20 Computer Algebra Systems- Example Mathematica

20.1 Introduction

20.1.1 Brief Characterization of Computer Algebra Systems

20.1.1.1 General Purpose of Computer Algebra Systems

The development of computers has made possible the introduction of computer algebra systems for "doing mathematics". They are software systems able to perform mathematical operations formally. These systems, such as Macsyma, Reduce, Derive, Maple, Mathematica, Matlab, Sage, can also be used on relatively small computers (PC), and with their help, we can transform complicated expressions, calculate derivatives and integrals, solve systems of equations, represent functions of one and of several variables variables graphically, etc. They can *manipulate* mathematical expressions, i.e., they can transform and simplify mathematical expressions according to mathematical rules if this is possible in closed form. They also provide a wide range of numerical solutions to required accuracy, and they can represent functional dependence between data sets graphically.

Most computer algebra systems can import and export data. Besides a basic offer of definitions and procedures which are activated at every start of the system, most systems provide a large variety of libraries and program packages from special fields of mathematics, which can be loaded and activated on request (see [20.15],[20.16]). Computer algebra systems allow users to build up their own packages [20.11]–[20.14].

However, the possibilities of computer algebra systems should not be overestimated. They spare us the trouble of boring, time-demanding, and mechanical computations and transformations, but they do not save us from thinking.

For frequent errors see 19.8.2, p. 1004.

20.1.1.2 Restriction to Mathematica

The systems are under perpetual developing. Therefore, every concrete representation reflects only a momentary state. Here we introduce the basic idea and applications of these systems for the most important fields of mathematics. This introduction will help for the first steps in working with computer algebra systems. In particular, we discuss Mathematica compatible until Version 10. This system seems to be very popular among users, and the basic structures of the other systems are similar.

In this book, we do not discuss how computer algebra systems are installed on computers. It is assumed that the computer algebra system has already been started by a command, and it is ready to communicate by command lines or in a Windows-like graphical environment.

The input and output is always represented for Mathematica (see 19.8.4.2, **1.**, p. 1016) in rows which are distinguished from other text parts, e.g., in the form

$$\texttt{In[1]} := \texttt{Solve}[3\ x - 5 == 0, x] \qquad (20.1)$$

System specific symbols (commands, type notation, etc.) will be represented in typewriter style.

In order to save space, we often write the input and the output in the same row in this book, and we separate them by the symbol $\longrightarrow$.

20.1.1.3 Two Introducing Examples of Basic Application Fields

1. Manipulation of Formulas

Formula manipulation means here the transformation of mathematical expressions in the widest sense, e.g., simplification or transformation into a useful form, representation of the solution of equations or systems of equations by algebraic expressions, differentiation of functions or determination of indefinite integrals, solution of differential equations, formation of infinite series, etc.

I.N. Bronshtein et al., *Handbook of Mathematics*,
DOI 10.1007/978-3-662-46221-8_20

■ Solution of the following quadratic equation:

$$x^2 + ax + b = 0 \quad \text{with} \quad a, b \in \mathbb{R}. \tag{20.2a}$$

In Mathematica, one types:

$$\texttt{Solve}[x^2 + a\ \ x + b == 0, x]\,. \tag{20.2b}$$

After pressing the corresponding input key/keys (ENTER or SHIFT+RETURN, depending on the operation system), Mathematica replaces this row by

$$\texttt{In[1]} := \texttt{Solve}[x^2 + a\ x + b == 0, x] \tag{20.2c}$$

and starts the evaluation process. In a moment, the answer appears in a new row

$$\texttt{Out[1]} = \{\{x \text{->} \frac{1}{2}\left(-a - \sqrt{a^2 - 4b}\right)\}, \{x \text{->} \frac{1}{2}\left(-a + \sqrt{a^2 - 4b}\right)\}\}. \tag{20.2d}$$

Mathematica has solved the equation and both solutions are represented in the form of a *list* consisting of two sublists.

2. Numerical Calculations

Computer algebra systems provide many procedures to handle numerical problems of mathematics. These are solutions of algebraic equations, linear systems of equations, the solutions of transcendental equations, calculation of definite integrals, numerical solutions of differential equations, interpolation problems, etc.

■ Problem: Solution of the equation

$$x^6 - 2x^5 - 30x^4 + 36x^3 + 190x^2 - 36x - 150 = 0. \tag{20.3a}$$

Although this equation of degree six cannot be solved in closed form, it has six real roots, which are to be determined numerically.

In Mathematica the input is:

$$\texttt{In[1]} := \texttt{NSolve}[x^6 - 2x^5 - 30x^4 + 36x^3 + 190x^2 - 36x - 150 == 0, x] \tag{20.3b}$$

It results in the answer:

$$\texttt{Out[1]} = \{\{x \text{->} -4.42228\}, \{x \text{->} -2.14285\}, \{x \text{->} -0.937347\}, \{x \text{->} 0.972291\},$$
$$\{x \text{->} 3.35802\}, \{x \text{->} 5.17217\}\} \tag{20.3c}$$

This is a list of the six solutions with a certain accuracy which will be discussed later.

20.2 Important Structure Elements of Mathematica

Mathematica is a computer algebra system, developed by Wolfram Research Inc. A detailed description of Mathematica can be found in [20.11]–[20.16]. For the current version 10 see the Virtual Book in the online Help.

20.2.1 Basic Structure Elements of Mathematica

In Mathematica the basic structure elements are called *expressions*. Their syntax is (it is emphasized again, that the current objects are given by their corresponding symbol, by their names):

$$\texttt{obj}_0[\texttt{obj}_1, \texttt{obj}_2, \ldots, \texttt{obj}_n] \tag{20.4}$$

$\texttt{obj}_0$ is called the head of the expression; the number 0 is assigned to it. The parts $\texttt{obj}_i$ $(i = 1, \ldots, n)$ are the *elements* or *arguments* of the expression, and one can refer to them by their numbers $1, \ldots, n$. In many cases the head of the expression is an operator or a function, the elements are the operands or variables on which the head acts.

Also the head, as an element of an expression, can be an expression, too. Square brackets are reserved in Mathematica for the representation of an expression, and they can be applied only in this relation.

■ The term x^2+2*x+1, which can also be entered in this infix form (and also in the nicer, preferred form x^2+2x+1) in Mathematica, has the complete form (FullForm)

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

which is also an expression. Plus, Power and Times denote the corresponding arithmetical operations.

The example shows that all simple mathematical operators exist in prefix form in the internal representation, and the term notation is only a facility in Mathematica.
Parts of expressions can be separated. This can be done with Part[*expr*, *i*], where i is the number of the corresponding element. In particular, $i = 0$ gives back the head of the expression.

■ If entering in Mathematica

In[1] := x^2+2x+1,

then after the SHIFT and ENTER keys together are pressed, Mathematica answers

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

Mathematica analyzed the input and returned it in mathematical standard form. If the input had been terminated by a semicolon, then the output would have been suppressed.
If entering

In[2] := FullForm[%]

then the answer is

Out[2] = Plus[1, Times[2, x], Power[x, 2]]

The sign % in the square brackets tells Mathematica that the argument of this input is the last output. From this expression it is possible to get, e.g., the third element

In[3] := Part[%, 3] as *Out[3]* = x^2

which is again an expression in this case.

Symbols in Mathematica are the notation of the basic objects; they can be any sequence of letters and numbers but they must not begin with a number. The special sign $ is also allowed. Upper-case and lower-case letters are distinguished. Reserved symbols begin either with a capital letter, or with the sign $, and in compound words also the second word begins with a capital letter, if it has a separate meaning. Users are advised to create their own symbols starting with lower-case letters.

20.2.2 Types of Numbers in Mathematica

20.2.2.1 Basic Types of Numbers

Mathematica knows four types of numbers represented in **Table 20.1**.

Table 20.1 Types of numbers in Mathematica

Type of number	Head	Characteristic	Input
Integers	Integer	exact integer, arbitrarily long	*nnnnn*
Rational numbers	Rational	fraction of coprimes in form Integer/Integer	*pppp/qqqq*
Real numbers	Real	floating-point number, arbitrary given precision	*nnnn.mmmm*
Complex numbers	Complex	complex number in the form *number*+*number*∗I	

Real numbers, i.e., floating-point numbers, can be arbitrarily long. If an integer *nnn* is written in the form *nnn.*, then Mathematica considers it as a floating-point number, that is, of type Real.
The type of a number x can be determined with the command Head[x]. Hence, *In[1]* := Head[51] results in *Out[1]* = Integer, while *In[2]* := Head[51.] *Out[2]* = Real. The real and imaginary components of a complex number can belong to any type of numbers. A number such as 5.731 + 0 I is considered as a Real type by Mathematica, while 5.731 + 0. I is of type Complex, since 0. is considered as a floating-point approximation of 0.

There are some further operations, which give information about numbers. So,

In[3] := NumberQ[51] results in *Out[3]* = True, (20.5a)

Otherwise, if x is manifestly not a number, as e.g. $x = \pi$, then the output is *Out[3]* =False. However, NumericQ[π] gives True. Here, True and False are the symbols for Boolean constants.
IntegerQ[x] tests if x is an integer, or not, so

In[4] := IntegerQ[2.] $\longrightarrow$ *Out[4]* = False (20.5b)

Similar tests can be performed for numbers with heads EvenQ, OddQ and PrimeQ. Their meanings are obvious. So, one gets

In[5] := PrimeQ[1075643] $\longrightarrow$ *Out[5]* = True (20.5c)

while

In[6] := PrimeQ[1075641] $\longrightarrow$ *Out[6]* = False (20.5d)

These last tests belong to a group of test operators, called predicates or criteria, which all end by Q and always answer True or False in the sense of a logical test (including a type check).

20.2.2.2 Special Numbers

In Mathematica, there are some special numbers which are often needed, and they can be called with arbitrary accuracy. They include π with the symbol Pi, e with the symbol E, $\frac{\pi}{180^\circ}$ as the transformation factor from degree measure into radian measure with the constant Degree, Infinity as the symbol for ∞ and the imaginary unit I.

20.2.2.3 Representation and Conversion of Numbers

Numbers can be represented in different forms which can be converted into each other. So, every real number x can be represented by a floating-point number N[x, n] with an n-digit precision.

IN[1] := N[E, 20] yields $\longrightarrow$ *Out[1]* = 2.7182818284590452354 (20.6a)

With Rationalize[x, dx], the number x with an accuracy dx can be converted into a rational number, i.e., into the fraction of two integers.

In[2] := Rationalize[%, 10^-5] $\longrightarrow$ *Out[2]* = $\frac{1071}{394}$ (20.6b)

With 0 accuracy, Mathematica gives the possible best approximation of the number x by a rational number.
Numbers of different number systems can be converted into each other. With BaseForm[x, b], the number x given in the decimal system is converted into the corresponding number in the number system with base $b \leq 36$. If $b > 10$, then the consecutive letters of the alphabet $a, b, c, \ldots$ are used for the further digits having a meaning greater than ten.

■ **A**: *In[1]* := BaseForm[255, 16] $\longrightarrow$ *Out[1]* = ff_1 (20.7a)

In[2] := BaseForm[N[E, 10], 8] $\longrightarrow$ *Out[2]* = 2.5576052131_8 (20.7b)

The reversed transformation can be performed by b^^$mmmm$.

■ **B**: *In[1]* := 8^^735 $\longrightarrow$ *Out[1]* = 477 (20.7c)

Numbers can be represented with arbitrary precision (the default here is the hardware precision), and for large numbers so-called scientific form is used, i.e., the form $n.mmmm10^\wedge \pm qq$.

20.2.3 Important Operators

Several basic operators can be written in infix form (as in the classical form in mathematics) $< symb_1\, op\, symb_2 >$. However, in every case, the complete form of this simplified notation is the expression $op[symb_1, symb_2]$. The most often occurring operators and their complete form are collected in **Table 20.2**. Most symbols in **Table 20.2** are obvious. For multiplication in the form $a\,b$, the space between the factors is very important.

