , where  $c$  denotes an arbitrary constant. We can obtain this kind of answer by solving the *differential equation*

$$\frac{dF(x)}{dx} = f(x).$$

This is done using the function `DSolve`. For example:

```
In[.]:= DSolve[F'[x] == x*Cos[x], F[x], x]
```

```
Out[.]:= {{F[x] -> C[1] + Cos[x] + x Sin[x]}}
```

<https://doi.org/10.1515/9783110590142-007>

The second natural question is: can we find explicitly an antiderivative of a given function? If the function  $f$  is continuous, then the fundamental theorem of calculus (which we will discuss later in this chapter) gives us an antiderivative  $\int_a^x f(x) dx$ .

However, one really wishes to have something analogous to the case of differentiation. In other words, we would like to have an algorithm which, for some large family of functions whose antiderivatives belong to this family, computes the antiderivatives of functions obtained by algebraic operations and compositions of functions in the family and expresses them again in terms of functions in the family. Let us make this statement a little more precise. We consider the family of “elementary functions”, which includes rational functions, exponentials, logarithms and algebraic functions (e. g., solutions of polynomial equations whose coefficients are elementary functions), as well as trigonometric, inverse trigonometric, hyperbolic and inverse hyperbolic functions. We require this family to be closed under composition. The question can now be formulated as follows: is there an algorithm which, given an elementary function, returns an elementary function which is its antiderivative? (Of course, this means that when an elementary function does not have an elementary antiderivative the algorithm should inform us of that.)

The study of this problem began in the nineteenth century. For a long time it was believed that no algorithm of this kind could exist. Instead, a number of “heuristic tricks” were developed which try to reduce certain integrals to certain already known ones. One such trick is the so called “integration by parts”. This is based on the Leibniz formula for differentiation and takes the form

$$\int f(x) \frac{dg(x)}{dx} dx = f(x)g(x) - \int g(x) \frac{df(x)}{dx} dx.$$

The point of this kind of formula is that it sometimes allows to replace the problem of finding the antiderivative of a function by the problem of finding the antiderivative of a simpler function. Successful use of such tricks requires skill and luck. For example,

$$\int xe^x dx = \int x \frac{de^x}{dx} dx = xe^x - \int e^x \frac{dx}{dx} dx = e^x(x - 1) + c.$$

However, the indefinite integral  $\int xe^{x^2} dx$  cannot be found using integration by parts and requires a different trick (substitution). Such a “bag of tricks” is still taught in most university courses. We shall not consider it here, as there are many books on the subject and Mathematica<sup>®</sup> uses a very different approach. It is based on an algorithm discovered in the 1960s by Risch, based on nineteenth century work of Liouville and Hermite and early twentieth century work of Hardy [8]. The algorithm is known as the *Risch algorithm* and is used by most modern computer algebra systems, including Mathematica<sup>®</sup>. The algorithm can very rapidly find the antiderivatives of extremely complicated looking functions. For example [8], the following complicated elementary function has an elementary antiderivative:



$$\mathbf{In[.]} := \int \left( (x(x+1)) \left( (x^2 E^{2x^2} - \text{Log}[x+1])^2 + 2 E^{3x^2} x (x - (2x^3 + 2x^2 + x + 1) \text{Log}[x+1]) \right) / \left( (x+1) \text{Log}[x+1]^2 - (x^3 + x^2) E^{2x^2} \right)^2 \right) dx$$

$$\mathbf{Out[.]} := x - \text{Log}[1+x] - \frac{E^{x^2} x \text{Log}[1+x]}{E^{2x^2} x^2 + \text{Log}[1+x]^2} - \frac{1}{2} \text{Log}[E^{x^2} x - \text{Log}[1+x]] + \frac{1}{2} \text{Log}[E^{x^2} x + \text{Log}[1+x]]$$

In principle the algorithm can decide if the antiderivative of an elementary function can be expressed in an elementary form and returns either this antiderivative or information that no such derivative exists. But this is currently impossible to realize in practice. First of all, the Risch algorithm has many branches and some of them have a very high complexity, which means that a complicated case may take impossibly long to compute. Most computer programs including *Mathematica*<sup>®</sup> do not implement such branches at all and when *Mathematica*<sup>®</sup> enters into one of them it sometimes returns the answer unevaluated.

There is another problem that results sometimes from the use of the Risch algorithm, illustrated in the following example. Let us consider the rational function

$$\mathbf{In[.]} := f[x_] := (x^2 + 2x + 4)/(x^4 - 7x^2 + 2x + 17)$$

This function is defined and continuous on  $\mathbb{R}$  since its denominator has no roots in  $\mathbb{R}$ :

$$\mathbf{In[.]} := \text{Solve}[x^4 - 7x^2 + 2x + 17 == 0, x, \text{Reals}]$$

$$\mathbf{Out[.]} := \{\}$$

However, the antiderivative returned by *Mathematica*<sup>®</sup> is not continuous:

$$\mathbf{In[.]} := g[x_] = \text{Integrate}[f[x], x]$$

$$\mathbf{Out[.]} := \frac{1}{2} \text{ArcTan}\left[\frac{-1-x}{-4+x^2}\right] - \frac{1}{2} \text{ArcTan}\left[\frac{1+x}{-4+x^2}\right]$$

$$\mathbf{In[.]} := \text{Plot}[g[x], \{x, 1, 3\}, \text{Exclusions} \rightarrow 2]$$

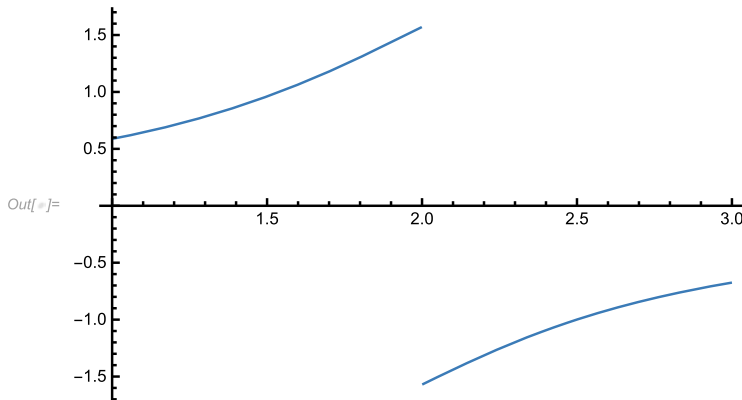


Figure 7.1

As we will see later, when we discuss the fundamental theorem of calculus, we know that a continuous function always has a continuous antiderivative. Looking at the above graph we can see that we could get such an antiderivative by shifting the left branch down or the right branch up so that they meet. We can actually give an explicit formula for a continuous antiderivative by computing first the size of a jump:

```
In[.]:= Limit[g[x], x -> 2, Direction -> "FromBelow"] -
      Limit[g[x], x -> 2, Direction -> "FromAbove"]
```

```
Out[.]:=  $\pi$ 
```

```
In[.]:= h[x_] := Piecewise[{{g[x], x < 2}, {g[x] + Pi, x >
      2}}, Limit[g[x], x -> 2, Direction -> "FromBelow"]]
```

```
In[.]:= Plot[h[x], {x, 1, 3}]
```

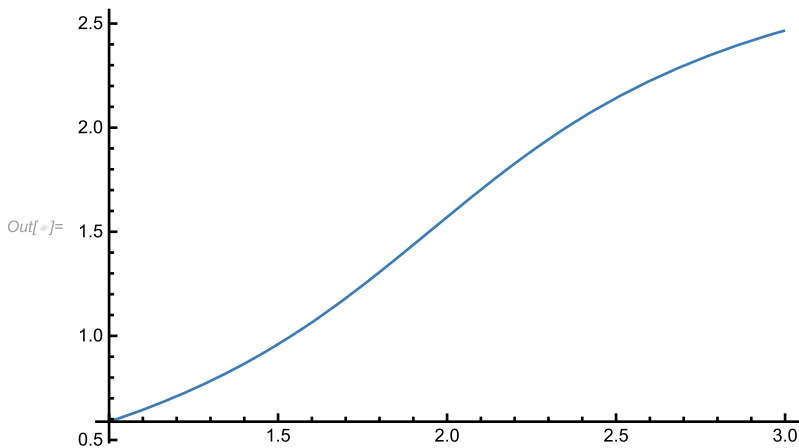


Figure 7.2

The fact that Mathematica<sup>®</sup> sometimes returns a discontinuous antiderivative of a continuous function is not a bug but a consequence of the way the Risch algorithm works. However, note that when we compute a definite integral over an interval, in which the antiderivative found by the Risch algorithm has a discontinuity, Mathematica<sup>®</sup> notices that and returns the correct answer:

```
In[.]:= Integrate[f[x], {x, 1, 3}]
```

```
Out[.]:=  $\pi - \text{ArcTan}\left[\frac{2}{3}\right] - \text{ArcTan}\left[\frac{4}{5}\right]$ 
```

If we simply used the fundamental theorem of calculus with the discontinuous derivative we would have got an incorrect answer:

```
In[.]:= g[3] - g[1]
```

```
Out[.]:=  $-\text{ArcTan}\left[\frac{2}{3}\right] - \text{ArcTan}\left[\frac{4}{5}\right]$ 
```

## 7.2 The Risch algorithm

We will not try to describe the complete Risch algorithm but we will give a sketch of the main ideas. The most detailed survey can be found in [5, 6].

The first observation is that, although the algorithm finds antiderivatives of both real and complex elementary functions, it actually needs complex numbers to work. When we work over the complex numbers, there are fewer elementary functions since all trigonometric and inverse trigonometric functions can be expressed in terms of the exponential function and its inverse, the logarithm. In Mathematica® we can accomplish this conversion by using the function `TrigToExp`:

```
In[.]:= TrigToExp /@ {Sin[x], ArcSin[x], Tan[x], ArcTan[x]}
```

```
Out[.]:= { $\frac{1}{2} I E^{-I x} - \frac{1}{2} I E^{I x}, -I \operatorname{Log}[I x + \sqrt{1-x^2}],$   

 $\frac{I (E^{-I x} - E^{I x})}{E^{-I x} + E^{I x}}, \frac{1}{2} I \operatorname{Log}[1 - I x] - \frac{1}{2} I \operatorname{Log}[1 + I x]}$ }
```

There is a problem here: the logarithm as a function on the set of non-zero complex numbers is not an ordinary function but a multi-valued one:

```
In[.]:= Simplify[Exp[x + 2*Pi*I*n], Element[n, Integers]]
```

```
Out[.]:= Ex
```

Hence, for a complex  $y$  the logarithm of  $y$  has infinitely many branches differing by integer multiples of  $2\pi i$ . The Risch algorithm, being purely algebraic, ignores this issue and this sometimes results in the “wrong branch” problem we saw in the example above.

The most important mathematical result which is the basis of the Risch algorithm was actually proved by Liouville in the nineteenth century and is known as *Liouville’s Principle*. In order to state it in its modern formulation we need some concepts from modern algebra.

### 7.2.1 Differential algebras

**Definition 5.** A *differential field* is a field  $F$  together with an operator  $\mathcal{D}_F : F \rightarrow F$  such that for all  $f, g \in F$

$$\mathcal{D}_F(f + g) = \mathcal{D}_F(f) + \mathcal{D}_F(g),$$

$$\mathcal{D}_F(f g) = g \mathcal{D}_F(f) + f \mathcal{D}_F(g).$$

The operator  $\mathcal{D}_F$  is called a *derivation* or a *differential operator*.

**Definition 6.** A differential field  $(E, \mathcal{D}_E)$  is called an *extension of a differential field*  $(F, \mathcal{D}_F)$  if  $F \subset E$  and for all  $f \in F$

$$\mathcal{D}_E(f) = \mathcal{D}_F(f).$$

**Definition 7.** The *field of constants* of  $(E, \mathcal{D}_E)$  is the set (actually a field) of elements  $K \subset E$  such that

$$K = \{k \in E, \mathcal{D}_E(k) = 0\}.$$

For example, the field of rational functions with rational coefficients  $\mathbb{Q}(x) = \{f(x)/g(x)\}$ , where  $f$  and  $g$  are polynomials, is equipped with the differential operator  $\mathcal{D}(x) = 1$ , which is the ordinary differentiation.

In general  $F(\theta)$  denotes the field of rational functions  $\{f(\theta)/g(\theta)\}$ , where  $f$  and  $g$  are polynomials with coefficients in  $F$ . This is the smallest field that contains  $F$  and  $\theta$ . By induction we can define  $F(\theta_1, \theta_2, \dots, \theta_n) := F(\theta_1, \theta_2, \dots, \theta_{n-1})(\theta_n)$ .

**Definition 8.** Let  $F$  be a differential field and let  $E$  be a differential extension of  $F$ .

1. An element  $v \in E$  for which there exists  $u \in F$  such that

$$\mathcal{D}(v) = \frac{\mathcal{D}(u)}{u}$$

is said to be *logarithmic* over  $F$ . In this case we write  $v = \log(u)$ .

2. An element  $v \in E$  for which there exists  $u \in F$  such that

$$\frac{\mathcal{D}(v)}{v} = \mathcal{D}(u)$$

is said to be *exponential* over  $F$ . In this case we write  $v = \exp(u)$ .

3. If for  $v \in E$  there exists a polynomial  $g \in F[z]$  such that  $g(v) = 0$ , we say that  $v$  is *algebraic* over  $F$ . If  $v$  is not algebraic over  $F$  we say that it is *transcendental* over  $F$ .
4. Let  $E$  be an extension field of a differential field  $F$ .  $E$  is called an *elementary extension* of  $F$  if it can be obtained from  $F$  by successively taking logarithmic, exponential or algebraic extensions. In other words, we can write  $E = F(\theta_1, \theta_2, \dots, \theta_n)$ , where  $\theta_i$  is logarithmic, exponential or algebraic over  $F(\theta_1, \theta_2, \dots, \theta_{i-1})$  for each  $i = 1, \dots, n$ .

Now we can state Liouville's Principle.

**Theorem 25** (Liouville's Principle). *Let  $F$  be a differential field with a constant field  $K$ . For  $f \in F$ , suppose there is some  $g \in E$ , where  $E$  is an elementary extension of  $F$  with the same field of constants  $K$ , such that  $\mathcal{D}(g) = f$ . Then there exist  $v_0, \dots, v_m \in F$  and constants  $c_1, \dots, c_m \in K$  such that*

$$f = \mathcal{D}(v_0) + \sum_{i=1}^m \frac{c_i \mathcal{D}(v_i)}{v_i}.$$

This can also be written as

$$\int f = v_0 + \sum_{i=1}^m c_i \log(v_i). \quad (7.10)$$

Liouville's Principle does not, of course, give us an algorithm for finding antiderivatives: such algorithms have to be constructed separately. However, the principle tells us that any elementary function, which has an elementary antiderivative, must have one of the form (7.10) (it may also have antiderivatives which do not have this form). There are, in fact, a number of algorithms which compute antiderivatives in the form (7.10) for different types of extensions. The difference between them concerns essentially efficiency of computation – a very important issue for practical computations but one which we will ignore here.

### 7.2.2 Example 1: integration of rational functions

Let us first describe the most basic algorithm which illustrates Liouville's Principle – an algorithm for finding antiderivatives of rational functions with coefficients in  $\mathbb{R}$ . A version of this algorithm is taught in all calculus courses. Rather than trying to describe the general theory, we will consider a typical example. Suppose we wish to find the antiderivative of the following rational function:

$$\mathbf{In[ ]} := f[x\_ ] := (x^5 + x + 1)/(x^4 - 2x^3 + 2x^2 - 2x + 1)$$

Mathematica<sup>®</sup> returns the answer

$$\begin{aligned} \mathbf{In[ ]} &:= \text{Integrate}[f[x], x] \\ \mathbf{Out[ ]} &:= \frac{1}{4} \left( -10 - \frac{6}{-1+x} + 8x + 2x^2 - 4\text{ArcTan}[x] \right. \\ &\quad \left. + 6\text{Log}[-1+x] + \text{Log}[1+x^2] \right) \end{aligned}$$

This does not have the Liouville form (7.10) but we know that we can convert it to one by means of `TrigToExp`. So let us now consider how to obtain an antiderivative in Liouville's form directly.

In general, given a rational function of the form  $p(x)/q(x)$ , where the degree of  $p(x)$  is larger than that of  $q(x)$ , we use polynomial division to reduce it to the form  $h(x) + p_1(x)/q(x)$ , where  $\deg p_1(x) < \deg q(x)$ . In Mathematica<sup>®</sup> we can use the function `PolynomialQuotientRemainder` to do this for us:

$$\begin{aligned} \mathbf{In[ ]} &:= \text{pqr} = \text{PolynomialQuotientRemainder}[\text{Numerator}[f[x]], \\ &\quad \text{Denominator}[f[x]], x] \\ \mathbf{Out[ ]} &:= \{2 + x, -1 + 4x - 2x^2 + 2x^3\} \end{aligned}$$

Hence  $f(x)$  is equal to

$$\begin{aligned} \mathbf{In[ ]} &:= \text{First}[pqr] + \text{Last}[pqr]/\text{Denominator}[f[x]] \\ \mathbf{Out[ ]} &:= 2 + x + \frac{-1 + 4x - 2x^2 + 2x^3}{1 - 2x + 2x^2 - 2x^3 + x^4} \end{aligned}$$

So we only need to find the antiderivative of

$$\mathbf{In[ ]} := g[x\_ ] := \text{Last}[pqr]/\text{Denominator}[f[x]]$$

Next, we compute all the complex roots of the denominator:

```
In[.]:= roots = x /. Solve[Denominator[g[x]] == 0, x]
Out[.]:= {-I, I, 1, 1}
```

Thus our denominator factorizes as

```
In[.]:= Times @@ (x - #1 &) /@ roots
Out[.]:= (-1 + x)^2 (-I + x) (I + x)
```

For this situation when  $\deg p(x) < \deg q(x)$ , one can prove that an antiderivative of the following form always exists:

$$\int \frac{p(x)}{q(x)} dx = \frac{p_1(x)}{q_1(x)} + \sum_{i=1}^n c_i \log(x - a_i),$$

where  $q(x) = \prod_{i=1}^n (x - a_i)^{d_i}$ ,  $q_1(x) = \prod_{i=1}^n (x - a_i)^{d_i-1}$  and  $p_1(x)$  is a polynomial with  $\deg p_1(x) < \deg q_1(x)$ .

In our case an antiderivative of  $g(x)$  must have the form

```
In[.]:= int = a/(x - 1) + b*Log[x - 1] + c*Log[x - I]
      + d*Log[x + I];
```

We can now compute the parameters  $a, b, c, d$  by

```
In[.]:= sols = SolveAlways[g[x] == D[int, x], x]
Out[.]:= {{a -> -3/2, b -> 3/2, c -> 1/4 + I/2, d -> 1/4 - I/2}}
```

and obtain the antiderivative in Liouville's form:

```
In[.]:= int /. sols[[1]]
Out[.]:= -3/(2(-1 + x)) + 3/2 Log[-1 + x] + (1/4 + I/2) Log[-I + x]
      + (1/4 - I/2) Log[I + x]
```

If we want to get a solution not containing complex numbers, we can take the real part and use `ComplexExpand`:

```
In[.]:= ComplexExpand[Re[%]]
Out[.]:= -3/(2(-1 + x)) - 1/2 Arg[-I + x] + 1/2 Arg[I + x]
      + 3/4 Log[(-1 + x)^2] + 1/4 Log[1 + x^2]
```

The output still contains  $I$ , but the argument function `Arg` is real-valued and can be expressed in terms of `ArcTan`. With the help of a little trigonometry we now see that it differs by a constant from the one returned by `Mathematica`®.