Table 20.2 Important Operators in Mathematica

$a + b$	`Plus`$[a, b]$	$u == v$	`Equal`$[u, v]$
$a\,b$ or $a * b$	`Times`$[a, b]$	$w != v$	`Unequal`$[w, v]$
$a\hat{}b$ ore a^b	`Power`$[a, b]$	$r > t$	`Greater`$[r, t]$
a/b	`Times`$[a,$ `Power`$[b, -1]]$	$r \geq t$	`GreaterEqual`$[r, t]$
u–> v	`Rule`$[u, v]$	$s < t$	`Less`$[s, t]$
$r = s$	`Set`$[r, s]$	$s \leq t$	`LessEqual`$[s, t]$

The expressions with the heads `Rule` and `Set` will be explained. `Set` assigns the value of the expression s on the right-hand side, e.g., a number, to the expression r on the left-hand side, e.g., a variable. From here on, r is represented by this value until this assignment is changed. The change can be done either by a new assignment or by $x = .$ or by `Clear`$[x]$, i.e., by releasing every assignment so far. The construction `Rule` should be considered as a transformation rule. It occurs together with /. which is the substitution operator.

`Replace`$[t, u$ –> $v]$ or $t/.\, u$–> v means that every occurance u in the expression t will be replaced by the expression v.

■ *In[1]* := $x + y^2\,/.\, y$–> $a + b$ ⟶ *Out[1]* $= (a + b)^2 + x$

It is typical in the case of both operators that the right-hand side is evaluated immediately after the assignment or transformation rule. So, the left-hand side will be replaced by this evaluated right-hand side at every later call.

Here, two further operators have to be mentioned with delayed evaluation.

The `FullForm` of $u := v$ is `SetDelayed`$[u, v]$ and (20.8a)

the `FullForm` of $u :-> v$ is `RuleDelayed`$[u, v]$ (20.8b)

The assignment or the transformation rule are also valid here until it is changed. Although the left-hand side is always replaced by the right side, the right-hand side is evaluated for the first time only at the moment when the left one is called.

The expression $u == v$ or `Equal`$[u, v]$ returns `True` if u and v are identical. `Equal` is used, e.g., in manipulation of equations.

20.2.4 Lists

20.2.4.1 Notions

Lists are important tools in Mathematica for the manipulation of whole groups of quantities, which are important in higher-dimensional algebra and analysis.

A *list* is a collection of several objects into a new object. In the list, each object is distinguished only by its place in the list. The construction of a list is made either by the command

`List`$[a1, a2, a3, \ldots]$ or by $\{a1, a2, a3, \ldots\}$ (20.9)

if the elements can be simply enumerated. To explain the work with lists, a particular list is used, denoted by $l1$:

In[1] := $l1 =$ `List`$[a1, a2, a3, a4, a5, a6]$ ⟶ *Out[1]* $= \{a1, a2, a3, a4, a5, a6\}$ (20.10)

Mathematica applies a short form to the output of the list: It is enclosed in curly braces.

Table 20.3 represents commands which choose one or more elements from a list, and the output is a "sublist".

■ For the list $l1$ in (20.9) one gets, e.g.,

In[2] := `First`$[l1]$ ⟶ *Out[2]* $= a1$ *In[3]* := $l1$[[3]] ⟶ *Out[3]* $= a3$

In[4] := $l1$[[{2, 4, 6}]] ⟶ *Out[4]* $= \{a2, a4, a6\}$

In[5] := `Take`$[l1, 2]$ ⟶ *Out[5]* $= \{a1, a2\}$

Table 20.3 Commands for the choice of list elements

`First`[l], `Last`[l]	selects the first/last element
`Most`[l], `Rest`[l]	selects the elements except the last/first one
`Part`[l, n] or l[[n]]	selects the n-th element
`Part`[l, {$n1$, $n2$, ...}]	gives a list of the elements with the given numbers
l[[{$n1$, $n2$, ...}]]	equivalent to the previous operation
`Take`[l, m]	gives the list of the first m elements of l
`Take`[l, {m, n}]	gives the list of the elements from m through n
`Drop`[l, n]	gives the list without the first n elements
`Drop`[l, {m, n}]	gives the list without the elements from m through n

20.2.4.2 Nested Lists

The elements of lists can also be lists, so nested lists can be obtained. If entering, e.g., for the elements of the previous list $l1$ (20.10)

$$\texttt{In[1]} := a1 = \{b11, b12, b13, b14, b15\}$$

$$\texttt{In[2]} := a2 = \{b21, b22, b23, b24, b25\}$$

$$\texttt{In[3]} := a3 = \{b31, b32, b33, b34, b35\}$$

and analogously for $a4$, $a5$ and $a6$, then because of (20.10) a nested list (an array) is obtained which is not shown here explicitly. One can refer to the j-th element of the i-th sublist with the command `Part`[l, i, j]. The expression l[[i, j]] has the same result. In the above example (p. 1027), e.g.,

$\texttt{In[4]} := l1$[[3, 4]] yields $\texttt{Out[4]} = b34$

Furthermore, `Part`[l, {$i1$, $i2$ ...}, {$j1$, $j2$ ...}] or l[[{$i1$, $i2$, ...}, {$j1$, $j2$, ...}]] results in a list consisting of the elements numbered with $j1, j2 \ldots$ from the lists numbered with $i1, i2, \ldots$.

■ For the above example in 20.2.4.1, p. 1027

$\texttt{In[1]} := l1$[[{3, 5}, {2, 3, 4}]] $\longrightarrow$ $\texttt{Out[1]} = \{\{b32, b33, b34\}, \{b52, b53, b54\}\}$

The idea of nesting lists is obvious from these examples. It is easy to create lists of three or higher dimensions, and it is easy to refer to the corresponding elements.

20.2.4.3 Operations with Lists

Mathematica provides several further operations by which lists can be monitored, enlarged or shortened (**Table 20.4**).

Table 20.4 Operations with lists

`Position`[l, a]	gives a list of the positions where a occurs in the list
`MemberQ`[l, a]	checks whether a is an element of the list
`Select`[l, *crit*]	picks out all elements of the list for which crit holds
`Cases`[l, *pattern*]	gives a list of elements which match the pattern
`FreeQ`[l, a]	checks if a does not occur in the list
`Prepend`[l, a]	changes the list by adding a to the front
`Append`[l, a]	changes the list by appending a to the end
`Insert`[l, a, i]	inserts a at position i in the list
`Delete`[l, {i, j, ...}]	delete the elements at positions $i, j, \ldots$ from the list
`ReplacePart`[l, a, i]	replace the element at position i by a

■ With `Delete`, the original list $l1$ (20.10) can be shortened by the term $a6$:

$\texttt{In[1]} := l2 =$ `Delete`[$l1$, 6] $\longrightarrow$ $\texttt{Out[1]} = \{a1, a2, a3, a4, a5\}$,

where in the output the ai are shown by their values – they are lists themselves.

20.2.4.4 Tables

In Mathematica, several operations are available to create lists. One of them, which often occurs in working with mathematical functions, is the command `Table` shown in **Table 20.5**.

Table 20.5 Operation `Table`

`Table`[f, {$imax$}]	creates a list with $imax$ values of f: $f(1), f(2), \ldots, f(imax)$
`Table`[f, {i, $imin$, $imax$}]	creates a list with values of f from $imin$ to $imax$
`Table`[f, {i, $imin$, $imax$, di}]	the same as the last one, but by steps di

■ Table of binomial coefficients for $n = 7$:

In[1] := `Table`[`Binomial`[7, i], {i, 0, 7}]] ⟶ *Out[1]* = {1, 7, 21, 35, 35, 21, 7, 1}

With `Table`, also higher-dimensional arrays can be created. With the expression

`Table`[f, {i, $i1$, $i2$}, {j, $j1$, $j2$}, ...]

a higher-dimensional, multiple nested table is obtained, i.e., entering

In[2] := `Table`[`Binomial`[i, j], {i, 1, 7}, {j, 0, i}]

the binomial coefficients are got up to degree 7:

Out[2] = {{1, 1}, {1, 2, 1}, {1, 3, 3, 1}, {1, 4, 6, 4, 1},

{1, 5, 10, 10, 5, 1}, {1, 6, 15, 20, 15, 6, 1}, {1, 7, 21, 35, 35, 21, 7, 1}}

The operation `Range` produces a list of consecutive numbers or equally spaced numbers:

`Range`[n] yields the list {1, 2, ..., n}

Similarly, `Range`[$n1$, $n2$] and `Range`[$n1$, $n2$, dn] produce lists of numbers (arithmetic sequences) from $n1$ to $n2$ with step-size 1 or dn respectively. The command `Array` uses functions (as opposed to function values used by `Table`) to create lists. `Array[Exp,5]` yields $\{e, e^2, e^3, e^4, e^5\}$.

20.2.5 Vectors and Matrices as Lists

20.2.5.1 Creating Appropriate Lists

Several special (list) commands are available for defining vectors and matrices. A one-dimensional list of the form

$$v = \{v1, v2, \ldots, vn\} \tag{20.11}$$

can always be considered as a vector in n-dimensional space with components $v1, v2, \ldots, vn$. The special operation `Array`[v, n] produces the list (the vector) $\{v[1], v[2], \ldots, v[n]\}$. Symbolic vector operations can be performed with vectors defined in this way.

The two-dimensional lists $l1$ (see 20.2.4.2, p. 1028) and $l2$ (see 20.2.4.3, p. 1028) can be considered as matrices with rows i and columns j. In this case bij would be the element of the matrix in the i-th row and the j-th column. A rectangular matrix of type (6,5) is defined by $l1$, and a square matrix of type (5, 5) by $l2$.

With the operation `Array`[b, {n, m}] a matrix of type (n, m) is generated, whose elements are denoted by $b[i, j]$. The rows are numbered by i, i changes from 1 to n; by j the columns are numbered from 1 to m. In this symbolic form $l1$ can be created as

$$l1 = \texttt{Array}[b, \{6, 5\}], \quad \text{(20.12a)} \quad \text{where } b[i, j] = bij \quad (i = 1, \ldots, 6;\ j = 1, \ldots, 5). \quad \text{(20.12b)}$$

Summarizing, lists can be created either by enumeration or by using the functions `Array, Range, Table`. Note that lists are different from sets in mathematics.

The operation `IdentityMatrix`[n] produces the n-dimensional unit matrix.

With the operation `DiagonalMatrix`[*list*] a diagonal matrix is produced with the elements of the *list* in its main diagonal.

The operation `Dimension`[*list*] gives the size (number of rows, columns, ...) of a matrix, whose structure is given by a *list*. Finally, with the command `MatrixForm`[*list*], one gets a matrix-type representation of the *list*. A further possibility to define matrices is the following: Let $f(i, j)$ be a function of integers i and j. Then, the operation `Table`[f[i j], {i, n}, {j, m}] defines a matrix of type (n, m), whose elements are the corresponding $f(i, j)$.

20.2.5.2 Operations with Matrices and Vectors

Mathematica allows formal manipulation of matrices and vectors. The operations given in **Table 20.6** can be applied.

Table 20.6 Operations with matrices

$c\,a$	matrix a is multiplied by the scalar c
$a\,.\,b$	the product of matrices a and b
`Det`[a]	the determinant of matrix a
`Inverse`[a]	the inverse of matrix a
`Transpose`[a]	the transpose of matrix a
`MatrixExp`[a]	the exponential function of matrix a
`MatrixPower`[a, n]	the n-th power of matrix a
`Eigenvalues`[a]	the eigenvalues of matrix a
`Eigenvectors`[a]	the eigenvectors of matrix a

■ **A** : *In[1]* := r = `Array`[a, {4, 4}] ⟶ *Out[1]* = { { a[1, 1], a[1, 2], a[1, 3], a[1, 4] },
{ a[2, 1], a[2, 2], a[2, 3], a[2, 4] },
{ a[3, 1], a[3, 2], a[3, 3], a[3, 4] },
{ a[4, 1], a[4, 2], a[4, 3], a[4, 4] }}

In[2] := `Transpose`[r] ⟶ *Out[2]* = { { a[1, 1], a[2, 1], a[3, 1], a[4, 1] },
{ a[1, 2], a[2, 2], a[3, 2], a[4, 2] },
{ a[1, 3], a[2, 3], a[3, 3], a[4, 3] },
{ a[1, 4], a[2, 4], a[3, 4], a[4, 4] }}

Here, the transpose matrix $r^\top$ of r is produced.

Let the general four-dimensional vector v be defined by

In[3] := v = `Array`[u, 4] ⟶ *Out[3]* = {u[1], u[2], u[3], u[4]}

Now, the product of the matrix r and the vector v is again a vector (see Calculations with Matrices, 4.1.4, p. 272).