From the point of view of efficiency, the weakest point of the above algorithm is the need to completely factor the denominator of the rational function. There is

a method called *Hermite reduction* [5] that instead of complete factorization only requires the so called “squarefree factorization”, given in Mathematica<sup>®</sup> by the function `FactorSquareFree`:

```
In[.]:= FactorSquareFree[Denominator[f[x]]]
Out[.]:= (-1 + x)2 (1 + x2)
```

See also the Wolfram<sup>™</sup> demonstration by S. Blake.<sup>7</sup> Although Hermite reduction does not require the complete factorization, still in general algebraic numbers will appear among the coefficients of the antiderivative. For example:

```
In[.]:= Integrate[1/(x3 + x + 1), x]
Out[.]:= RootSum[1 + #1 + #13 &, Log[x - #1]/(1 + 3*#12) & ]
```

The function `RootSum` is just a short notation for the expression which is obtained by applying `Normal` to it:

```
In[.]:= Normal[Integrate[1/(x3 + x + 1), x]]
Out[.]:= 
$$\frac{\text{Log}[x - \text{Root}[1 + \#1 + \#1^3 \&, 1]]}{1 + 3\text{Root}[1 + \#1 + \#1^3 \&, 1]^2} + \frac{\text{Log}[x - \text{Root}[1 + \#1 + \#1^3 \&, 2]]}{1 + 3\text{Root}[1 + \#1 + \#1^3 \&, 2]^2} + \frac{\text{Log}[x - \text{Root}[1 + \#1 + \#1^3 \&, 3]]}{1 + 3\text{Root}[1 + \#1 + \#1^3 \&, 3]^2}$$

```

```
In[.]:= N[%]
Out[.]:= (-0.20861 - 0.18382 I) Log[(-0.34116 - 1.1615 I) + x]
- (0.20861 - 0.18382 I) Log[(-0.34116 + 1.1615 I)
+ x] + 0.41723 Log[0.68232 + x]
```

### 7.2.3 Example 2: the Risch algorithm for an exponential extension

Let us now consider the situation where we have an exponential extension. Suppose, for example, we want to compute the antiderivative of  $1/(e^x + 1)$ . Mathematica<sup>®</sup> returns the answer

```
In[.]:= Integrate[1/(Ex + 1), x]
Out[.]:= x - Log[1 + Ex]
```

We introduce a new variable  $y = e^x$ , and we try to compute  $\int 1/(y + 1) dx$ , where  $dy/dx = y$ . For this situation one can prove that if an antiderivative exists, then it has the following (compatible with Liouville’s Principle) form:

$$\int \frac{P(x, y)}{Q(x, y)} dx = \frac{p(x, y)}{q(x, y)} + \sum_{i=1}^m c_i \log(q_i),$$

<sup>7</sup> Blake S. Integration using Hermite reduction. Wolfram Demonstrations Project<sup>™</sup>, published March 7 2011. <https://demonstrations.wolfram.com/IntegrationUsingHermiteReduction/>

where  $Q(x, y) = \prod_{i=1}^m Q_i^{d_i}(x, y)$ ,  $q(x, y) = \prod_{i=1}^m Q_i^{c_i}(x, y)$  with  $c_i = d_i - 1$  except when  $Q_i(x, y) = y$ , in which case  $c_i = d_i$ , and  $q_i(x, y)$  are the irreducible factors of  $Q_i(x, y)$ , not including  $y$ . We write the two-variable polynomial  $p(x, y) = \sum_i y^i p_i(x)$ , where  $p_i(x)$  is a polynomial in  $x$ .

Hence in our case we have  $Q(x, y) = y + 1$ ,  $Q_1(x, y) = q_1(x, y) = y + 1$ ,  $q(x, y) = 1$ . We take  $p(x, y) = p_0(x)$  and, hence, the antiderivative must have the form

$$\int \frac{1}{y+1} dx = p_0(x) + c \log(y(x) + 1).$$

Differentiating this we obtain (below we use  $p\theta$  in Mathematica<sup>®</sup>'s input and output cells instead of  $p_0$ ):

**In[.]:** = D[pθ[x] + c\*Log[y[x] + 1], x]

**Out[.]:** =  $\frac{c y'[x]}{1 + y[x]} + p\theta'[x]$

Since  $y'(x) = y$  we get

**In[.]:** = int = (c\*y)/(y + 1) + pθ'[x];

Next we find the unknown function  $p_0(x)$  and the coefficient  $c$  by using

**In[.]:** = SolveAlways[int == 1/(y + 1), y]

**Out[.]:** = {{pθ'[x] -> 1, c -> -1}}

(here we should interpret -> as equality). Hence,  $p_0(x) = x$ ,  $c = -1$ . We finally obtain

**In[.]:** = int = x - Log[E^x + 1]

If we repeat the same process for the antiderivative  $\int x/(e^x + 1) dx$ , we get a contradiction (because  $c$  should be constant). Hence, in this case there is no elementary antiderivative. Indeed, Mathematica<sup>®</sup> returns the answer

**In[.]:** = Integrate[x/(E^x + 1), x]

**Out[.]:** =  $\frac{x^2}{2} - x \text{Log}[1 + E^x] - \text{PolyLog}[2, -E^x]$

which involves the non-elementary function  $\text{PolyLog}$ .

We will not consider any more examples (the interested reader can consult [9] or the survey article [5] for a complete account or [8]).

## 7.2.4 Limitations of Mathematica<sup>®</sup>'s integration

Mathematica<sup>®</sup> like most other computer algebra systems does not have the full Risch algorithm implemented. For example, let us compute the derivative of an elementary function:



$$\begin{aligned} \mathbf{In}[.] &:= D[x \cdot \sin[x^{\{\text{ArcSin}[x]\}], x] \\ \mathbf{Out}[.] &:= x^{1+\text{ArcSin}[x]} \cos[x^{\text{ArcSin}[x]}] \left( \frac{\text{ArcSin}[x]}{x} + \frac{\text{Log}[x]}{\sqrt{1-x^2}} \right) \\ &\quad + \sin[x^{\text{ArcSin}[x]}] \end{aligned}$$

Now we try to find the antiderivative:

$$\begin{aligned} \mathbf{In}[.] &:= \text{Integrate}[\%, x] \\ \mathbf{Out}[.] &:= \int \left( x^{1+\text{ArcSin}[x]} \cos[x^{\text{ArcSin}[x]}] \left( \frac{\text{ArcSin}[x]}{x} - \frac{\text{Log}[x]}{\sqrt{1-x^2}} \right) + \sin[x^{\text{ArcSin}[x]}] \right) dx \end{aligned}$$

We see that in this case Mathematica<sup>®</sup> is unable to find the antiderivative although the fully implemented Risch algorithm should be able to do so.

Note that when expressed in terms of exponential, logarithmic and algebraic functions the integrand takes the form:

$$\begin{aligned} \mathbf{In}[.] &:= \text{TrigToExp}[\%] \\ \mathbf{Out}[.] &:= \frac{1}{2} \int \left( e^{-I x^{-I \text{Log}[I x + \sqrt{1-x^2}]} - e^{I x^{-I \text{Log}[I x + \sqrt{1-x^2}]} \right) + \frac{1}{2} \left( e^{-I x^{-I \text{Log}[I x + \sqrt{1-x^2}]} + e^{I x^{-I \text{Log}[I x + \sqrt{1-x^2}]} \right) \\ &\quad x^{1-I \text{Log}[I x + \sqrt{1-x^2}]} \left( \frac{\text{Log}[x]}{\sqrt{1-x^2}} - \frac{I \text{Log}[I x + \sqrt{1-x^2}]}{x} \right) \end{aligned}$$

It is easy to see that the extension needed to compute this integral is a tower of transcendental and algebraic extensions. The original Risch algorithm did not include this case and it was only worked out by Manuel Bronstein in 1987. The most complete existing implementation of the Risch algorithm was written by Manuel Bronstein and Barry Trager in the computer algebra program Axiom.<sup>8</sup>

Since the Risch algorithm is very complicated and time consuming, Mathematica<sup>®</sup> often will not attempt to use it but try to find the derivative in terms of the so called “special functions”. “Special functions” is a name used for a large variety of mathematical functions, defined by methods such as power series expansions, solutions to certain differential equations and recursion. Mathematica<sup>®</sup> can compute the values of such built-in functions to arbitrary precision and knows many identities involving them. Wolfram Research<sup>™</sup> has a web site devoted to special functions, <http://functions.wolfram.com>, where a vast amount of information about them can be found. We have already seen that Mathematica<sup>®</sup> returns special functions to many integrals for which no elementary representation exists. It also happens sometimes in cases when an elementary antiderivative could be found.

## 7.3 The Riemann integral

We will only sketch briefly the basic idea of the Riemann integral. For details we refer the reader to [14]. Consider a bounded function  $f$  on a closed interval  $[a, b]$ . A *partition* of  $[a, b]$  is a finite ordered set of points  $a = x_0 < x_1 < \dots < x_n = b$ , which

<sup>8</sup> [https://en.wikipedia.org/wiki/Axiom\\_\(computer\\_algebra\\_system\)](https://en.wikipedia.org/wiki/Axiom_(computer_algebra_system))

divides the interval  $[a, b]$  into  $n$  closed subintervals. Let  $\Delta_k = x_k - x_{k-1}$ ,  $k = 1, 2, \dots, n$ . Let  $M_k = \sup_{x \in [x_{k-1}, x_k]} f(x)$ ,  $m_k = \inf_{x \in [x_{k-1}, x_k]} f(x)$ . Then the sum  $\sum_{k=1}^n M_k \Delta_k$  is called an *upper Darboux sum* and  $\sum_{k=1}^n m_k \Delta_k$  a *lower Darboux sum* of  $f$  on  $[a, b]$ . Geometrically, an upper (lower) sum represents the area of the union of rectangles with  $[x_{k-1}, x_k]$  as base and height the maximum (minimum) value of  $f$  on  $[x_{k-1}, x_k]$ . Obviously any lower Darboux sum is smaller than any upper Darboux sum. Note also that when we replace a partition by a subpartition (in other words, we subdivide some intervals) an upper sum will decrease and a lower sum will increase. Let  $U(f)$  ( $L(f)$ ) denote the infimum (supremum) of the set of all upper (lower) Darboux sums. This is called the upper (lower) *Darboux integral*. When  $U(f) = L(f)$  we say that the function  $f$  is *Darboux integrable* and the common value  $U(f) = L(f)$  is called the *Darboux (or Riemann) integral* and is denoted by  $\int_a^b f(x) dx$ .

```
In[.]:= f[x_] := Sin[8*x]/2 + 1/2
```

```
In[.]:= areaMax[f_, x_][a_, b_] := First[Maximize[{f[x],
a <= x <= b}, x]]*(b - a)
```

```
In[.]:= areaMin[f_, x_][a_, b_] := First[Minimize[{f[x],
a <= x <= b}, x]]*(b - a)
```

```
In[.]:= max[f_, x_][a_, b_] := First[Maximize[{f[x], a <=
x <= b}, x]]
```

```
In[.]:= min[f_, x_][a_, b_] := First[Minimize[{f[x], a <=
x <= b}, x]]
```

```
In[.]:= Manipulate[Module[{ll, maxs, mins, maxArea, minArea,
maxRectangles, minRectangles, g2 = Plot[f[x],
{x, 0, 1}]}], BlockRandom[SeedRandom[rr]; ll =
Partition[Join[{0}, Sort[RandomReal[{0, 1}, {m}]],
{1}], 2, 1]; maxs = Apply[max[f, x], ll, {1}];
mins = Apply[min[f, x], ll, {1}];
maxArea = maxs . Abs[Apply[Subtract, ll, {1}]];
minArea = mins . Abs[Apply[Subtract, ll, {1}]];
maxRectangles = Table[Rectangle[ll[[i, 1]], 0],
{ll[[i, 2]], maxs[[i]]}], {i, 1, Length[ll]}];
minRectangles = Table[Rectangle[ll[[i, 1]], 0],
{ll[[i, 2]], mins[[i]]}], {i, 1, Length[ll]}];
Show[Graphics[{Text[StringJoin["max area = ",
ToString[maxArea]], {0.6, 0.6}], Text[StringJoin[
"min area = ", ToString[minArea]], {0.6, 0.5}], Red,
Opacity[0.2], maxRectangles, Blue, minRectangles}],
g2]], {{m, 8, "number of points"}, 8, 30, 1,
ControlType -> PopupMenu}, {{rr, 0, ""},
Button["new random partition", rr =
RandomInteger[2^64 - 1]] & }, SaveDefinitions -> True]
```

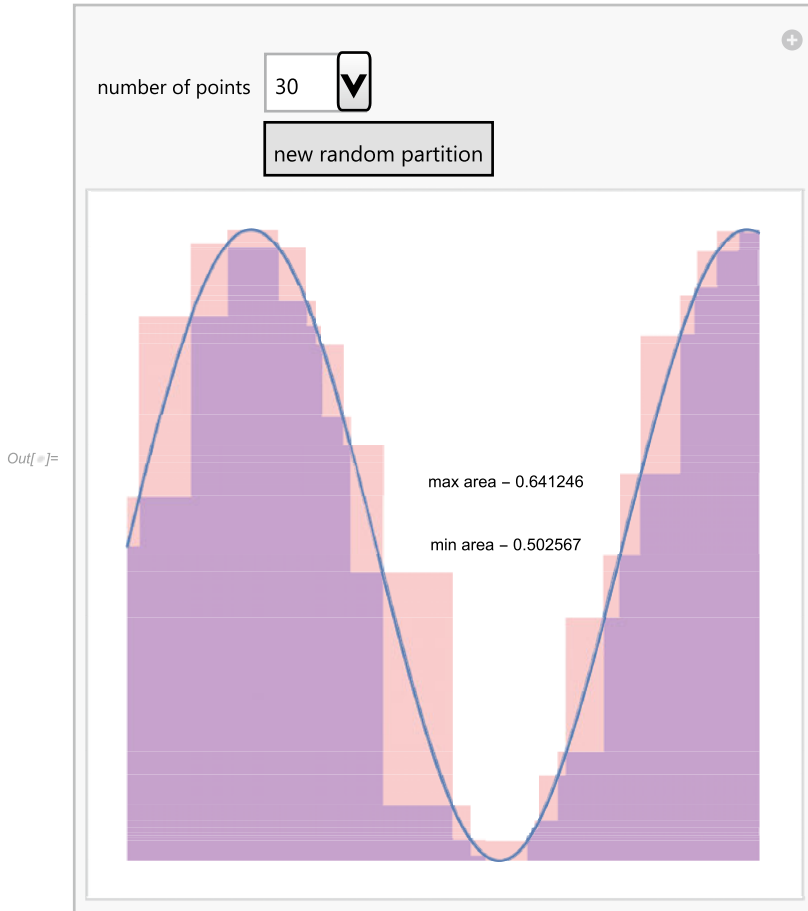


Figure 7.3

The illustration above shows upper and lower Darboux sums (pink and violet rectangles) for a continuous function and a random choice of the partition. `SeedRandom` is used to generate a new random sample. `BlockRandom` assures that only the extra points are generated when we increase the number of points of the partition.

Note that if the function is not continuous, then the supremum and infimum values may not be attained on the intervals of the partition. There is an alternative approach to integration due to Riemann. In this approach for each partition  $\mathcal{P}$ ,  $a = x_0 < x_1 < \dots < x_n = b$ , we choose points  $\bar{x}_1, \dots, \bar{x}_n$  with  $\bar{x}_k \in [x_{k-1}, x_k]$  for  $k = 1, 2, \dots, n$  and consider sums  $S_n = \sum_{k=1}^n f(\bar{x}_k) \Delta_k$ . Clearly this (Riemann) sum always lies between the corresponding Darboux sums. For any  $\mathcal{P}$  the norm  $\|\mathcal{P}\|$  is defined as the maximum  $\max_{1 \leq k \leq n} \Delta_k$ . A function  $f$  is said to be *Riemann integrable* if and only if there exists a number  $\mathcal{I}$  with the following property: for each  $\varepsilon > 0$  there is a  $\delta > 0$  such that for ev-

ery Riemann sum  $\sigma$  associated with a partition  $\mathcal{P}$  with  $\|\mathcal{P}\| < \delta$  one has  $|\sigma - \mathcal{I}| < \varepsilon$ . The number  $\mathcal{I}$  is called the *Riemann integral* of  $f$  on  $[a, b]$ . One can show that  $\mathcal{I}$  is independent of the particular choice of partition and subinterval points  $\bar{x}_k$  and the concepts of Darboux integral and Riemann integral coincide (see [14, Chapter 6]). From now we shall generally speak only of Riemann integral and Riemann integrability (or just integrability).

The Darboux definition is particularly useful for determining which functions are integrable while the Riemann approach (as we shall soon see) is more useful in concrete computations. With the help of the Darboux definition it is easy to show that all continuous functions and also bounded functions that are continuous except for a countable number of jump discontinuities are integrable (see [14, p. 221] and [14, Theorem 6.21, p. 231]). There are also functions that can be easily shown not to be Riemann integrable, e. g., the Dirichlet function, which is defined by

```
In[.] := f[x_] := Piecewise[{{1, Element[x, Rationals]}}]
In[.] := f /@ {Sqrt[2], Pi, 1/2, E}
Out[.] := {0, 0, 1, 0}
```

We immediately see that the function is not Riemann integrable. Indeed, let  $x_0, \dots, x_n$  be any partition of  $[0, 1]$ . Since every interval contains both rationals and irrationals, we see that  $L(f) = 0$  and  $U(f) = 1$ . The function is thus not Riemann integrable. Nevertheless, `Integrate` gives

```
In[.] := Integrate[f[x], {x, 0, 1}]
Out[.] := 0
```

This is, indeed, the correct answer but it needs a different and more powerful concept of integral, the so called Lebesgue integral. The Dirichlet function is not Riemann integrable but it is Lebesgue integrable and `Integrate` returns the result of the Lebesgue integral. (All functions that are Riemann integrable are also Lebesgue integrable and the corresponding integrals are the same.)

The Darboux definition of the integral can also be used to prove the basic properties of integrals. These follow in a straightforward manner from properties of sums and limits. Here we only state them (for the proofs see [14, Theorems 6.25, 6.26]).

**Theorem 26** (General properties of the definite integral). *Let  $f$  and  $g$  be integrable on  $[a, b]$ .*

1. *If  $c_1$  and  $c_2$  are constants, then the functions  $c_1f + c_2g$  are integrable and*

$$\int_a^b (c_1f(x) + c_2g(x)) dx = c_1 \int_a^b f(x) dx + c_2 \int_a^b g(x) dx.$$

2. *If  $f(x) \leq g(x)$  on  $[a, b]$ , then*

$$\int_a^b f(x) dx \leq \int_a^b g(x) dx.$$

3. Assume that  $f$  is bounded on  $[a, b]$  and  $c \in [a, b]$ . Then  $f$  is integrable on  $[a, b]$  if and only if  $f$  is integrable on  $[a, c]$  and  $[c, b]$  and

$$\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx.$$

4. The functions  $fg$  and  $|f|$  are also integrable on  $[a, b]$  and the following inequality holds:

$$\left| \int_a^b f(x) dx \right| \leq \int_a^b |f(x)| dx.$$

The notions of the definite and indefinite integrals are related by one of the most famous results in mathematics, known as the *Fundamental Theorem of Calculus*. In fact, this theorem is often presented in the form of two theorems, called the First and the Second Fundamental Theorem of Calculus.