In[4] := $r\,.\,v$ ⟶ *Out [4]* = { a[1, 1] u[1] + a[1, 2] u[2] + a[1, 3] u[3] + a[1, 4] u[4] ,
a[2, 1] u[1] + a[2, 2] u[2] + a[2, 3] u[3] + a[2, 4] u[4] ,
a[3, 1] u[1] + a[3, 2] u[2] + a[3, 3] u[3] + a[3, 4] u[4] ,
a[4, 1] u[1] + a[4, 2] u[2] + a[4, 3] u[3] + a[4, 4] u[4] }.

There is no difference between row and column vectors in Mathematica. In general, matrix multiplication is not commutative (see Calculations with Matrices 4.1.4, p. 272). The expression $r\,.\,v$ corresponds to the product in linear algebra when a matrix is multiplied by a column vector from the right, while $v\,.\,r$ means a multiplication by a row vector from the left.

■ **B:** In the section on Cramer's rule (4.5.2.3, p. 311) the linear system of equations $pt = b$ is solved with the matrix

In[1] := `MatrixForm`[p = {{2, 1, 3}, {1, −2, 1}, {3, 2, 2}}] ⟶ *Out[1]* = $\begin{pmatrix} 2 & 1 & 3 \\ 1 & -2 & 1 \\ 3 & 2 & 2 \end{pmatrix}$

and vectors

In[2] := t = `Array`[x, 3] ⟶ *Out[2]* = {x[1], x[2], x[3]}

In[3] := b = {9, −2, 7} ⟶ *Out[3]* = {9, −2, 7}.

Since in this case `Det[p]==13` $\neq 0$ holds, the system can be solved by $t = p^{-1}b$. This can be done by

In[4] := `Inverse`$[p].b$ with the output of the solution vector *Out[4]* = $\{-1, 2, 3\}$.

Note that `a b` calculates the componentwise product and `Exp[a]` gives the matrix containing the exp of the components of the matrix `a`.

20.2.6 Functions

20.2.6.1 Standard Functions

Mathematica knows several standard mathematical functions, which are listed in **Table 20.7**.

Table 20.7 A few standard functions

Exponential function	`Exp`[x]
Logarithmic functions	`Log`[x], `Log`[b,x]
Trigonometric functions	`Sin`[x], `Cos`[x], `Tan`[x], `Cot`[x], `Sec`[x], `Csc`[x]
Arc functions	`ArcSin`[x], `ArcCos`[x], `ArcTan`[x], `ArcCot`[x], `ArcSec`[x], `ArcCsc`[x]
Hyperbolic functions	`Sinh`[x], `Cosh`[x], `Tanh`[x], `Coth`[x], `Sech`[x], `Csch`[x]
Area functions	`ArcSinh`[x], `ArcCosh`[x], `ArcTanh`[x], `ArcCoth`[x], `ArcSech`[x], `ArcCsch`[x]

All these functions can be applied even with complex arguments.

In every case must be considered the single-valuedness of the functions. For real functions one branch of the function has to be chosen (if it is needed); for functions with complex arguments the principal value (see 14.5, p. 758) should be chosen.

20.2.6.2 Special Functions

Mathematica knows several special functions, which are not standard functions. **Table 20.8** lists some of these functions.

Table 20.8 Special functions

Bessel functions $J_n(z)$ and $Y_n(z)$	`BesselJ`[n,z], `BesselY`[n,z]
Modified Bessel functions $I_n(z)$ and $K_n(z)$	`BesselI`[n,z], `BesselK`[n,z]
Legendre polynomials $P_n(x)$	`LegendreP`[n,x]
Spherical harmonic $Y_l^m(\vartheta, \phi)$	`SphericalHarmonicY`$[l, m, \theta, \phi]$

Further functions can be loaded with the corresponding special packages of Mathematica

20.2.6.3 Pure Functions

Mathematica supports the use of so-called pure functions. A pure function is an anonymous function, an operation with no name assigned to it. They are denoted by `Function`[x, *body*]. The first argument specifies the formal parameters and the second one is the body of the function, i.e., *body* is an expression for the function of the variable x.

In[1] := `Function`$[x, x^3 + x^2]$ $\longrightarrow$ *Out[1]* = `Function`$[x, x^3 + x^2]$ (20.13)

and so

In[2] := `Function`$[x, x^3 + x^2][c]$ gives *Out[2]* $= c^2 + c^3$. (20.14)

We can use a simplified version of this command. It has the form *body* &, where the variable is denoted by # . Instead of the previous two rows one can also write

In[3] := $(\#^3 + \#^2)\,\&\,[c]$ *Out[3]* $= c^2 + c^3$. (20.15)

It is also possible to define pure functions of several variables:

`Function`[$\{x_1, x_2, \ldots\}$, *body*] or in short form *body* &, where the variables in *body* are denoted by the elements $\#1, \#2, \ldots$. The sign & is very important for closing the expression, since it can be seen from this sign that the previous expression should be considered as a pure function. Let us remark that the

pure function #& is nothing else than the identity function: to any argument x it assigns x. Similarly, #1& corresponds to the projection onto the first coordinate axis.

20.2.7 Patterns

Mathematica allows users to define their own functions and to use them in calculations. With the command

$$\texttt{In[1]} := \texttt{f}[x\_] := \text{Polynomial}[x] \tag{20.16}$$

with Polynomial(x) as an arbitrary polynomial of variable x, a special function is defined by the user.

In the definition of the function f, there is no simple x, but $x\_$ (pronounced x-blank) with a symbol _ for the blank. The symbol $x\_$ means "something with the name x". From here on, every time when the expression f[*something*] occurs, Mathematica replaces it by its definition given above. This type of definition is called a *pattern*. The symbol *blank*_ denotes the basic element of a pattern; $y\_$ stands for y as a pattern. It is also possible to apply in the corresponding definition only a "_", that is $y\hat{}\_$. This pattern stands for an arbitrary power of y with any exponent, thus, for an entire class of expressions with the same structure.

The essence of a pattern is that it defines a *structure*. When Mathematica checks an expression with respect to a pattern, it compares the structure of the elements of the expression to the elements of the pattern, Mathematica does not check mathematical equality! This is important in the following example: Let l be the list

$$\texttt{In[2]} := l = \{1,\, y,\, y^a,\, y^{\sqrt{x}},\, \{\texttt{f}[y^{r/q}],\, 2^y\}\} \tag{20.17}$$

If one writes

$$\texttt{In[3]} := l\,/.\, y\hat{}\_ \mathrel{:>} yes \tag{20.18}$$

then Mathematica returns the list

$$\texttt{Out[3]} = \{1,\, y,\, yes,\, yes,\, \{\texttt{f}[yes],\, 2^y\}\} \tag{20.19}$$

Mathematica checked the elements of the list with respect to its structural identity to its pattern $y\hat{}\_$ and in every case when it determined coincidence it replaced the corresponding element by yes. The elements 1 and y were not replaced, since they have not the given structure, even though $y^0 = 1, y^1 = y$ holds.

Remark: Pattern comparison always happens in FullForm. If

$$\texttt{In[4]} := b/y\,/.\, y\hat{}\_ \mathrel{:>} yes \quad \text{is examined then} \quad \texttt{Out[4]} = b\; yes \tag{20.20}$$

This is a consequence of the fact that FullForm of b/y is Times[b, Power[y, -1]], and for structure comparison the second argument of Times is identified as the structure of the pattern.

With the definition

$$\texttt{In[5]} := \texttt{f}[x\_] := x^3 \tag{20.21a}$$

Mathematica replaces, corresponding to the given pattern,

$$\texttt{In[6]} = \texttt{f}[r] \quad \text{by} \quad \texttt{Out[6]} = r^3 \quad \text{etc.} \tag{20.21b}$$

$$\texttt{In[7]} := \texttt{f}[a] + \texttt{f}[x] \quad \text{yields} \quad \texttt{Out[7]} = a^3 + x^3 \tag{20.21c}$$

If however,

$$\texttt{In[8]} := \texttt{f}[x] := x^3, \quad \text{then for the same input} \quad \texttt{In[9]} := \texttt{f}[a] + \texttt{f}[x] \tag{20.21d}$$

the output would be

$$\texttt{Out[9]} = \texttt{f}[a] + x^3 \tag{20.21e}$$

In this case only the (fixed, single) input x corresponds to the definition.

20.2.8 Functional Operations

Functions operate on numbers and expressions. Mathematica can also perform operations with functions, since the names of functions are handled as expressions so they can be manipulated as expressions.

1. Inverse Function, Inverse Series The determination of the inverse function of a given function f can be made by the functional operation `InverseFunction` or `InverseSeries`.

■ **A:** *In[1]* := `InverseFunction`[f] [x] ⟶ *Out[1]* = $f^{-1}[x]$

■ **B:** *In[1]* := `InverseFunction`[`Exp`] ⟶ *Out[1]* = `Log`

■ **C:** *In[1]* := `InverseSeries`[`Series`[$g[x], \{x, 0, 2\}$]]

$$\textit{Out[1]} = \frac{x - g[0]}{g'[0]} - \frac{g''[0](x - g[0])^2}{2g'[0]^3} + O[x - g[0]]^3$$

2. Differentiation Mathematica uses the possibility that the differentiation of functions can be considered as a mapping in the space of functions. In Mathematica, the differentiation operator is `Derivative`[1][f] or in short form `f'`. If the function `f` is defined, then its derivative can be got by `f'`.

■ *In[1]* := `f`[$x\_$] := `Sin`[x] `Cos`[x] With

In[2] := `f'` follows *Out[2]* = `Cos`[#1]^2 − `Sin`[#1]^2 &,

hence `f'` is represented as a pure function and it evaluates to

In[3] := % [x] ⟶ *Out[3]* = `Cos`[x]^2 − `Sin`[x]^2

3. Nest The command `Nest`[f, x, n] means that the function `f` nested n times into itself should be applied on x. The result is `f`[`f`[...`f`[x]]...].

4. NestList By `NestList`[f, x, n] a list $\{x$, `f`[x], `f`[`f`[x]], ...$\}$ will be shown, where finally f is nested n times. `FoldList`[f, x, *list*] iterates a two-variable function.

5. FixedPoint For `FixedPoint`[f, x], the function is applied repeatedly until the result does not change.

6. FixedPointList The functional operation `FixedPointList`[f, x] shows the continued list with the results after `f` is applied, until the value no longer changes.

■ As an example for this type of functional operation the `NestList` operation will be used for the approximation of a root of an equation $f(x) = 0$ with Newton's method (see 19.1.1.2, p. 950). A root of the equation $x \cos x = \sin x$ is needed in the neighborhood of $3\pi/2$:

In[1] := `f`[$x\_$] := x − `Tan`[x] *In[2]* := `f'`[x] ⟶ *Out[2]* = 1 − `Sec`[x]^2

In[3] = `g`[$x\_$] := x − `f`[x]/`f'`[x]

In[4] := `NestList`[g, 4.6, 4] ⟶ *Out[4]* = {4.6, 4.54573, 4.50615, 4.49417, 4.49341}

In[5] := `FixedPoint`[g, 4.6] ⟶ *Out[5]* = 4.49341

A higher precision of the result can also be achieved.

7. Apply Let f be a function which is considered in connection with a list $\{a, b, c, \ldots\}$. Then

`Apply`[f, $\{a, b, c, \ldots\}$] $f[a, b, c, \ldots]$ (20.22)

■ *In[1]* := `Apply`[`Plus`, $\{u, v, w\}$] ⟶ *Out[1]* = $u + v + w$

In[2] := `Apply`[`List`, $a + b + c$] ⟶ *Out[2]* = $\{a, b, c\}$

Here, the general scheme of how Mathematica handles expressions of expressions can be easily recognized. The `FullForm` of the last operation is:

In[3] := `FullForm`[`Apply`[`List`, `Plus`[a, b, c]]] ⟶ *Out[3]* = `List`[a, b, c]

The functional operation `Apply` obviously replaces the `Head` of the considered expression `Plus` by the required `List`.

8. Map With a defined function f the operation `Map` gives:

`Map`[f, $\{a, b, c, \ldots\}$] ⟶ $\{f[a], f[b], f[c], \ldots\}$ (20.23)

`Map` generates a list whose elements are the values when f is applied to the original list.

■ Let f be the function $f(x) = x^2$. It is defined by

In[1] := `f`[$x\_$] := x^2 With this f one gets

In[2] := Map[f, {u, v, w}] ⟶ *Out[2]* = {u^2, v^2, w^2}

Map can be applied for more general expressions:

In[3] := Map[f, Plus[a, b, c]] ⟶ *Out[3]* = $a^2 + b^2 + c^2$

20.2.9 Programming

Mathematica can handle the loop constructions known from other languages for procedural programming. The two basic commands are

Do[*expr*, {$i, i1, i2, di$}] and (20.24a)

While[*test*, *expr*] (20.24b)

The first command evaluates the expression *expr*, where i runs over the values from $i1$ to $i2$ in steps di. If di is omitted, the step size is one. If $i1$ is also missing, then it starts from 1.
The second command evaluates the expression so far as *test* has the value True.

■ In order to determine an approximate value of e^2, the series expansion of the exponential function is used:

```
In[1]:= sum = 1.0;
        Do[sum = sum + (2^i/i!), {i, 1, 10}];
        sum
Out[1]= 7.38899
```
(20.25)

The Do loop evaluates its argument a previously given number of times, while the While loop evaluates as far as a previously given condition becomes false.

Among other things, Mathematica provides the possibility of defining and using local variables. This can be done by the command

Module[{$t1, t2, \ldots$}, *procedure*] (20.26)

The variables or constants enclosed in the list are locally usable in the module; their values assigned here are not valid outside of this module.

■ **A:** A procedure is to be defined which calculates the sum of the square roots of the integers from 1 to n.

```
In[1]:= sumq[n_] :=
          Module[{sum = 1.},
            Do[sum = sum + N[Sqrt[i]], {i, 2, n}];
            sum  ];
```
(20.27)

The call sumq[30] results in 112.083.

The real power of the programming capabilities of Mathematica is, first of all, the use of functional methods in programming, which are made possible by the operations Nest, NestWhile, Apply, Map, MapThread, Distribute and by some further ones.

■ **B:** Example **A** can be written in a functional manner for the case when an accuracy of ten digits is required:

sumq[$n\_$] := N[Apply[Plus, Table[Sqrt[i], {$i, 1, n$}]], 10],

sumq[30] results in 112.0828452. Total[$\sqrt{\text{N[Range[n], 10]}}$] gives the same result without using an index, increasing its value continuously, without needing the variable *sum* and its initial value.
For the details, see [20.16].

20.2.10 Supplement about Syntax, Information, Messages

20.2.10.1 Contexts, Attributes

Mathematica must handle several symbols; among them there are those which are used in further program modules loaded on request. To avoid many-valuedness, the names of symbols in Mathematica consist of two parts, the context and the short name.

Short names mean here the names (see 20.2, p. 1024) of heads and elements of the expressions. In addition, in order to name a symbol Mathematica needs the determination of the program part to which the symbol belongs. This is given by the *context*, which holds the name of the corresponding program part. The complete name of a symbol consists of the context and the short name, which are connected by the ' sign.

When Mathematica starts, then there are always two contexts present: *System'* and *Global'*. Information about other available program modules can be obtained by the command `Contexts[]`.

All built-in functions of Mathematica belong to the context *System'*, while the functions defined by the user belong to the context *Global'*.

If a context is actual, thus, the corresponding program part is loaded, then the symbols can be referred to by their short names.
For the input of a further Mathematica program module by `<< NamePackage`, the corresponding context is opened and introduced into the previous list. It can happen that a symbol has already been introduced with a certain name before this module is loaded, and in this newly opened context the same name occurs with another definition. In this case Mathematica gives a warning to the user. Then the previously defined name can be erased by the command `Remove[Global`name]`, or the *complete* name for the newly loaded symbol can be applied.

Besides the properties that the symbols have per definition, it is possible to assign to them some other general properties, called *attributes*, like `Orderless`, i.e., unordered or commutative, `Protected`, i.e., values cannot be changed, or `Locked`, i.e, attributes cannot be changed, etc. Informations about the already existing attributes of the considered object can be obtained by `Attributes[`f`]`.

Some symbols can be protected by `Protect[somesymbol]`; then no other definition can be introduced for this symbol. This attribute can be erased with the command `Unprotect`.

20.2.10.2 Information

Information can be obtained about the fundamental properties of objects by the commands

?*symbol*	information about the object given by the name *symbol*,
??*symbol*	detailed information about the object,
?$B*$	information about all Mathematica objects, whose names begin with B.

It is also possible to get information about special operators, e.g., by ? := about the SetDelay operator.

However, the most useful possibility is to put the cursor anywhere in the cell containing the symbol of the object in question and press the key F1.

20.2.10.3 Messages

Mathematica has a message system which can be activated and used for different reasons. The messages are generated and shown during the calculations. Their presentation has a uniform form: *symbol* :: *tag*, providing the possibility to refer to them later. (Such messages can also be created by the user.) Consider the following examples as illustrations.

■ **A:** *In[1]* := `f[x_] := 1/x;` *In[2]* := `f[0]`

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

Out[2] = `ComplexInfinity`

■ **B:** *In[1]* := `Log[3, 16, 25]`

`Log: :argt:Log called with 3 arguments; 1 or 2 arguments are expected.`

Out[1] =Log[3, 16, 25]

In example **A**, Mathematica warns us that during the evaluation of an expression it got the value ∞. The calculation itself can be performed. In example **B** the call of logarithm contains three arguments, which is not allowed according to the definition. Calculations cannot be performed. Mathematica cannot do anything with the expression. The user can switch off a message with Off[*s* :: *tag*]. With On the message will appear again. Quiet switches off all the messages.
With Messages[*symbol*] all messages associated to the symbol with the name *symbol* can be recalled.

20.3 Important Applications with Mathematica

This section describes how to handle mathematical problems with computer algebra systems. The choice of the considered problems is organized according to their frequency in practice and also according to the possibilities of solving them with a computer algebra system. Examples will be given for functions, commands, operations and supplementary syntax. When it is important, the corresponding special package is also discussed briefly.

20.3.1 Manipulation of Algebraic Expressions

In practice, further operations must usually be performed with the occurring algebraic expressions (see 1.1.5, p. 10) such as differentiation, integration, series representation, limiting or numerical evaluation, transformations, etc. In general, these expressions are considered over the ring of integers (see 5.3.7, p. 361) or over the field (see 5.3.7.1, **2.**, p. 361) of real numbers. Computer algebra systems can handle, e.g., polynomials also over finite fields or over extension fields (see 5.3.7.1, **3.**, p. 362) of the rational numbers. Interested people should study the special literature. The algebraic operations with polynomials over the field of rational numbers have special importance. Mathematica provides the functions and operations represented in **Table 20.9** for transformation of algebraic expressions. See also the menu item Palettes|Other|Algebraic Manipulation.

Table 20.9 Commands for manipulation of algebraic expressions

Expand[p]	expands the powers and products in a polynomial p by multiplication
Expand[p, r]	multiplies only the parts in p, which contain r
PowerExpand[a]	expands also the powers of products and powers of powers
Factor[p]	factorizes a polynomial completely
Collect[p, x]	orders the polynomial with respect the powers of x
Collect[$p, \{x, y, \ldots\}$]	the same as the previous one, with several variables
ExpandNumerator[r]	expands only the numerator of a rational expression
ExpandDenominator[r]	expands only the denominator
ExpandAll[r]	expands both numerator and denominator completely
Together[r]	combines the terms in the expression over a common denominator
Apart[r]	represents the expression in partial fractions
Cancel[r]	cancels the common factors in the fraction

20.3.1.1 Multiplication of Expressions

The operation of multiplication of algebraic expressions can always be performed. The coefficients can also be undefined expressions.

■ *In[1]* := Expand[$(x + y - z)^4$] gives

Out[1] = $x^4 + 4\,x^3\,y + 6\,x^2\,y^2 + 4\,x\,y^3 + y^4 - 4\,x^3\,z - 12\,x^2\,y\,z - 12\,x\,y^2\,z - 4\,y^3\,z$
$+6\,x^2\,z^2 + 12\,x\,y\,z^2 + 6\,y^2\,z^2 - 4\,x\,z^3 - 4\,y\,z^3 + z^4$

Similarly,

In[3] := Expand[$(a\,x + b\,y^2)(c\,x^3 - d\,y^2)$]

Out[3] = $a\,c\,x^4 - a\,d\,x\,y^2 + b\,c\,x^3\,y^2 - b\,d\,y^4$

20.3.1.2 Factorization of Polynomials

Mathematica performs factorization over the integer or rational numbers if it is possible. Otherwise the original expression is returned.

■ *In[1]* := $p = x^6 + 7x^5 + 12x^4 + 6x^3 - 25x^2 - 30x - 25$;
In[2] := Factor[p] , gives
Out[2] = $(5 + x)\ (1 + x + x^2)\ (-5 + x^2 + x^3)$

Mathematica decomposes the polynomial into three factors which are irreducible over the rational numbers.

If a polynomial can be completely decomposed over the complex rational numbers, then this can be obtained by the option GaussianIntegers.

■ *In[1]* := Factor[$x^2 - 2x + 5$] ⟶ *Out[1]* = $5 - 2x + x^2$, but
In[2] := FactorGaussianIntegers–> True]
Out[2] = $((-1 - 2\mathrm{I}) + x)((-1 + 2\mathrm{I}) + x)$

20.3.1.3 Operations with Polynomials

Table 20.10 contains a collection of operations by which polynomials can be algebraically manipulated over the field of rational numbers.

Table 20.10 Algebraic polynomial operations

PolynomialGCD[$p1, p2$]	determines the greatest common divisor of $p1$ and $p2$
PolynomialLCM[$p1, p2$]	determines the least common multiple of $p1$ and $p2$
PolynomialQuotient[$p1, p2, x$]	divides $p1$ (as a function of x) by $p2$, the residue is omitted
PolynomialRemainder[$p1, p2, x$]	determines the residue on dividing $p1$ by $p2$
MonomialList[p]	gives the list of all monomials in the polynomial p

■ Two polynomials are defined:
In[1] := $p = x^6 + 7x^5 + 12x^4 + 6x^3 - 25x^2 - 30x - 25$;
$q = x^4 + x^3 - 6x^2 - 7x - 7$;

With these polynomials the following operations are performed:

In[2] := PolynomialGCD[p, q] ⟶ *Out[2]* = $1 + x + x^2$
In[3] := PolynomialLCM[p, q]//Factor
Out[3] = $(5 + x)(-7 + x^2)(1 + x + x^2)(-5 + x^2 + x^3)$
In[4] := PolynomialQuotient[p, q, x] ⟶ *Out[4]* = $12 + 6x + x^2$
In[5] := PolynomialRemainder[p, q, x] ⟶ *Out[5]* = $59 + 96x + 96x^2 + 37x^3$

With the two last results one gets

$$\frac{x^6 + 7x^5 + 12x^4 + 6x^3 - 25x^2 - 30x - 25}{x^4 + x^3 - 6x^2 - 7x - 7} = x^2 + 6x + 12 + \frac{37x^3 + 96x^2 + 96x + 59}{x^4 + x^3 - 6x^2 - 7x - 7}.$$

20.3.1.4 Partial Fraction Decomposition

Mathematica can decompose a fraction of two polynomials into partial fractions, of course, over the field of rational numbers. The degree of the numerator of any part is always less than the degree of the denominator.

■ Using the polynomials p and q from the previous example one gets

In[1] := Apart[q/p] ⟶ *Out[1]* = $-\dfrac{6}{35\ (5 + x)} + \dfrac{-55 + 11\,x + 6\,x^2}{35\ (-5 + x^2 + x^3)}$

20.3.1.5 Manipulation of Non-Polynomial Expressions

Complicated expressions, not necessarily polynomials, can often be simplified by the help of the command `Simplify`. Mathematica will always try to manipulate algebraic expressions, independently of the nature of the symbolic quantities. Here, certain built-in knowledge is used. Mathematica knows the rules of powers (see 1.1.4.1, p. 7):

$$\texttt{In[1]} := \texttt{Simplify}[a^n/a^m)] \;\longrightarrow\; \texttt{Out[1]} = a^{-m+n} \tag{20.28}$$

With the option `Trig -> True`, the commands `Expand` and `Factor` can express powers of trigonometric functions by trigonometric functions with multiple arguments, and conversely. Alternatively, `TrigExpand, TrigFactor, TrigFactorList, TrigReduce, ExpToTrig, TrigToExp` can be applied.