**Theorem 27** (The First Fundamental Theorem of Calculus, the Newton–Leibniz formula). *If  $f$  is integrable on  $[a, b]$  and  $F$  is an antiderivative of  $f$  on  $[a, b]$ , then*

$$\int_a^b f(x) dx = F(b) - F(a).$$

Note that here we use the word antiderivative in the strict mathematical sense, which means that  $F$  has to be a differentiable and therefore continuous function on  $[a, b]$ . Recall that the Risch algorithm sometimes returns an “antiderivative” that may have points of discontinuity. Nevertheless, Mathematica<sup>®</sup> generally computes definite integrals correctly (see the example and discussion at the end of Section 7.1).

The First Fundamental Theorem of Calculus can be thought of as saying that integration is a left inverse of differentiation. To see this let us replace  $b$  by  $x$  and consider  $\int_a^x f(t) dt$  as a function of  $x$ . The theorem now says

$$\int_a^x \frac{dF(t)}{dt} = F(x) - F(a).$$

In other words, integrating the derivative of a function returns the same function (up to a constant). The Second Fundamental Theorem of Calculus says that integration is also the right inverse of differentiation.

**Theorem 28** (The Second Fundamental Theorem of Calculus). *Let  $f$  be an integrable function on  $[a, b]$ . We define*

$$F(x) = \int_a^x f(t) dt$$

*for  $a \leq x \leq b$ . Then  $F$  is continuous on  $[a, b]$  and differentiable at every point  $c$  at which  $f$  is continuous. In this case  $F'(c) = f(c)$ .*

Note that this gives us a way to construct an antiderivative of any continuous function on  $[a, b]$ . Mathematica® actually “knows” this theorem:

```
In[.]:= Clear[f]
In[.]:= D[Integrate[f[t], {t, a, x}], x]
Out[.]:= f[x]
```

Note, however, that if we try to do the same thing with a defined function, the evaluation may take much longer and the answer can be complicated:

```
In[.]:= f[x_] := 1/(x^3 + 1)
In[.]:= D[Integrate[f[t], {t, 1, x}], x]
Out[.]:= ConditionalExpression[
  
$$\frac{1}{9} \left( -\frac{3}{\left(1 + \frac{1}{x}\right)x^2} - \frac{3(-1)^{2/3}}{\left(1 - \frac{(-1)^{1/3}}{x}\right)x^2} + \frac{3(-1)^{2/3}x\left(\frac{1}{x} - \frac{(-1)^{2/3}+x}{x^2}\right)}{(-1)^{2/3} + x} \right), \text{Re}[x] > 1 \ \&\& \ \text{Im}[x] == 0]$$

```

We will explain why we get this kind of answer in the next subsection. Using Assumptions makes the answer simpler:

```
In[.]:= D[Integrate[f[t], {t, 1, x}, Assumptions -> x > 1], x]
Out[.]:=  $\frac{1}{18} \left( \frac{6}{1+x} - \frac{3(-1+2x)}{1+(-1+x)x} + \frac{12}{1+\frac{1}{3}(-1+2x)^2} \right)$ 
In[.]:= % // FullSimplify
Out[.]:=  $\frac{1}{1+x^3}$ 
```

However, if we define  $f$  to be a function whose antiderivative Mathematica® does not know, we get the answer quite quickly:

```
In[.]:= f[x_] := x^x
In[.]:= D[Integrate[f[t], {t, 1, x}], x]
Out[.]:= x^x
```

The difference is that in the first case Mathematica® first found a complicated antiderivative using the Risch algorithm and then tried to differentiate it, resulting in a complicated answer (made even more complicated by Mathematica®’s attempt to give conditions for its validity), and in the second case, Mathematica® quickly decided that it could not find an antiderivative so it applied the Second Fundamental Theorem of Calculus to the original input and quickly returned the answer. Hence if we wanted Mathematica® to not evaluate the integral but to use the Second Fundamental Theorem of Calculus, we could do this:

```
In[.]:= f[x_] := 1/(x^3 + 1)
In[.]:= Block[{f}, D[Integrate[f[t], {t, a, x}], x]]
Out[.]:=  $\frac{1}{1+x^3}$ 
```

Using `Block` in this way turns  $f$  temporarily into an undefined function; `Mathematica`<sup>®</sup> then applies the Second Fundamental Theorem and finally replaces the symbol  $f$  by its definition.

### 7.3.1 Using `Integrate` and `NIntegrate` with definite integrals

Here we will try to explain briefly various kinds of definite integrals that appear in `Mathematica`<sup>®</sup>. These are obtained by using the function `Integrate` with various kinds of limits (symbolic or numerical) and possibly with assumptions. (There are also numerical integrals that can be computed with `NIntegrate` which we will briefly discuss below.)

The most general type of definite integral in `Mathematica`<sup>®</sup> has the form `Integrate[f[x], {x, a, b}]`, where at least one of the limits  $a$  and  $b$  is symbolic. In this case `Mathematica`<sup>®</sup> assumes that the integral is a path integral computed in the complex plane along a straight line joining the points corresponding to the complex numbers  $a$  and  $b$ . `Mathematica`<sup>®</sup> then tries to find the most general complex antiderivative and determine the region of values of  $a$  and  $b$  in which the formula for the antiderivative is valid. This problem is difficult and `Mathematica`<sup>®</sup> generally returns a suboptimal region where the answer is correct. Such formulas can be very complicated. The reader may consider, for example, the following simple definite integral for which `Mathematica`<sup>®</sup> gives a very complicated looking answer (which we omit):

```
In[ ]:= Integrate[1/x, {x, a, b}]
```

There are two ways to avoid this issue. We can instruct `Mathematica`<sup>®</sup> that we do not want any conditions (which may cause problems when we use the formula):

```
In[ ]:= Integrate[1/x, {x, a, b}, GenerateConditions -> False]
Out[ ]:= -Log[a] + Log[b]
```

or we can specify conditions on  $a$  and  $b$  by using `Assumptions`:

```
In[ ]:= Integrate[1/x, {x, a, b}, Assumptions -> {b > a > 0}]
Out[ ]:= Log[ $\frac{b}{a}$ ]
```

When the limits are numerical and exact, an exact numerical answer will be given if `Mathematica`<sup>®</sup> is able to compute it:

```
In[ ]:= Integrate[1/Sqrt[4 - x^2], {x, -1, 1}]
Out[ ]:=  $\frac{\pi}{3}$ 
```

Finally, there is `NIntegrate`, a powerful function that performs numerical integration using many methods (including Riemann sums), controlled by the option `Method`. This function can compute values of integrals that cannot be found by the exact method of `Integrate` to arbitrary precision. For example,

```
In[.]:= Integrate[1/Log[4 + x^3], {x, -1, 1}]
```

$$\mathbf{Out[.]}$$

$$:= \int_{-1}^1 \frac{1}{4 + x^3} dx$$

```
In[.]:= NIntegrate[1/Log[4 + x^3], {x, -1, 1},
WorkingPrecision -> 20]
```

```
Out[.]:= 1.4547429629288028678
```

We can also do this:

```
In[.]:= N[Integrate[1/Log[4 + x^3], {x, -1, 1}]]
```

```
Out[.]:= 1.45474
```

However, the reader should be aware of the fact that this answer was not obtained by `Integrate`. In fact, `Integrate` could not find the answer and `Mathematica`<sup>®</sup> passed the problem to `NIntegrate`. In the examples below two different methods are used to obtain the same answer and therefore, they can be used to check correctness of the answer:

```
In[.]:= N[Integrate[x^2, {x, -1, 1}]]
```

```
Out[.]:= 0.666667
```

```
In[.]:= NIntegrate[x^2, {x, -1, 1}]
```

```
Out[.]:= 0.666667
```

`NIntegrate` is a very reliable function. It is extremely rare for it to return wrong answers, something that happens much more frequently for `Integrate`.

### 7.3.2 Riemann sums

Recall that by a *Riemann sum* of a function  $f$  defined on  $[a, b]$  we mean a sum of the form  $\sum_{k=1}^n f(\bar{x}_k) \Delta_k$ . The point  $\bar{x}_k$  lies in the interval  $[x_{k-1}, x_k]$ , with the most common choices being  $\bar{x}_k = x_{k-1}$ ,  $\bar{x}_k = x_k$  and  $\bar{x}_k = (x_{k-1} + x_k)/2$ . If  $f$  is integrable, then we can compute its integral by choosing a sequence of partitions  $\mathcal{P}_k$  with  $\|\mathcal{P}_k\| \rightarrow 0$  and finding the limit. The most natural “partitioning scheme” is simply division into intervals of equal length, in which case the condition  $\|\mathcal{P}_k\| \rightarrow 0$  is always satisfied. The integral takes the form  $\lim_{n \rightarrow \infty} (\sum_{k=1}^n (b-a)f(\bar{x}_k)/n)$ . Thus this method reduces computing definite integrals to computing limits of Riemann sums. Usually, however, it is easier to compute integrals by using the First Fundamental Theorem of Calculus. In fact we can sometimes reverse the process and compute limits of Riemann sums by finding antiderivatives.

#### 7.3.2.1 Example

Compute  $\lim_{n \rightarrow \infty} (\sum_{k=n+1}^{2n} 1/k)$ .



First note that Mathematica<sup>®</sup> can compute this limit directly:

```
In[.]:= Limit[Sum[1/k, {k, n + 1, 2*n}], n -> Infinity]
Out[.]:= Log[2]
```

```
In[.]:= Sum[1/k, {k, n + 1, 2*n}]
Out[.]:= -PolyGamma[0, 1 + n] + PolyGamma[0, 1 + 2 n]
```

Mathematica<sup>®</sup> does this by expressing the sum in terms of special functions and then computing the limit by using its extensive knowledge of such functions. We will find it easier, however, to note that the sum  $\sum_{k=n+1}^{2n} 1/k$  can be written in the form

$$\sum_{k=1}^n \frac{1}{k+n} = \sum_{k=1}^n \frac{1}{n} \frac{1}{1+k/n}.$$

If we divide the interval  $[0, 1]$  into  $n$  equal parts and take  $\bar{x}_k = x_{k-1} = k/n$ , we see that we have the  $n$ -th Riemann sum of the function  $f(x) = 1/(1+x)$  on  $[0, 1]$ . Hence the limit is

```
In[.]:= Integrate[1/(1 + x), {x, 0, 1}]
Out[.]:= Log[2]
```

### 7.3.2.2 Example

Sums whose limits as  $n \rightarrow \infty$  are integrals (Riemann sums) have to have a special form. However, the method of Riemann sums works also for certain sums which are not actually Riemann sums but are “close” to them. Here is one example. Suppose we wish to compute the limit

$$\lim_{n \rightarrow \infty} \left( \sum_{k=1}^n \frac{n}{8kn + 2k + 5n^2} \right).$$

Mathematica<sup>®</sup> can do it directly:

```
In[.]:= Limit[Sum[n/(8*k*n + 2*k + 5*n^2), {k, 1, n}],
n -> Infinity]
Out[.]:= 1/8 Log[13/5]
```

Inspecting the summand we see that we are not dealing with a Riemann sum. However, if we remove the summand  $2k$  from the denominator, we get a Riemann sum, which can be evaluated by integration:

$$\lim_{n \rightarrow \infty} \left( \sum_{k=1}^n \frac{n}{8kn + 5n^2} \right) = \lim_{n \rightarrow \infty} \left( \frac{1}{n} \sum_{k=1}^n \frac{1}{8k/n + 5} \right).$$

This is

```
In[.]:= Integrate[1/(5 + 8*x), {x, 0, 1}]
Out[.]:= 1/8 Log[13/5]
```

We obtain the same answer. Let us try to prove that the two limits are the same without computing them. This is equivalent to

$$\mathbf{In[.]} := \text{Limit}[\text{Sum}[n/(8*k*n + 2*k + 5*n^2) - n/(8*k*n + 5*n^2), \{k, 1, n\}], n \rightarrow \text{Infinity}]$$

$$\mathbf{Out[.]} := 0$$

Consider the sum

$$\sum_{k=1}^n \left( \frac{n}{8kn + 2k + 5n^2} - \frac{n}{8kn + 5n^2} \right).$$

The summand is

$$\mathbf{In[.]} := \text{Together}[n/(8*k*n + 2*k + 5*n^2) - n/(8*k*n + 5*n^2)]$$

$$\mathbf{Out[.]} := -\frac{2k}{(8k + 5n)(2k + 8kn + 5n^2)}$$

So we need to show that the limit of the sum

$$\sum_{k=1}^n \frac{2k}{(8k + 5n)(8kn + 2k + 5n^2)}$$

is 0 as  $n \rightarrow \infty$ . This sum is clearly smaller than

$$\frac{2n^2}{(5n)(5n^2)} = \frac{2}{25n},$$

which tends to 0 as  $n \rightarrow \infty$ . One can generalize this argument to other sums which are asymptotic Riemann sums.

## 7.4 Improper integrals

The definite integrals we have considered so far have all involved bounded functions defined on closed intervals. However, in many applications these kinds of integrals are insufficient: sometimes we want to integrate functions which are unbounded or where the domain over which we want to integrate is infinite. Such integrals can be defined by a simple extension of Riemann integration and are known as *improper integrals*. Improper integrals come in two types (kinds), known as the first type and the second type.

### 7.4.1 Integrals over infinite intervals (improper integrals of the first type)

The first type of integrals are integrals defined over infinite intervals, for example:

$$\mathbf{In[.]} := \text{Integrate}[\text{Log}[x]/x^3, \{x, 1, \text{Infinity}\}]$$

$$\mathbf{Out[.]} := \frac{1}{4}$$

```
In[.] := Integrate[Exp[x], {x, -Infinity, 1}]
```

```
Out[.] := E
```

```
In[.] := Integrate[x/E^x^2, {x, -Infinity, Infinity}]
```

```
Out[.] := 0
```

The first and second integrals above (with only one infinite limit) are defined as limits of ordinary Riemann integrals, e. g.,

$$\int_1^{\infty} \frac{\log x}{x^3} dx = \lim_{t \rightarrow \infty} \int_1^t \frac{\log x}{x^3} dx,$$

$$\int_{-\infty}^1 e^x dx = \lim_{t \rightarrow -\infty} \int_t^1 e^x dx.$$

They can be computed by Mathematica<sup>®</sup> by means of these definitions, e. g.,

```
In[.] := Limit[Integrate[Log[x]/x^3, {x, 1, t}], t -> Infinity]
```

```
Out[.] := 1/4
```

but this may take much longer than using infinity directly in the limit, since Mathematica<sup>®</sup> first attempts to compute an integral with a symbolic limit. Compare, for example,

```
In[.] := Integrate[1/(x^3 + 1), {x, 1, Infinity}]
```

```
Out[.] := 1/9 (sqrt(3) pi - Log[8])
```

with

```
In[.] := Limit[Integrate[1/(x^3 + 1), {x, 1, t}], t -> Infinity]
```

```
Out[.] := $Aborted
```

If we want to use the second method, it is much better to use Assumptions:

```
In[.] := Limit[Integrate[1/(x^3 + 1), {x, 1, t},  
Assumptions -> {t > 1}], t -> Infinity]
```

```
Out[.] := 1/9 (sqrt(3) pi - Log[8])
```

Although generally it is preferable to use the first approach (infinite limits), like in the case of infinite sums, the two methods can return different answers when the integrals are not convergent:

```
In[.] := Integrate[1/x, {x, 1, Infinity}]
```

```
... Integrate: Integral of 1/x does not converge on {1, infinity}.
```

```
Out[.] := Integrate[1/x, {x, 1, infinity}]
```

```
In[.] := Limit[Integrate[1/x, {x, 1, t}, Assumptions ->  
{t > 1}], t -> Infinity]
```

```
Out[.] := infinity
```

The following integral is defined as the sum of two integrals:

$$\int_{-\infty}^{\infty} f(x) dx = \int_{-\infty}^c f(x) dx + \int_c^{\infty} f(x) dx,$$

where  $c$  is any number. It is easy to show that the answer is always independent of the choice of  $c$ . Thus,

```
In[.]:= Integrate[x/E^x^2, {x, -Infinity, 1}] +
      Integrate[x/E^x^2, {x, 1, Infinity}]
```

```
Out[.]:= 0
```

There are also situations when the two integrals  $\int_{-\infty}^c f(x) dx$  and  $\int_c^{\infty} f(x) dx$  do not exist, but the symmetric limit  $\lim_{t \rightarrow \infty} \int_{-t}^t f(x) dx$  exists. In such cases we call the value of this limit the *Cauchy principal value of the integral* (the integral is still considered divergent). One can compute the Cauchy principal value in **Mathematica**<sup>®</sup> using the option `PrincipalValue`:

```
In[.]:= Integrate[x^3, {x, -Infinity, Infinity}]
      ... Integrate: Integral of x^3 does not converge on {-∞,∞}.
```

```
Out[.]:=  $\int_{-\infty}^{\infty} x^3 dx$ 
```

```
In[.]:= Integrate[x^3, {x, -Infinity, Infinity},
      PrincipalValue -> True]
```

```
Out[.]:= 0
```

## 7.4.2 Improper integrals of the first type and infinite sums

There is both an analogy and a relationship between improper integrals of the first type and infinite sums. All basic convergence tests for series, such as the comparison test, the limit comparison test and the Dirichlet test, have their analogues for improper integrals of the first type (see [14, Theorem 7.5] for the comparison test and [14, Theorem 7.17] for the limit comparison test). The Dirichlet test for integrals is as follows.

**Theorem 29** (Dirichlet test for integrals). *If  $f, g : [a, \infty) \rightarrow \mathbb{R}$ ,  $g$  is decreasing with  $\lim_{x \rightarrow \infty} g(x) = 0$ , for every  $r \in [a, \infty)$   $f$  is Riemann integrable on  $[a, r]$  and there exists  $M \in \mathbb{R}$  such that  $|\int_a^r f(x) dx| < M$  for all  $r \in [a, \infty)$ , then  $\int_a^{\infty} f(x)g(x) dx$  converges.*

For example, this test shows that the integrals  $\int_1^{\infty} (\sin x)/x dx$  and  $\int_1^{\infty} (\cos x)/x^{3/2} dx$  are convergent. Unlike in the case of sums, **Mathematica**<sup>®</sup> does not have a functions for testing convergence of integrals but it performs a test of convergence every time an improper integral is computed:

```
In[.]:= Integrate[Sin[x]/x, {x, 1, Infinity}]
```

```
Out[.]:=  $\frac{1}{2} (\pi - 2 \text{SinIntegral}[1])$ 
```

```
In[.]:= Integrate[1/Log[x], {x, 2, Infinity}]
... Integrate: Integral of  $\frac{1}{\text{Log}[x]}$  does not converge on {2,∞}.
```

$$\mathbf{Out[.]}$$

$$:= \int_2^{\infty} \frac{1}{\text{Log}[x]} dx$$

Note that Mathematica<sup>®</sup> was able to compute the first integral above in terms of a value of a special function. We can also compute the integral using NIntegrate and compare the answers (as mentioned before they are computed in a different way):

```
In[.]:= NumberForm[N[Integrate[Sin[x]/x, {x, 1,
Infinity}], 11], 10]
```

```
Out[.]//NumberForm= 0.6247132564
```

```
In[.]:= NumberForm[NIntegrate[Sin[x]/x, {x, 1,
Infinity}, PrecisionGoal -> 11,
WorkingPrecision -> 11], 10]
```

```
Out[.]//NumberForm= 0.6247132564
```

(NumberForm tells Mathematica<sup>®</sup> how many digits to show on the screen.)