■

```
In[1] := TrigExpand[Sin[2x]Cos[2y]]

Out[1] = 2Cos[x]Cos[y]^2Sin[x]-2Cos[x]Sin[x]Sin[y]^2

In[2] := Factor[Sin[4x], Trig-> True] - 8Cos[x]^3Sin[x] + 4Cos[x] Sin[x])

Out[2] = 0

In[3] := Factor[Cos[5x], Trig-> True]

Out[3] = Cos[x] (1 - 2 Cos[2x] + 2 Cos[4x]) .
```

Remark: The command `ComplexExpand[expr]` assumes a real variable `expr`, while in the command `ComplexExpand`$[expr, \{x1, x2, \ldots\}]$ the variables xi are supposed to be complex.

■

```
In[1]  := ComplexExpand[Sin[2 x],{x}]
Out[1] = Cosh[2 Im[x]] Sin[2 Re[x]]+I Cos[2 Re[x]] Sinh[2 Im[x]]
```

20.3.2 Solution of Equations and Systems of Equations

Computer algebra systems know procedures to solve equations and systems of equations. If the equation can be solved explicitly in the domain of algebraic numbers, then the solution will be represented with the help of radicals. If it is not possible to give the solution in closed form, then at least numerical solutions can be found with a given accuracy. In the following some basic commands will be introduced. The solution of systems of linear equations (see 4.5.2, p. 308) is discussed here in a special section (see 20.3.2.4, p. 1040).

20.3.2.1 Equations as Logical Expressions

Mathematica allows the manipulation and solution of equations within a wide range. In Mathematica, an equation is considered as a logical expression. If one writes

$$\texttt{In[1]} := g = x^2 + 2x - 9 == 0, \tag{20.29a}$$

then Mathematica considers it as a definition of a function with Boolean values. Giving the input

$$\texttt{In[2]} := \%/.\ x\text{-}> 2, \;\text{ yields }\; \texttt{Out[2]} = \texttt{False}, \tag{20.29b}$$

since with this value of x the left-hand side and right-hand side are not equal.
The command `Roots`$[g, x]$ transforms the above identity into a form which contains x explicitly. Mathematica represents the result with the help of the logical OR in the form of a logical statement:

$$\texttt{In[3]} := \texttt{Roots}[g, x] \;\longrightarrow\; \texttt{Out[3]} = x == -1 - \sqrt{10} || x == -1 + \sqrt{10} \tag{20.29c}$$

In this sense, logical operations can be performed with equations.
With the operation `ToRules`, the last logical type equations can be transformed as follows:

$$\texttt{In[4]} := \{\texttt{ToRules}[\%]\}$$

$$\texttt{Out[4]} = \{\{x \text{-}>\ -1 - \sqrt{10}\}, \{x \text{-}> -1 + \sqrt{10}\}\} \tag{20.29d}$$

20.3.2.2 Solution of Polynomial Equations

Mathematica provides the command `Solve` to solve equations. In a certain sense, `Solve` perform the operations `Roots` and `ToRules` after each other.

Mathematica solves polynomial equations in symbolic form up to fourth degree, since for these equations solutions can be given in the form of algebraic expressions. However, if equations of higher degree can be transformed into a simpler form by algebraic transformations, such as factorization, then Mathematica provides symbolic solutions. In these cases, `Solve` tries to apply the built-in operations `Expand` and `Decompose`.

In Mathematica numerical solutions are also available.

■ The general solution of an equation of third degree:

In[1] := `Solve`$[x^3 + a\,x^2 + b\,x + c == 0, x]$

Mathematica gives

$$\begin{aligned}\textit{Out[1]} = \{\{x \to & -\frac{a}{3} \\ & -\frac{2^{1/3}\,(-a^2+3\,b)}{3\left(-2\,a^3+9\,a\,b-27\,c+3^{3/2}\sqrt{-(a^2\,b^2)+4\,b^3+4\,a^3\,c-18\,a\,b\,c+27\,c^2}\right)^{1/3}} \\ & +\frac{\left(-2\,a^3+9\,a\,b-27\,c+3^{3/2}\sqrt{-(a^2\,b^2)+4\,b^3+4\,a^3\,c-18\,a\,b\,c+27\,c^2}\right)^{1/3}}{32^{1/3}}\}, \\ & \ldots\}\end{aligned}$$

The solution list here shows only the first term explicitly because of the length of their terms. If an equation with given coefficients a, b, c has to be solved, then it is better to handle the equation itself with `Solve` than to substitute a, b, c into the solution formula.

■ **A:** For the cubic equation (see 1.6.2.3, p. 40) $x^3 + 6x + 2 = 0$ one gets:

In[1] := `Solve`$[x^3 + 6\,x + 2 == 0, x]$

$$\textit{Out[1]} = \{\{x \to 2^{1/3} - 2^{2/3}\}, \{x \to \frac{1-\mathrm{I}\sqrt{3}}{2^{1/3}} - \frac{1+\mathrm{I}\sqrt{3}}{2^{2/3}}\}, \{x \to -\frac{1-\mathrm{I}\sqrt{3}}{2^{2/3}} + \frac{1+\mathrm{I}\sqrt{3}}{2^{1/3}}\}\}$$

■ **B:** Solution of an equation of sixth degree:

In[2] := `Solve`$[x^6 - 6x^5 + 6x^4 - 4x^3 + 65x^2 - 38x - 120 == 0, x]$

Out[2] $= \{\{x \to -1\}, \{x \to -1 - 2\,\mathrm{I}\}, \{x \to -1 + 2\,\mathrm{I}\}, \{x \to 2\}, \{x \to 3\}, \{x \to 4\}\}$

Mathematica succeeded in factorizing the equation in **B** with internal tools; then it is solved without difficulty.

If numerical solutions are required, then the command `NSolve` can be used.

■ The following equation is solved by `NSolve`:

In[3] := `NSolve`$[x^6 - 4x^5 + 6x^4 - 5x^3 + 3x^2 - 4x + 2 == 0, x]$

Out[3] $= \{\{x \to -0.379567 - 0.76948\,\mathrm{I}\}, \{x \to -0.379567 + 0.76948\,\mathrm{I}\},$
$\{x \to 0.641445\}, \{x \to 1. - 1.\,\mathrm{I}\}, \{x \to 1. + 1.\,\mathrm{I}\}, \{x \to 2.11769\}\}$

20.3.2.3 Solution of Transcendental Equations

Mathematica can solve transcendental equations, as well. In general, this is not possible symbolically, and these equations often have infinitely many solutions. In these cases, an estimate of the domain should be given, where Mathematica has to find the solutions. This is possible with the command `FindRoot`$[g, \{x, x_s\}]$, where x_s is the initial value for the search of the root.

■ *In[1]* := `FindRoot`$[x + $`ArcCoth`$[x] - 4 == 0, \{x, 1.1\}]$

Out[1] $= \{x \to 1.00502\}$ and

$$In[2] := \texttt{FindRoot}[x + \texttt{ArcCoth}[x] - 4 == 0, \{x, 5\}] \longrightarrow Out[2] = \{x \text{-> } 3.72478\}$$

20.3.2.4 Solution of Systems of Equations

Mathematica can solve simultaneous equations. The operations, built-in for this purpose, are represented in **Table 20.11**, and they present the symbolical solutions, not the numerical ones.
Similarly to the case of one unknown, the command `NSolve` gives the numerical solution(s). The solution of systems of linear equations is discussed in 20.3.3, p. 1040.

Table 20.11 Operations to solve systems of equations

`Solve`$[\{l_1 == r_1, l_2 == r_2, \ldots\}, vars]$	solves the given system of equations with respect to *vars*
`Eliminate`$[\{l_1 == r_1, \ldots\}, vars]$	eliminates *vars* from the system of equations
`Reduce`$[\{l_1 == r_1, \ldots\}, vars]$	simplifies the system of equations and gives the possible solutions
`FindInstance`$[expr, vars]$	finds an instance of *vars* that make *expr* true

20.3.3 Linear Systems of Equations and Eigenvalue Problems

In 20.2.4, p. 1027, the notion of matrix and several operations with matrices were defined on the basis of lists. Mathematica applies these notions in the theory of systems of linear equations. In the following command m, n denote given integers, not variables.

$$P = \texttt{Array}[p, \{m, n\}] \tag{20.30}$$

defines a matrix of type (m, n) with elements $p_{ij} = \texttt{p}[[i, j]]$. Furthermore

$$X = \texttt{Array}[x, \{n\}] \text{ und } B = \texttt{Array}[b, \{m\}] \tag{20.31}$$

are n- or m-dimensional vectors. With these definitions the general system of linear homogeneous or inhomogeneous equations can be written in the form (see 4.5.2, p. 308)

$$P\,.\,X == B \qquad P\,.\,X == 0 \quad \text{or} \quad \texttt{Thread}[\texttt{P . X} == \texttt{B}] \qquad \texttt{Thread}[\texttt{P . X} == \texttt{0}] \tag{20.32}$$

1. Special Case $n = m$, $\det P \neq 0$

In the special case $n = m$, $\det P \neq 0$, the system of inhomogeneous equations has a unique solution, which can be determined directly by

$$X = \texttt{Inverse}[P].\, B \tag{20.33}$$

Mathematica can handle such systems of up to ca. 5000 unknowns in a reasonable time, depending on the computer system. An equivalent, but much faster solution is obtained by `LinearSolve`$[P, B]$.

2. General Case

With the commands `LinearSolve` and `NullSpace`, all the possible cases can be handled as discussed in 4.5.2, p. 308, i.e., it can be determined first if any solution exists, and if it does, then it is calculated. Now, some of the examples from 4.5.2, p. 308ff. will be discussed.

■ **A:** The example in 4.5.2.1, **2.**, p. 310, is a system of homogeneous equations

$$\begin{aligned} x_1 - x_2 + 5x_3 - x_4 &= 0 \\ x_1 + x_2 - 2x_3 + 3x_4 &= 0 \\ 3x_1 - x_2 + 8x_3 + x_4 &= 0 \\ x_1 + 3x_2 - 9x_3 + 7x_4 &= 0 \end{aligned}$$

which has non-trivial solutions. These solutions are the linear combinations of the basis vectors of the null space of matrix p. It is the subspace of the n-dimensional vector space which is mapped into the zero by the transformation p. A basis for this space can be generated by the command `NullSpace`$[p]$. With the input

$$In[1] := p = \{\{1, -1, 5, -1\}, \{1, 1, -2, 3\}, \{3, -1, 8, 1\}, \{1, 3, -9, 7\}\}$$

a matrix, whose determinant is actually zero is defined (check it by calculating $\texttt{Det}[p]$). Now

$\textit{In[2]} := \texttt{NullSpace}[p]$ and $\textit{Out[2]} = \{\{-1,-2,0,1\},\{-3,7,2,0\}\}$

is displayed, a list of two linearly independent vectors of four-dimensional space, which form a basis for the two-dimensional null-space of matrix p. An arbitrary linear combination of these vectors is also in the null-space, so it is a solution of the system of homogeneous equations. This solution coincides with the solution found in 4.5.2.1, **2.**, p. 310.

■ **B:** Consider the example **A** in 4.5.2.1, **2.**, p. 309,

$$\begin{aligned} x_1 - 2x_2 + 3x_3 - x_4 + 2x_5 &= 2 \\ 3x_1 - x_2 + 5x_3 - 3x_4 - x_5 &= 6 \\ 2x_1 + x_2 + 2x_3 - 2x_4 - 3x_5 &= 8 \end{aligned}$$

with matrix $m1$ of type $(3,5)$, and vector $b1$

$\textit{In[1]} := m1 = \{\{1,-2,3,-1,2\},\{3,-1,5,-3,-1\},\{2,1,2,-2,-3\}\};$

$\textit{In[2]} := b1 = \{2,6,8\};$

For the command

$\textit{In[3]} := \texttt{LinearSolve}[m1,b1]$ the response is

$\texttt{LinearSolve}$: : nosol: Linear equation encountered which has no solution.

The input appears as output.

■ **C:** According to example **B** from 4.5.2.1, **1.**, p. 309,

$$\begin{aligned} x_1 - x_2 + 2x_3 &= 1 \\ x_1 - 2x_2 - x_3 &= 2 \\ 3x_1 - x_2 + 5x_3 &= 3 \\ -2x_1 + 2x_2 + 3x_3 &= -4 \end{aligned}$$

the input is

$\textit{In[1]} := m2 = \{\{1,-1,2\},\{1,-2,-1\},\{3,-1,5\},\{-2,2,3\}\};$

$\textit{In[2]} := b2 = \{1,2,3,-4\};$

To learn how many equations have independent left-hand sides, one calls

$\textit{In[3]} := \texttt{RowReduce}[m2]; \longrightarrow \textit{Out[3]} = \{\{1,0,0\},\{0,1,0\},\{0,0,1\},\{0,0,0\}\}$

Then the input is

$$\textit{In[4]} := \texttt{LinearSolve}[m2,b2]; \longrightarrow \textit{Out[4]} = \{\frac{10}{7},-\frac{1}{7},-\frac{2}{7}\}$$

The answer is the known solution.

3. Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors of matrices are defined in 4.6, p. 314. Mathematica provides the possibility of determining eigenvalues and eigenvectors by special commands. So, the command $\texttt{Eigenvalues}[m]$ produces a list of eigenvalues of a square matrix m, $\texttt{Eigenvectors}[m]$ creates a list of the eigenvectors of m, whereas $\texttt{Eigensystem}[m]$ gives both. If $\texttt{N}[m]$ is substituted instead of m, then one gets the numerical eigenvalues. In general, if the order of the matrix is greater than four ($n > 4$), then no algebraic expression can be obtained, since the characteristic polynomial has degree higher than four. In this case, one should ask for numerical values.

■ $\textit{In[1]} := h = \texttt{Table}[1/(i+j-1),\{i,5\},\{j,5\}]$

This generates a five-dimensional so-called Hilbert matrix.

$$\textit{Out[1]} = \{\{1,\frac{1}{2},\frac{1}{3},\frac{1}{4},\frac{1}{5}\},\{\frac{1}{2},\frac{1}{3},\frac{1}{4},\frac{1}{5},\frac{1}{6}\},\{\frac{1}{3},\frac{1}{4},\frac{1}{5},\frac{1}{6},\frac{1}{7}\},\{\frac{1}{4},\frac{1}{5},\frac{1}{6},\frac{1}{7},\frac{1}{8}\},\{\frac{1}{5},\frac{1}{6},\frac{1}{7},\frac{1}{8},\frac{1}{9}\}\}$$

With the command

In[2] := `Eigenvalues`[h]

the answer (which may be not useful) is

{`Root[-1 + 307505 #1 - 1022881200 #1`^{2} $+ \ldots$}

But with the command

In[3] := `Eigenvalues[N[h]]` one gets

Out[3] $= \{1.56705, 0.208534, 0.0114075, 0.000305898, 3.28793 \times 10^{-6}\}$

20.3.4 Differential and Integral Calculus

The notation of the derivative as a functional operator was introduced in 20.2.8, p. 1032. Mathematica provides several possibilities to apply the operations of analysis, e.g., determination of the derivative of arbitrarily high order, of partial derivatives, of the complete differential, determination of indefinite and definite integrals, series expansion of functions, and also solutions of differential equations.

20.3.4.1 Calculation of Derivatives

1. Differentiation Operator

The differentiation operator (see 20.2.8, p. 1032) is `Derivative`. Its complete form is

`Derivative`[$n_1, n_2, \ldots$] (20.34)

The arguments say how many times the function is to be differentiated with respect to the current variables. In this sense, it is an operator of partial differentiation. Mathematica tries to represent the result as a pure function.

2. Differentiation of Functions

The differentiation of a given function can be performed in a simplified manner with the operator `D`. With `D`[$f[x], x$], the derivative of the function f at the argument x will be determined.
`D` belongs to a group of differential operations, which are enumerated in **Table 20.12**.

Table 20.12 Operations of differentiation

`D`[$f[x], \{x, n\}$]	yields the n-th derivative of function $f(x)$ with respect to x
`D`[$f, \{x_1, n_1\}, \{x_2, n_2\}, \cdots$]	multiple derivatives, n_i-th derivative with respect to x_i $(i = 1, 2, \cdots)$
`Dt`[f]	the complete differential of the function f
`Dt`[f, x]	the complete differential $\dfrac{df}{dx}$ of the function f
`Dt`[$f, x_1, x_2, \ldots$]	the complete differential of a function of several variables

■ **A** : *In[1]* := `D[Sqrt[`x^3 `Exp[`$4x$`] Sin[`x`]], `x`]`

Out[1] $= \dfrac{\mathrm{E}^{4x}\, x^3\, \mathtt{Cos}[x] + 3\,\mathrm{E}^{4x}\, x^2 \mathtt{Sin[x]} + 4\mathrm{E}^{4x}\, x^3\, \mathtt{Sin}[x]}{2\sqrt{\mathrm{E}^{4x}\, x^3\, \mathtt{Sin}[x]}}$

■ **B** : *In[1]* := `D`$[(2x+1)^{3x}, x] \longrightarrow$ *Out[1]* $= (1+2\,x)^{3x}\left(\dfrac{6x}{1+2\,x} + 3\mathtt{Log[1+2\,x]}\right)$

The command `Dt` results in the complete derivative or complete differential.

■ **C** : *In[1]* := `Dt`$[x^3 + y^3] \longrightarrow$ *Out[1]* $= 3x^2\mathtt{Dt}[x] + 3y^2\,\mathtt{Dt}[y]$

■ **D** : *In[1]* := `Dt`$[x^3 + y^3, x] \longrightarrow$ *Out[1]* $= 3x^2 + 3y^2\mathtt{Dt}[y, x]$

In this last example, Mathematica supposes y to be a function of x, which is not known, so it writes the second part of the derivative in a symbolic way. The preferable forms of writing are: $\texttt{D}[x[t^3]+y[t]^3,t]$ and $\texttt{D}[x^3+y[x]^3,x]$ showing explicitly the independent variables.

If Mathematica finds a symbolic function while calculating a derivative, it leaves it in this general form, and expresses its derivative by f'.

■ **E** : *In[1]* := $\texttt{D}[x\ \texttt{f}[x]^3,x]$ ⟶ *Out[1]* = $\texttt{f}[x]^3+3x\texttt{f}[x]^2\texttt{f'}[x]$

Mathematica knows the rules for differentiation of products and quotients, it knows the chain rule, and it can apply these rules formally:

■ **F** : *In[1]* := $\texttt{D}[\texttt{f}[\texttt{u}[x]],\ x]$ ⟶ *Out[1]* = $\texttt{f}'[\texttt{u}[x]]\ \texttt{u}'[x]$

■ **G** : *In[1]* := $\texttt{D}[\texttt{u}[x]/\texttt{v}[x],x]$ ⟶ *Out[1]* = $\dfrac{\texttt{u}'[x]}{\texttt{v}[x]}-\dfrac{\texttt{u}[x]\ \texttt{v}'[x]}{\texttt{v}[x]^2}$

20.3.4.2 Indefinite Integrals

With the command $\texttt{Integrate}[f,x]$, Mathematica tries to determine the indefinite integral $\int f(x)\,dx$.

If Mathematica knows the integral, it gives it without the integration constant. Mathematica supposes that every expression not containing the integration variable does not depend on it.

In general, Mathematica finds an indefinite integral, if there exists one which can be expressed in closed form by elementary functions, such as rational functions, exponential and logarithmic functions, trigonometric and their inverse functions, etc. If Mathematica cannot find the integral, then it returns the original input. Mathematica knows some special functions which are defined by non-elementary integrals, such as the elliptic functions, and some others.

To demonstrate the possibilities of Mathematica, some examples will be shown, which are discussed in 8.1, p. 480ff.

1. Integration of Rational Functions

(see also 8.1.3.3, p. 485ff.)

■ **A** : *In[1]* := $\texttt{Integrate}[(2x+3)/(x^3+x^2-2x),x]$

Out[1] = $\dfrac{5}{3}\texttt{Log}[-1+x]-\dfrac{3\,\texttt{Log}[x]}{2}-\dfrac{1}{6}\texttt{Log}[2+x]$

■ **B** : *In[1]* := $\texttt{Integrate}[(x^3+1)/(x(x-1)^3),x]$

Out[1] = $-\dfrac{1}{(-1+x)^2}-\dfrac{1}{-1+x}+2\,\texttt{Log}[-1+x]-\texttt{Log}[x]$ (20.35)

On the monitor can be seen in the left corner of the next cell a plus sign. Clicking on it one may choose either the free-from input or the Wolfram-Alpha query. If one types the integral into one of these then there is given the possibility to have a look at all the details of the process of integration.

2. Integration of Trigonometric Functions

(see also 8.1.5, p. 491ff.)

■ **A:** The example **A** in 8.1.5.2, p. 492, with the integral $\int \sin^2 x\cos^5 x\,dx$ is calculated (substitution is done by the program automatically, if needed):

In[1] := $\texttt{Integrate}[\texttt{Sin}[x]^2\texttt{Cos}[x]^5,x]$

Out[1] = $\dfrac{5\,\texttt{Sin}[x]}{64}-\dfrac{1}{192}\texttt{Sin}[3\,x]-\dfrac{3}{320}\,\texttt{Sin}[5\,x]-\dfrac{1}{448}\texttt{Sin}[7\,x]$

■ **B:** The example **B** in 8.1.5.2, p. 492, with the integral $\int \frac{\sin x}{\sqrt{\cos x}}\,dx$ is calculated:

In[1] := Integrate[Sin[x]/Sqrt[Cos[x]], x] ⟶ *Out[1]* = $-2\sqrt{\text{Cos}[x]}$

Remark: In the case of non-elementary integrals Mathematica may do nothing.

■ *In[1]* := $\int x^x \text{dx}$ ⟶ *Out[1]* = $\int \text{x}^\text{x}\text{dx}$

20.3.4.3 Definite Integrals and Multiple Integrals

1. Definite Integrals

With the command Integrate[$f, \{x, x_a, x_e\}$], Mathematica can evaluate the definite integral of the function $f(x)$ with a lower limit x_a and upper limit x_e.

■ **A** : *In[1]* := Integrate[Exp[$-x^2$], $\{x, 0,$ Infinity$\}$] ⟶ *Out[1]* = $\frac{\sqrt{\pi}}{2}$

(see **Table 21.8**, p. 1098, No. 25 for $a = 1$).

■ **B:** If the input is

In[1] := Integrate[$\frac{1}{x^2}, \{x, -1, 1\}$] one gets

Out[1] = Integrate::idiv: "Integral of $\frac{1}{x^2}$ does not converge on {-1,1}. "

In the calculation of definite integrals one should be careful. If the properties of the integrand are not known, then it is recommended to ask for a graphical representation of the function in the considered domain before integration.

2. Multiple Integrals

Definite double integrals can be called by the command

Integrate[f[x, y], $\{x, x_a, x_e\}, \{y, y_a, y_e\}$] (20.36)

The evaluation is performed from right to left, so, first the integration is evaluated with respect to y. The limits y_a and y_e can be functions of x, which are substituted into the primitive function. Then the integral is evaluated with respect to x.

■ For the integral A, which calculates the area between a parabola and a line intersecting it twice, in 8.4.1.2, p. 524, one gets

In[1] := Integrate[$x\ y^2, \{x, 0, 2\}, \{y, x^2, 2x\}$] ⟶ *Out[1]* = $\frac{32}{5}$.

Also in this case, it is important to be careful with the discontinuities of the integrand. The domain of integration can also be specified with inequalities: Integrate[Boole[$x^2+y^2 \le 1, \{x, -1, 1\}, \{y, -1, 1\}$]] gives π.

20.3.4.4 Solution of Differential Equations

Mathematica can handle ordinary differential equations symbolically if the solution can be given in closed form. In this case, Mathematica gives the solution in general. The commands discussed here are listed in **Table 20.13**.

The solutions (see 9.1, p. 540) are represented as general solutions with the arbitrary constants $C[i]$. Initial values and boundary conditions can be introduced in the part of the list which contains the equation or equations. In this case a special solution is returned. As examples, two differential equations are solved here from 9.1.1.2, p. 542.

■ **A:** The solution of the differential equation $y'(x) - y(x)\tan x = \cos x$ is to be determined.

In[1] := DSolve[$y'[x] - y[x]$ Tan[x] == Cos[x], y, x]

Mathematica solves this equation, and gives the solution as a pure function with the integrations constant $C[1]$.