Just like for sums, there is also the concept of absolute convergence for integrals, which is stronger than ordinary convergence. Mathematica<sup>®</sup> often finds absolute convergence difficult to deal with; for example, it is obvious that  $\int_1^{\infty} |\sin x|/x^2 dx$  is convergent by the comparison test with the following integral:

```
In[.]:= Integrate[1/x^2, {x, 1, Infinity}]
```

```
Out[.]:= 1
```

The following computation, however, does not end in a reasonable time:

```
In[.]:= Integrate[Abs[Sin[x]]/x^2, {x, 1, Infinity}]
```

```
Out[.]:= $Aborted
```

On the other hand, this is computed quickly:

```
In[.]:= Integrate[Sin[x]/x^2, {x, 1, Infinity}]
```

```
Out[.]:= -CosIntegral[1] + Sin[1]
```

In addition to the analogy between infinite series and improper integrals of the first type, there is also a more direct relationship.

**Theorem 30** (The integral test [14, Theorem 7.25]). *Suppose that  $f$  is a non-negative, continuous and decreasing function of  $x$  for  $x \geq 1$ . Then  $\sum_{k=1}^{\infty} f(k)$  is convergent if and only if  $\int_1^{\infty} f(x) dx$  is convergent. Moreover, when the series converges*

$$\int_1^{\infty} f(x) dx \leq \sum_{k=1}^{\infty} f(k) \leq \int_1^{\infty} f(x) dx + f(1)$$

or, equivalently,

$$\sum_{k=2}^{\infty} f(k) \leq \int_1^{\infty} f(x) dx \leq \sum_{k=1}^{\infty} f(k).$$

A rigorous proof is given in many textbooks (see, for example, [14, p. 300]) but the basic idea is based on the following picture which shows that the area under the graph of  $f$  over a finite interval lies between the two sums of areas of the rectangles, given by the two Riemann sums (which in this case coincide with Darboux sums).

```
In[.]:= Manipulate[Module[{f = 1/#1^2 & , g1, g2, g},
  g = Plot[f[x], {x, 1, m}, PlotRange -> {{1, m},
    {0, 0.3}}]; g1 = Graphics[Table[{Opacity[0.2], Blue,
  Rectangle[{n, 0}, {n + 1, f[n + 1]}]}, {n, 1, m}]];
  g2 = Graphics[Table[{Opacity[0.2], Red,
  Rectangle[{n, 0}, {n + 1, f[n]}]}, {n, 1, m}]];
  Show[g, g1, g2, PlotRange -> {{1, m}, {0, 0.3}},
  AxesOrigin -> {0, 0}], {m, 10, "m"}, 5, 20, 1,
  Appearance -> "Labeled"]
```

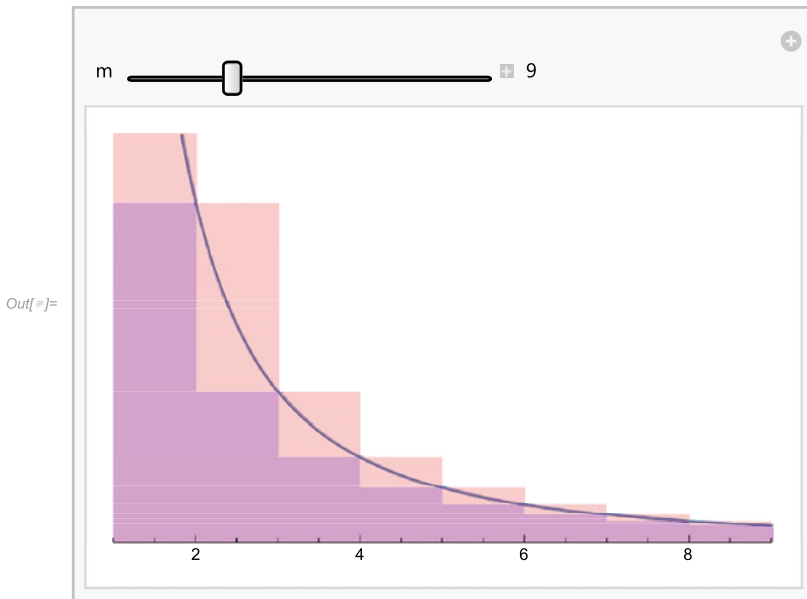


Figure 7.4

Since in general it is easier to test integrals for convergence than series, this theorem is most often used for testing the convergence of infinite series by reducing it to the question of convergence of improper integrals (rather than the other way round). In

fact, this test is exactly the `IntegralTest`, which is a value of the option `Method` to the function `SumConvergence` discussed earlier (See Section 3.4).

### 7.4.2.1 Example

Test for convergence the series  $\sum_{n=2}^{\infty} 1/(n \log^a(n))$ .

The integral test reduces the problem to the convergence of the integral  $\int_2^{\infty} 1/(x \log^a(x)) dx$ . We could easily solve it by using the substitution  $u = \log(x)$ . In Mathematica<sup>®</sup>

```
In[.]:= Integrate[1/(x*Log[x]^a), {x, 2, t}, Assumptions
-> {Element[a, Reals], t > 2}]
```

```
Out[.]:=  $\frac{\text{Log}[2]^{1-a} - \text{Log}[t]^{1-a}}{-1 + a}$ 
```

This is, of course, convergent if and only if  $a > 1$  since  $\log(t)^{1-a} \rightarrow 0$  as  $t \rightarrow \infty$ .

### 7.4.2.2 Example

Compute

$$\lim_{x \rightarrow 0^+} \left( \sum_{n=1}^{\infty} \frac{\sin x}{(nx)^4 + 4} \right).$$

Mathematica<sup>®</sup> gives the answer quickly:

```
In[.]:= Limit[Sum[Sin[x]/(4 + (n*x)^4), {n, 1, Infinity}],
x -> 0, Direction -> "FromAbove"]
```

```
Out[.]:=  $\frac{\pi}{8}$ 
```

but in order to do so it first computes a sum:

```
In[.]:= Sum[Sin[x]/(4 + (n*x)^4), {n, 1, Infinity}]
```

```
Out[.]:=  $\frac{1}{x} \left( \frac{1}{16} + \frac{1}{16} \right) \left( (-1 + i)x + \pi \text{Cot} \left[ \frac{(1 + i)\pi}{x} \right] + \pi \text{Coth} \left[ \frac{(1 + i)\pi}{x} \right] \right) \text{Sin}[x]$ 
```

We can solve the problem by using only integration by applying the first inequality in the above theorem according to which the sum  $\sum_{n=1}^{\infty} (\sin x)/((nx)^4 + 4)$  must lie between two expressions

$$\sin x \int_1^{\infty} \frac{1}{(nx)^4 + 4} dn$$

and

$$\sin x \int_1^{\infty} \frac{1}{(nx)^4 + 4} dn + \frac{\sin x}{x^4 + 4}.$$

Since

**In[.]**:= Limit[Sin[x]/(x^4 + 4), x -> 0]

**Out[.]**:= 0

we have

$$\lim_{x \rightarrow 0^+} \left( \sum_{n=1}^{\infty} \frac{\sin x}{(nx)^4 + 4} \right) = \lim_{x \rightarrow 0^+} \left( \int_1^{\infty} \frac{\sin x}{(nx)^4 + 4} dn \right).$$

It remains to compute  $\int_1^{\infty} 1/((nx)^4 + 4) dn$ , which can be done either by hand or in Mathematica® as follows:

**In[.]**:= Integrate[1/((n\*x)^4 + 4), {n, 1, Infinity},  
Assumptions -> x > 0]

**Out[.]**:=  $\frac{\pi + \text{ArcTan}[1 - x] - \text{ArcTan}[1 + x] - \text{ArcTanh}\left[\frac{2x}{2+x^2}\right]}{8x}$

**In[.]**:= Limit[Sin[x]\*%, x -> 0, Direction -> "FromAbove"]

**Out[.]**:=  $\frac{\pi}{8}$

### 7.4.3 Integrals of unbounded functions (improper integrals of the second type)

The theory of improper integrals of the second type is very similar to the theory of improper integrals of the first type. These involve integrals of functions which have singularities at some points in their domains, near which the values tend to  $\infty$  or  $-\infty$ . We start with functions which have a singularity at an endpoint. Consider for example

**In[.]**:= Integrate[Log[x]^2, {x, 0, 1}]

**Out[.]**:= 2

The function  $\log^2(x)$  is not defined at 0 and tends to  $\infty$  as  $x \rightarrow 0$ . The integral is defined as

**In[.]**:= Limit[Integrate[Log[x]^2, {x, t, 1}, Assumptions ->  
{0 < t < 1}], t -> 0]

**Out[.]**:= 2

Improper integrals with a singularity at the right endpoint are defined similarly. When we have singularities only at both endpoints, we choose any point inside and represent the integral as the sum of two integrals:

$$\int_0^1 \frac{\log(x)}{x-1} dx = \left( \int_0^{1/2} + \int_{1/2}^1 \right) \frac{\log(x)}{x-1} dx = \lim_{t \rightarrow 0} \int_t^{1/2} \frac{\log(x)}{x-1} dx + \lim_{s \rightarrow 1} \int_{1/2}^s \frac{\log(x)}{x-1} dx.$$



**In[.]**:= Limit[Integrate[Log[x]/(x - 1), {x, t, 1/2},  
Assumptions -> {t > 0}], t -> 0]

**Out[.]**:=  $\frac{1}{12} (\pi^2 + 6 \text{Log}[2]^2)$

**In[.]**:= Limit[Integrate[Log[x]/(x - 1), {x, 1/2, s},  
Assumptions -> {1/2 < s < 1}], s -> 1]

**Out[.]**:=  $\frac{1}{12} (\pi^2 - 6 \text{Log}[2]^2)$

**In[.]**:= Integrate[Log[x]/(x - 1), {x, 0, 1}]

**Out[.]**:=  $\frac{\pi^2}{6}$

There are also improper integrals with singularities inside the domain, e. g.,

**In[.]**:= Integrate[1/(x - 2), {x, 0, 3}]

... Integrate: Integral of  $\frac{1}{-1+x}$  does not converge on {0, 3}.