Table 20.13 Commands to solve differential equations

`DSolve`$[deq, y[x], x]$	solves the differential equation for $y[x]$ (if it is possible); $y[x]$ may be given in implicit form
`DSolve`$[deq, y, x]$	gives the solution of the differential equation in the form of a pure function
`DSolve`$[\{deq_1, deq_2, \ldots\}, y, x]$	solves a system of ordinary differential equations

$$\texttt{Out[1]} = \left\{\left\{y \to \texttt{Function[x, C[1] Sec[x] + Sec[x]} \left(\frac{x}{2} + \frac{1}{4}\texttt{Sin[2 x]}\right)\right]\right\}\right\}$$

If it is required to get the solution value $y[x]$, then Mathematica gives

$$\texttt{In[2]} := y[x]/.\ \%1 \longrightarrow \texttt{Out[2]} = \left\{\text{C}[1]\texttt{Sec}[x] + \texttt{Sec}[x]\left(\frac{x}{2} + \frac{1}{4}\texttt{Sin}[2\,x]\right)\right\}$$

One also could make the substitution for other quantities, e.g., for $y'[x]$ or $y[1]$. The advantage of using pure functions is obvious here.

■ **B:** The solution of the differential equation $y'(x)x(x - y(x)) + y^2(x) = 0$ (see 9.1.1.2, **2.**, p. 542) is to be determined.

$$\texttt{In[1]} := \texttt{DSolve}[y'[x]\ x(x - y[x]) + y[x]\texttt{\^{}}2 == 0, y[x], x]$$

$$\texttt{Out[1]} = \left\{\left\{y[x] \longrightarrow \texttt{-x ProductLog}[-\frac{E^{-\text{C}[1]}}{x}]\right\}\right\}$$

Here `ProductLog[z]` gives the principal solution for w in $z = we^w$. The solution of this differential equation was given in implicit form (see 9.1.1.2, **2.**, p. 542).

If Mathematica cannot solve a differential equation it returns the input without any comment. In such cases, or also, if the symbolic solution is too complicated, the solutions can be found by numerical solutions (see 19.8.4.2, **5.**, p. 1018). Also in the case of symbolic solutions of differential equations, like in the evaluation of indefinite integrals, the efficiency of Mathematica should not be overestimated. If the result cannot be expressed as an algebraic expression of elementary functions, the only way is to find a numerical solution.

Remark: Mathematica can solve some partial differential equations both symbolically and numerically, as well, even on complicated multidimensional domains.

20.4 Graphics with Mathematica

By providing routines for graphical representation of mathematical relations such as the graphs of functions, space curves, and surfaces in three-dimensional space, modern computer algebra systems provide extensive possibilities for combining and manipulating formulas, especially in analysis, vector calculus, and differential geometry, and they provide immeasurable help in engineering designing. Graphics is a special strength of Mathematica.

20.4.1 Basic Elements of Graphics

Mathematica builds graphical objects from built-in *graphics primitives*. These are objects such as points (`Point`), lines (`Line`) and polygons (`Polygon`) and properties of these objects such as thickness and color.

Mathematica has several options to specify the environment for graphics and how the graphical objects should be represented.

With the command `Graphics`[*list*], where *list* is a list of graphics primitives, Mathematica is called to generate a graphic from the listed objects. The object list can follow a list of options about the appearance of the representation.
With the input

In[1] := g = `Graphics`[{`Line`[{{0, 0}, {5, 5}, {10, 3}}], `Circle`[{5, 5}, 4], (20.37a)

`Text`[`Style`["Example","Helvetica",Bold,25], {5, 6}]}, `AspectRatio`–> `Automatic`] (20.37b)

a graphic is built from the following elements:

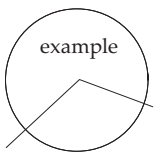

Figure 20.1

a) Broken line of two line segments starting at the point $(0, 0)$ through the point $(5, 5)$ to the point $(10, 3)$.

b) Circle with the center at $(5, 5)$ and radius 4.

c) Text with the content "Example", written in Helvetica font, boldface (the text appears centered with respect to the reference point $(5, 6)$).
With the call `Show`[g], Mathematica displays the figure **(Fig. 20.1)**.

Certain options might be previously specified. Here the option `AspectRatio` is set to `Automatic`.

By default Mathematica makes the ratio of the height to the width of the graph 1 : `GoldenRatio` (see e.g. 3.5.2.3,**3.**, p. 194). It corresponds to a relation between the extension in the x direction to the one in $1 : 1/1.618 = 1 : 0.618$. With this option the circle would be deformed into an ellipse. The value of the option `Automatic` ensures that the representation is not deformed.

20.4.2 Graphics Primitives

Mathematica provides the two-dimensional graphic objects enumerated in **Table 20.14**.
Besides these objects Mathematica provides further primitives to control the appearance of the representation, the graphics commands. They specify how graphic objects should be represented. The commands are listed in **Table 20.15**.
There is a wide scale of colors to choose from but their definitions are not discussed here.

Table 20.14 Two-dimensional graphic objects

`Point`[{x, y}]	point at position x, y
`Line`[{{x_1, y_1}, {x_2, y_2}, ...}]	broken line through the given points
`Rectangle`[{x_{lu}, y_{lu}}, {x_{ro}, y_{ro}}]	shaded rectangle with the given coordinates left-down, right-up
`Polygon`[{{x_1, y_1}, {x_2, y_2}, ...}]	shaded polygon with the given vertices
`Circle`[{x, y}, r]	circle with radius r around the center x, y
`Circle`[{x, y}, r, {α_1, α_2}]	circular arc with the given angles as limits
`Circle`[{x, y}, {a, b}]	ellipse with half-axes a and b
`Circle`[{x, y}, {a, b}, {α_1, α_2}]	elliptic arc
`Disk`[{x, y}, r], `Disk`[{x, y}, {a, b}]	shaded circle or ellipse
`Text`[*text*, {x, y}]	writes *text* centered to the point x, y

Table 20.15 Graphics commands

`PointSize`[a]	a dot is drawn with radius a as a fraction of the total picture
`AbsolutePointSize`[b]	denotes the absolute radius b of the dot (measured in American pt (0.3515 mm))
`Thickness`[a]	draws lines with relative thickness a
`AbsoluteThickness`[b]	draws lines with absolute thickness b (also in pt)
`Dashing`[{$a_1, a_2, a_3, \ldots$}]	draws a line as a sequence of stripes with the given length (in relative measure)
`AbsoluteDashing`[{$b_1, b_2, \ldots$}]	the same as the previous one but in absolute measure
`GrayLevel`[p]	specifies the level of shade ($p = 0$ is for black, $p = 1$ is for white)

20.4.3 Graphical Options

Mathematica provides several graphical options which have an influence on the appearance of the entire picture. **Table 20.16** gives a selection of the most important commands. For a detailed explanation, see [20.16].

Table 20.16 Some graphical options

`AspectRatio` –> w	sets the ratio w of height and width. `Automatic` determines w from the absolute coordinates; the default setting is $w = 1$: `GoldenRatio`
`Axes` –> `True`	draws coordinate axes
`Axes` –> `False`	does not draw coordinate axes
`Axes` –> {`True`, `False`}	shows only the x-axis
`Frame` –> `True`	shows frames
`GridLines` –> `Automatic`	shows grid lines
`AxesLabel` –> $\{x_{symbol}, y_{symbol}\}$	denotes axes with the given symbols
`Ticks` –> `Automatic`	denotes scaling marks automatically; with `None` they can be suppressed
`Ticks` –> $\{\{x_1, x_2, \ldots\}, \{y_1, y_2, \ldots\}\}$	scaling marks are placed at the given nodes

20.4.4 Syntax of Graphical Representation

20.4.4.1 Building Graphic Objects

If a graphic object is to be built from primitives, then first a list of the corresponding objects with their global definition should be given in the form

$$\{object_1, object_2, \ldots\}, \tag{20.38a}$$

where the objects themselves can be lists of graphic objects. Let object1 be, e.g.,

$$\textit{In[1]} := o1 = \{\texttt{Circle}[\{5,5\},\{5,3\}], \texttt{Line}[\{\{0,5\},\{10,5\}\}]\}$$

and corresponding to it

$$\textit{In[2]} := o2 = \{\texttt{Circle}[\{5,5\},3]\}$$

as in **Fig.20.1**. If a graphic object, e.g., $o2$, is to be provided with certain graphical commands, then it should be written into one list with the corresponding command

$$\textit{In[3]} := o3 = \{\texttt{Thickness}[0.01], o2\}$$

This command is valid for all objects in the *corresponding* braces, and also for nested ones, but not for the objects outside of the braces of the list.
From the generated objects two different graphic lists are defined:

$$\textit{In[4]} := g1 = \texttt{Graphics}[\{o1, o2\}] \;;\; g2 = \texttt{Graphics}[\{o1, o3\}]$$

which differs only in the second object by the thickness of the circle. The call

$$\texttt{Show}[g1] \text{ and } \texttt{Show}[g2, \texttt{Axes} \texttt{ –> } \texttt{True}] \tag{20.38b}$$

givs the pictures represented in **Fig. 20.2**.

In the call of the picture in **Fig. 20.2b**, the option `Axes –> True` was activated. This results in the representation of the axes with marks on them chosen by Mathematica and with the corresponding scaling.

20.4.4.2 Graphical Representation of Functions

Mathematica has special commands for the graphical representation of functions. With

$$\texttt{Plot}[\texttt{f}[x], \{x, x_{min}, x_{max}\}] \tag{20.39}$$

the function f is represented graphically in the domain between $x = x_{min}$ and $x = x_{max}$. Mathematica produces a function table by internal algorithms and reproduces the graphics following from this table by graphics primitives.

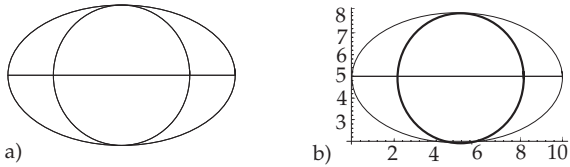

Figure 20.2

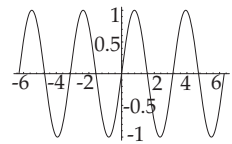

Figure 20.3

■ If the function $x \mapsto \sin 2x$ is to be graphically represented in the domain between -2π and 2π, then the input is

In[1] := `Plot[Sin[2x], {x, -2Pi, 2Pi}]`.

Mathematica produces the curve shown in **Fig. 20.3**.

It is obvious that Mathematica uses certain default graphical options in the representation as mentioned in 20.4.1, p. 1045. So, the axes are automatically drawn, they are scaled and denoted by the corresponding x and y values. In this example, the influence of the default `AspectRatio` can be seen. The ratio of the total width to the total height is 1 : 0.618.

With the command `InputForm[%]` the whole representation of the graphic objects can be shown. For the previous example one gets:

```
Graphics[{{{}, {}, {Directive[Opacity[1.], RGBColor[0.368417, 0.506779, 0.709798],
AbsoluteThickness[1.6]], Line[{{-6.283185050723043, 2.5645654335783057*^-7},
..., {6.283185050723043, -2.5645654335783057*^-7} }], {DisplayFunction -> Identity,
AspectRatio -> GoldenRatio^(-1), Axes -> {True, True}, AxesLabel -> {None, None},
AxesOrigin -> {0, 0}, DisplayFunction :> Identity,
Frame -> {{False, False}, {False, False}}, FrameLabel -> {{None, None},
{None, None}}, FrameTicks -> {{Automatic, Automatic}, {Automatic, Automatic}},
GridLines -> {None, None}, GridLinesStyle -> Directive[GrayLevel[0.5, 0.4]],
Method -> {"DefaultBoundaryStyle" -> Automatic, "ScalingFunctions" -> None},
PlotRange -> {{-2*Pi, 2*Pi}, {-0.9999996654606427, 0.9999993654113022}},
PlotRangeClipping -> True, PlotRangePadding -> {{Scaled[0.02], Scaled[0.02]},
{Scaled[0.05], Scaled[0.05]}}, Ticks -> {Automatic, Automatic}}]
```

Consequently, the graphic object consists of a few sublists. The first one contains the graphics primitive `Line` (slightly modified), with which the internal algorithm connects the calculated points of the curve by lines. The second sublist contains the options needed by the given graphic. These are the default options. If the picture is to be altered at certain positions, then the new settings in the `Plot` command must be set after the main input. With

In[2] := `Plot[Sin[2x], {x, -2Pi, 2Pi}, AspectRatio-> 1]` (20.40)

the representation would be done with equal length of axes x and y.