**Out[.]**:=  $\int_0^3 \frac{1}{-2+x} dx$

This integral is defined as the sum of two integrals over [0, 2) and (2, 3] but they are both not convergent. In such cases we can try the principal value approach:

**In[.]**:= Integrate[1/(x - 2), {x, 0, 3}, PrincipalValue -> True]

**Out[.]**:= -Log[2]

Finally let us note that improper integrals of the second type can often be converted to improper integrals of the first type by a suitable substitution. For example, consider

**In[.]**:= Integrate[Log[x]^2, {x, 0, 1}]

**Out[.]**:= 2

Using the substitution  $y = 1/x$ , we see that  $dx = -dy/y^2$  and the integral transforms into an integral of the first type:

**In[.]**:= Integrate[Log[y]^2/y^2, {y, 1, Infinity}]

**Out[.]**:= 2



# Bibliography

- [1] Apostol T. Calculus. Volume 1: one-variable calculus, with an introduction to linear algebra. 2nd edition, John Wiley & Sons, Inc. New York, 1967.
- [2] Balser W. From divergent power series to analytic functions. Theory and application of multisummable power series. Lecture Notes in Mathematics, 1582, Springer-Verlag, Berlin, 1994.
- [3] Boas RP, Boas HP. A primer of real functions. Mathematical Association of America Textbooks 13, The Carus Mathematical Monographs, The Mathematical Association of America; 4th edition, 1996.
- [4] Bridger M. Real analysis. A constructive approach. Pure and Applied Mathematics, New York [John Wiley & Sons], Wiley-Interscience, Hoboken, NJ, 2007.
- [5] Bronstein M. Symbolic integration tutorial. Course notes of an ISSAC'98 tutorial. Available at [www.sop.inria.fr/cafe/Manuel.Bronstein/publications](http://www.sop.inria.fr/cafe/Manuel.Bronstein/publications).
- [6] Bronstein M. Symbolic integration I. Transcendental functions, algorithms and computation in mathematics, Vol. 1, Springer Verlag, Berlin Heidelberg GmbH, 1997.
- [7] Dieudonne J. Foundations of modern analysis. Enlarged and corrected printing, Academic Press, New York and London, 1961.
- [8] Geddes KO, Czapor SR, Labahn G. Algorithms for computer algebra. Kluwer Academic Publishers, Bostom/Dordrecht/London, 1992.
- [9] Grozin A. Introduction to Mathematica<sup>®</sup> for physicists. Graduate Texts in Physics, Springer International Publishing, 2014.
- [10] Hardy GH. Divergent series. Oxford, at the Clarendon Press, 1949.
- [11] Körner TW. A companion to analysis. A second first and first second course in analysis. Graduate Studies in Mathematics 62, American Mathematical Society, Providence, RI, 2004.
- [12] Lang S. Undergraduate analysis. Undergraduate Texts in Mathematics, Springer-Verlag, New York, 1997.
- [13] Moszyński M. Analiza matematyczna dla informatyków. Wykłady dla pierwszego roku informatyki na Wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego, 2010 (in Polish). Available at <https://www.mimuw.edu.pl/~mmoszyns/Analiza-dla-Informatykov-2017-18/SKRYPT>.
- [14] Ponnusamy S. Foundations of mathematical analysis. Birkhäuser/Springer, New York, 2012.
- [15] Suppes P. Axiomatic set theory. Dover Books on Mathematics, Dover Publications, 1st edition, 1972.
- [16] Wagner DB. Dynamic programming. The Mathematica Journal 1995, 5, 42–51. <https://www.mathematica-journal.com/issue/v5i4/columns/wagner/42-51wagner.mj.pdf>.
- [17] Wagon S. Mathematica<sup>®</sup> in action: problem solving through visualization and computation. Springer, 3rd edition, 2010.

<https://doi.org/10.1515/9783110590142-008>



# Index

\*\* 12

\$Assumptions 5, 6

\$MachinePrecision 37

\$MaxPrecision 43

\$MinPrecision 43

Abel's continuity/limit theorem 125

Abel's summation formula 58

Abel's test 59

Abs 104

absolute convergence 51

Accumulate 47, 48

almost uniform convergence 144

antiderivative 159

arbitrary precision arithmetic 36

ArcSinh 119

ArcTan 137, 166

Arg 166

Assuming 5, 7

Assumptions 5–7, 174, 175, 179

Attribute 103

Block 44, 104, 175

BlockRandom 171

Cauchy principal value of the integral 180

Cauchy product 68

Cauchy–Hadamard Theorem 129

Cauchy's root test 52

Cesàro regularization 69

Chop 51

ColorFunction 123

ColorFunctionScaling 123

comparison test 53

ComplexExpand 10, 20, 29, 166

ComplexInfinity 14

ComplexityFunction 4, 56

concave 119

Constants 103

convex 119

Cosh 117

critical point 109

critical value 109

CubeRoot 101

D 103, 105

D'Alembert's ratio test 52

Darboux integrable 170

Darboux integral 170

derivation 163

derivative 97

Derivative 105

difference quotient 95

Differences 44, 47, 48

differentiable 97

differential equation 159

differential field 163

differential operator 163

DirectedInfinity 14

DirectedInfinity[] 14

Direction 62, 77

Dirichlet's test 59

Dirichlet's test for uniform convergence 152

discontinuous functions 79

discrete dynamical system 39

DiscreteLimit 24, 50, 75

DiscreteMaxLimit 24

DiscreteMinLimit 24

DiscretePlot 83

DiscretePlot3D 30

Distribute 12

domains 2

Dot 57

DSolve 159

Dt 103

dynamic programming 33

Element 2

Exists 7

Expand 3, 12

extension of a differential field 163

extremum 109

Factor 5

FactorSquareFree 167

Fermat's Interior Extremum Theorem 109

Fibonacci[n] 41

field of constants 164

FindRoot 22, 86

fixed point 39

FixedPoint 36

FixedPointList 36

ForAll 7

- FullSimplify 3, 8
- FunctionRange 118
- Fundamental Theorem of Calculus 173
  
- GenerateConditions 76
- greatest lower bound 27
  
- Head 132
  
- If 80
- improper integrals 178
- indefinite integral 159
- Indeterminate 14
- induction axiom 15
- inexact number 35
- infimum 27
- Infinity 13
- InputForm 36
- Integral test 53
- integral test 181
- IntegralTest 183
- Integrate 159, 172, 175, 176
- Intermediate value property 84
- InverseFunction 87, 117, 118, 135
- InverseSeries 135
  
- Jensen's inequality 123
- Join 1
  
- Lagrange Remainder Terms Theorem 138
- LeafCount 4
- least upper bound 27
- left differentiable 99
- left hand derivative 99
- Leibniz's test 59
- Limit 24, 49, 75, 76, 156
- limit comparison test 53
- limit point 75
- Liouville's Principle 163
- Lipschitz condition 91
- Lipschitz function 110
- local maximum 109
- local minimum 109
- lower bound 27
- lower Darboux sum 170
- LucasL 42
  
- MachinePrecision 35–37, 43, 51, 70
- Maclaurin series 132
  
- Manipulate 39
- Maximize 28–30, 111, 112
- MaxLimit 24, 129
- Mean 113
- Mean Value Theorem 110
- MemberQ 2
- Method 52, 53, 62, 183
- Minimize 28–30, 111
- MinLimit 24
- Module 104
  
- N[a, p] 36, 43
- Nest 34
- NestList 34
- Newton–Leibniz formula 173
- NIntegrate 175, 176, 181
- NMaximize 32
- norm 143
- Normal 132, 167
- NSolve 86
- NSum 50
- null sequence 47
- NumberForm 181
  
- O 131
  
- Partition 113
- partition 169
- Peano Remainder Terms Theorem 130
- Peano's axioms 15
- Piecewise 79, 80
- Plot 123
- Plot3D 30
- pointwise convergence 144
- PolyLog 126, 168
- PolynomialQuotientRemainder 165
- Power 101
- power series 71, 125
- PowerExpand 7
- Precision 36
- Prime[n] 17
- primitive function 159
- PrincipalValue 180
- Principle of Mathematical Induction 16
  
- Quiet 40
  
- Raabe's test 53
- Rationalize 139

- RecurrenceTable 33
- recursive sequence 33
- Reduce 8, 18, 113
- Refine 5
- Regularization 68
- regularized sum 68
- Resolve 8
- Riemann integrable 171
- Riemann integral 172
- Riemann sum 176
- Riemann's theorem 65
- right differentiable 99
- right hand derivative 99
- Risch algorithm 160
- Root 19, 20
- RootReduce 21
- Roots 19
- RootSum 167
- RSolve 34, 38, 89
  
- SeedRandom 171
- sequence 22
- series 48
- Series 125, 131, 132
- SeriesCoefficient 72, 133
- SeriesData 132
- sets 1
- Sign 44, 113
- Simplify 3
- Sinh 117
- Solve 18
  
- Sort 1
- Sum 49, 50, 68
- sum of the series 48
- SumConvergence 50, 52, 53, 62, 71, 127, 136, 183
- supremum 27
- Surd 101
  
- TargetFunctions 10
- Taylor polynomial 129
- Taylor series 130
- the least upper bound 27
- The Weierstrass theorem 84
- TimeConstraint 3
- Timing 35
- ToRadicals 21
- TraditionalForm 24, 101
- TransformationFunctions 4
- TrigReduce 4
- TrigToExp 163, 165
  
- uniform convergence 144
- uniformly continuous 90
- Union 1, 2
- upper bound 27
- upper Darboux sum 170
  
- VerifyConvergence 50
  
- Weierstrass M-test 152
- While 85, 139