It is possible to give several options at the same time after each other.With the input

`Plot[{`$f_1[x], f_2[x], \ldots$`}, {`x, x_{min}, x_{max}`}]` (20.41)

several functions are shown in the same graphic. With the command

`Show[`*plot*, *options*`]` (20.42)

an earlier picture can be renewed with other options.With

`Show[GraphicsArray[`*list*`]]`, (20.43)

(with *list* as lists of graphic objects) pictures can be placed next to each other, under each other, or they can be arranged in matrix form.

20.4.5 Two-Dimensional Curves

A series of curves from the chapter on functions and their representations (see 2.1, p. 48ff.) is shown as examples.

20.4.5.1 Exponential Functions

A family of curves with several exponential functions (see 2.6.1, p. 72) is generated by Mathematica **(Fig. 20.4a)** with the following input:

In[1] := `f[x_] := 2^x; g[x_] := 10^x;`

In[2] := `h[x_] := (1/2)^x; j[x_] := (1/E)^x; k[x_] := (1/10)^x;`

These are the definitions of the considered functions. There is no need to define the function e^x, since it is built into Mathematica. In the second step the following graphics are generated:

In[3] := $p1$ = `Plot[{f[x], h[x]}, {x, -4, 4}, PlotStyle-> Dashing[{0.01, 0.02}]]`

In[4] := $p2$ = `Plot[{Exp[x], j[x]}, {x, -4, 4}]`

In[5] := $p3$ = `Plot[{g[x], k[x]}, {x, -4, 4}, PlotStyle-> Dashing[{0.005, 0.02, 0.01, 0.02}]]`

The whole picture **(Fig. 20.4a)** can be obtained by:

In[6] := `Show[{`$p1, p2, p3$`}, PlotRange-> {0, 18}, AspectRatio-> 1.2]`

The question of how to write text on the curves is not discussed here. This is possible with the graphics primitive `Text`.

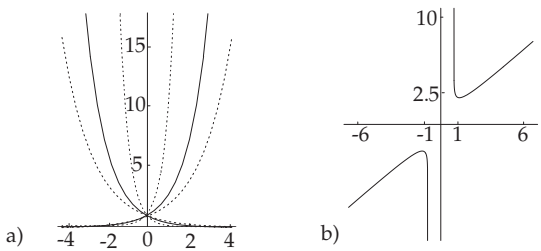

Figure 20.4

20.4.5.2 Function $y = x + \operatorname{Arcoth} x$

Considering the properties of the function $\operatorname{Arcoth} x$ discussed in 2.10, p. 93, the function $y = x + \operatorname{Arcoth} x$ can be graphically represented in the following way:

In[1] := $f1$ = `Plot[x + ArcCoth[x], {x, 1.000000000005, 7}]`

In[2] := $f2$ = `Plot[x + ArcCoth[x], {x, -7, -1.000000000005}]`

In[3] := `Show[{`$f1, f2$`}, PlotRange-> {-10, 10}, AspectRatio-> 1.2, Ticks->`

`{{{-6, -6}, {-1, -1}, {1, 1}, {6, 6}}, {{2.5, 2.5}, {10, 10}}}, AxesOrigin- > 0, 0]`

The high precision of the x values in the close neighborhood of 1 and -1 was chosen to get sufficiently large function values for the required domain of y. The result is shown in **Fig. 20.4b**.

20.4.5.3 Bessel Functions (see 9.1.2.6, 2., p. 562)

With the calls

In[1] := $bj0 =$ `Plot[{BesselJ[0, z], BesselJ[2, z], BesselJ[4, z]}, {z, 0, 10}, PlotLabel-> TraditionalForm[{BesselJ[0, z], BesselJ[2, z], BesselJ[4, z]}]` (20.44a)

In[2] := $bj1 =$ `Plot[{BesselJ[1, z], BesselJ[3, z], BesselJ[5, z]}, {z, 0, 10}, PlotLabel-> TraditionalForm[{BesselJ[1, z], BesselJ[3, z], BesselJ[5, z]}]]` (20.44b)

the graphics of the Bessel function $J_n(z)$ for $n = 0, 2, 4$ and $n = 1, 3, 5$ are generated, which are then represented by the call

In[3] := `GraphicsRow[{`$bj0, bj1$`}]]`

next to each other in **Fig. 20.5**.

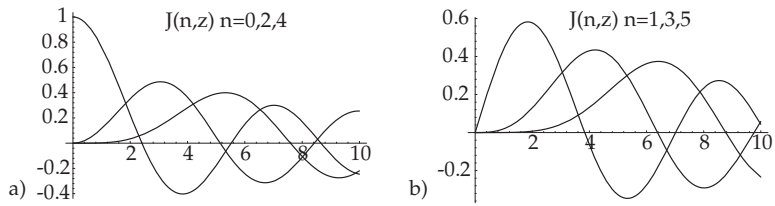

Figure 20.5

20.4.6 Parametric Representation of Curves

Mathematica has a special graphics command, with which curves given in parametric form can be graphically represented. This command is:

`ParametricPlot[{`$f_x(t), f_y(t)$`}, {`t, t_1, t_2`}]`. (20.45)

It provides the possibility of showing several curves in one graphic. A list of several curves must be given in the command. With the option `AspectRatio-> Automatic`, Mathematica shows the curves in their natural forms.

The parametric curves in **Fig. 20.6** are the Archimedean spiral (see 2.14.1, p. 105) and the logarithmic spiral (see 2.14.3, p. 106). They are represented with the input

In[1] := `ParametricPlot[{`t `Cos[`t`],` t `Sin[`t`]}, {`$t, 0,$ `3Pi}, AspectRatio-> Automatic]`

and

In[2] := `ParametricPlot[{Exp[0.1`t`] Cos[`t`], Exp[0.1`t`] Sin[`t`]}, {`$t, 0,$ `3Pi}, AspectRatio-> Automatic]`

With

In[3] := `ParametricPlot[{`$t -$ `2 Sin[`t`], 1 - 2 Cos[`t`]}, {`$t, -$`Pi, 11Pi}, AspectRatio-> 0.3]`

a trochoid (see 2.13.2, p. 102) is generated **(Fig. 20.7)**.

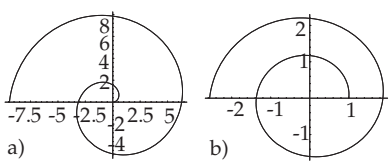

Figure 20.6

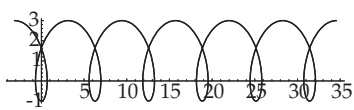

Figure 20.7

20.4.7 Representation of Surfaces and Space Curves

Mathematica provides the possibility of representing three-dimensional graphics primitives. Similarly to the two-dimensional case, three-dimensional graphics can be generated by applying different options. The objects can be represented and observed from different viewpoints and from different perspectives. Also the representation of curved surfaces in three-dimensional space, i.e., the graphical representation of functions of two variables, is possible. Furthermore it is possible to represent curves in three-dimensional space, e.g., if they are given in parametric form. For a detailed description of three-dimensional graphics primitives see [20.5], [20.16]. The introduction of these representations is similar to the two-dimensional case.

20.4.7.1 Graphical Representation of Surfaces

The command `Plot3D` in its basic form requires the definition of a function of two variables and the domain of these two variables:

$$\texttt{Plot3D}[\texttt{f}[x, y], \{x, x_a, x_e\}, \{y, y_a, y_e\}] \tag{20.46}$$

All options have the default setting.

■ For the function $z = x^2 + y^2$, with the input

In[1] := `Plot3D`$[x^2 + y^2, \{x, -5, 5\}, \{y, -5, 5\},$ `PlotRange`$\rightarrow \{0, 25\}]$

we get **Fig. 20.8a**, while **Fig. 20.8b** is generated by the command

In[2] := `Plot3D`$[(1 - \texttt{Sin}[x])\ (2 - \texttt{Cos}[2\ y]), \{x, -2, 2\}, \{y, -2, 2\}]$

For the paraboloid, the option `PlotRange` is given with the required z values, because the solid is cut at $z = 25$.

20.4.7.2 Options for 3D Graphics

The number of options for 3D graphics is large. In **Table 20.17**, only a few are enumerated, where options known from 2D graphics are not included. They can be applied in a similar sense. The option `ViewPoint` has special importance, by which very different observational perspectives can be chosen.

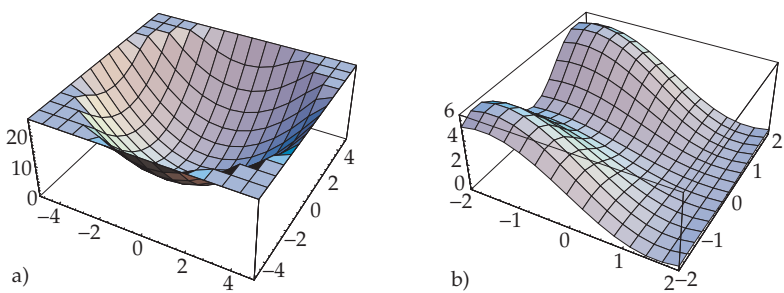

Figure 20.8

20.4.7.3 Three-Dimensional Objects in Parametric Representation

Similarly to 2D graphics, three-dimensional objects given in parametric representation can also be represented. With

$$\texttt{ParametricPlot3D}[\{f_x[t, u], f_y[t, u], f_z[t, u]\}, \{t, t_a, t_e\}, \{u, u_a, u_e\}] \tag{20.47}$$

a parametrically given surface is represented, with

$$\texttt{ParametricPlot3D}[\{f_x[t], f_y[t], f_z[t]\}, \{t, t_a, t_e\}] \tag{20.48}$$

a three-dimensional curve is generated parametrically.

Table 20.17 Options for 3D graphics

`Boxed`	default setting is `True`; it draws a three-dimensional frame around the surface
`HiddenSurface`	sets the non-transparency of the surface; default setting is `True`
`ViewPoint`	specifies the point (x, y, z) in space, from where the surface is observed. Default values are $\{1.3, -2.4, 2\}$
`Shading`	default setting is `True`; the surface is shaded; `False` yields white surfaces
`PlotRange`	$\{z_a, z_e\}$, $\{\{x_a, x_e\}, \{y_a, y_e\}, \{z_a, z_e\}\}$ can be chosen for the values `All`. Default is `Automatic`

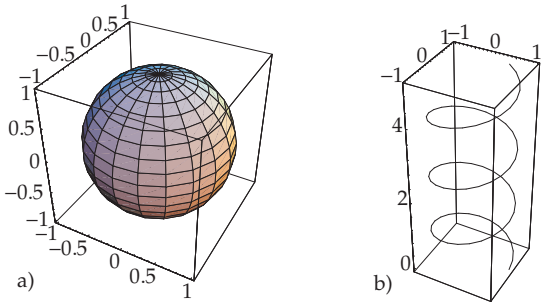

Figure 20.9

■ The objects in **Fig. 20.9a** and **Fig. 20.9b** are represented with the commands

$$\texttt{In[3]} := \texttt{ParametricPlot3D}[\{\texttt{Cos}[t]\ \texttt{Cos}[u], \texttt{Sin}[t]\ \texttt{Cos}[u], \texttt{Sin}[u]\}, \{t, 0, 2\texttt{Pi}\}\ \{u, -\texttt{Pi}/2, \texttt{Pi}/2\}] \tag{20.49a}$$

$$\texttt{In[4]} := \texttt{ParametricPlot3D}[\{\texttt{Cos}[t], \texttt{Sin}[t], t/4\}, \{t, 0, 20\}] \tag{20.49b}$$

Mathematica provides further commands by which density, and contour diagrams, bar charts and sector diagrams, and also a combination of different types of diagrams, can be generated.

■ The representation of the Lorenz attractor (see 17.2.4.3, p. 887) can easily be generated by Mathematica.

There is a series of recent developments most of which are not to be shown in a book. One can easily build a GUI (graphical user interface) to utilize interactive properties of the program. Most of the calculations are parallelized automatically, but functions such as `Parallelize` and `ParallelMap` provides the user to create his/her own parallel programs. The extremely fast graphic cards can be programmed at a very high level (as opposed to other languages) using such functions as `CUDALink`, `OpenCLFunctionLoad` etc. An extremely useful example of dynamic interactivity tool is `Manipulate` which in the simplest case shows you the parameter dependence of a family of curves. Working in the cloud or using the computer Raspberry Pi (which comes a free Mathematica license) should also not be unmentioned.