

1

Introduction to Computer Algebra

The goal of this chapter is to briefly describe what computer algebra is about, present a little history of computer algebra systems, give some examples of computer algebra usage, and discuss some advantages and limitations of this technological tool. We end with a sketch of the design of the Maple system.

The examples in this first chapter are sometimes of a rather advanced mathematical level. Starting with the second chapter, we give a detailed, step-by-step exposition of Maple as a symbolic calculator. The rest of the book does not depend much on this chapter. Thus, anyone who is so eager to learn Maple that he or she cannot wait any longer may skip this chapter and turn to it at any moment.

1.1 What is Computer Algebra?

Historically the verb “compute” has mostly been used in the sense of “computing with numbers.” Numerical computation is not only involved with basic arithmetic operations such as addition and multiplication of numbers, but also with more sophisticated calculations like computing numerical values of mathematical functions, finding roots of polynomials, solving systems of equations, and computing with matrices. It is essential in this type of computation that arithmetic operations are carried out on numbers and on numbers only. Furthermore, computations with numbers are in most cases not exact because in applications one is almost always dealing with floating-point numbers. Simple computations can be done with pencil and paper or with a pocket calculator; for large numerical computations, mainframes

serve as “number crunchers.” In the last fifty years numerical computation on computers flourished to such an extent that for many scientists mathematical computation on computers and numerical computation have become synonymous.

But mathematical computation has another important component, which we shall call *symbolic and algebraic computation*. In short, it can be defined as computation with symbols representing mathematical objects. These symbols may represent numbers like integers, rational numbers, real and complex numbers, and algebraic numbers, but they may also be used for mathematical objects like polynomials and rational functions, systems of equations, and even more abstractly for algebraic structures like groups, rings, and algebras, and elements thereof. Moreover, the adjective *symbolic* emphasizes that in many cases the ultimate goal of mathematical problem solving is expressing the answer in a closed formula or finding a symbolic approximation. By *algebraic* we mean that computations are carried out exactly, according to the rules of algebra, instead of using the approximate floating-point arithmetic. Examples of symbolic and algebraic computations are factorization of polynomials, differentiation, integration, and series expansion of functions, analytic solution of differential equations, exact solution of systems of equations, and simplification of mathematical expressions.

In the last thirty-five years great progress has been made regarding the theoretical background of symbolic and algebraic algorithms; moreover, tools have been developed to carry out mathematical computations on computers [26, 60, 91, 103, 154, 237]. This has lead to a new discipline, which is referred to by various names: symbolic and algebraic computation, symbolic computation, symbolic manipulation, formula manipulation, and computer algebra, to name a few. Tools for mathematical computation on a computer are given as many names as the discipline itself: symbolic computation programs, symbol crunchers, symbolic manipulation programs, symbolic calculators, and computer algebra systems. Unfortunately, the term “symbolic computation” is used in many different contexts, like logic programming and artificial intelligence in its broadest sense, which have very little to do with mathematical computation. To avoid misunderstanding, we shall henceforth adopt the term *computer algebra* and we shall speak of *computer algebra systems*.

1.2 Computer Algebra Systems

In this section, we shall give a very short, incomplete, and subjective overview of present-day computer algebra systems. For a more thorough overview we refer to [56, 108, 234] and to WWW-servers dedicated towards computer algebra [242]. Computer algebra systems can be conveniently

divided into two categories: *special purpose systems* and *general purpose systems*.

Special purpose systems are designed to solve problems in one specific branch of physics and mathematics. Some of the best-known special purpose systems used in physics are SCHOONSCHIP ([218] high-energy physics), CAMAL ([12] celestial mechanics), and SHEEP and STENSOR ([86, 133, 170] general relativity). Examples of special purpose systems in the mathematical arena are Cayley and GAP ([39, 42, 90] group theory), PARI, SIMATH, and KANT ([67, 132, 190, 196] number theory), CoCoA ([45, 101, 156] commutative algebra), Macaulay and SINGULAR ([110, 111, 112] algebraic geometry and commutative algebra), and LiE ([59] Lie theory). Our interest will be in the general purpose system Maple [173, 180, 181], but the importance of special purpose systems should not be underestimated: they have played a crucial role in many scientific areas [26, 41, 135]. Often they are more handsome and efficient than general purpose systems because of their use of special notations and data structures, and because of their implementation of algorithms in a low-level programming language.

General purpose systems please their users with a great variety of data structures and mathematical functions, trying to cover as many different application areas as possible (q.v., [122]). The oldest general purpose computer algebra systems still in use are MACSYMA [172] and REDUCE [124]. Both systems were born in the late sixties and were implemented in the programming language LISP. MACSYMA is a full-featured computer algebra system with a wide range of auxiliary packages, but its demands on computer resources are rather high and it is only available on a limited number of platforms. The release of the PC version of MACSYMA in 1992 has caused a revival of this system. REDUCE began as a special purpose program for use in high-energy physics, but gradually transformed into a general purpose system. Compared to MACSYMA the number of user-ready facilities in REDUCE is modest, but on the other hand it is a very open system (the complete source code is distributed!) making it easily extensible and modifiable. REDUCE is still under active development: REDUCE 3.7 is from 1999. It runs on a very wide range of computers and is well-documented.

In the eighties, MuMATH [239] and its successor DERIVE [157, 216] were the first examples of compact non-programmable symbolic calculators, designed for use on PC-type computers. DERIVE has a friendly menu-driven interface with graphical and numerical features. Considering its compactness and the limitations of DOS computers, DERIVE offers an amazing amount of user-ready facilities. Version 5 of 2000 also has limited programming facilities. Many of DERIVE's features have been built into the TI-92 calculator (Texas Instruments, 1995) and other high-end

calculators, thus making computer algebra available at small size computers. It also forms the computer algebra kernel of the computer learning environment called ‘TI Interactive!’ (Texas Instruments, 2000).

Most modern computer algebra systems are implemented in the programming language C. This language allows developers to write efficient, portable computer programs that really exploit the platforms for which they are designed. Many of these computer algebra systems work on a variety of computers, from supercomputers down to desktop computers.

In §1.6 we shall sketch the design of Maple [173, 180, 181]. Another notable modern general purpose system is *Mathematica* [238]. *Mathematica* was the first system in which symbolics, numerics, and graphics were incorporated in such a way that it could serve as a user-friendly environment for doing mathematics. There exists on most platforms the notebook interface, which is a tool (comparable with Maple worksheets) to create a structured text in which ordinary text is interwoven with formulas, programs, computations, and graphics. Another feature of *Mathematica* is the well-structured user-level programming language. With the publicity and marketing strategy that went into the production of *Mathematica*, commerce has definitely made its entry into the field of computer algebra, accompanied by less realistic claims about capabilities (q.v., [215]). On the positive side, the attention of many scientists has now been drawn to computer algebra and to the use of computer algebra tools in research and education. Another advantage has been the growing interest of developers of computer algebra systems in friendly user interfaces, good documentation, and ample user support. A wealth of information and user’s contributions can be found at the WWW server of Wolfram Research, Inc., and more specifically at the electronic library MathSource [174]. Version 4 of the system, launched in 1999, came with a new front end and with many new mathematical features. Later releases of *Mathematica* offer Internet facilities and connectivity with other programming languages.

The aforementioned general purpose systems manipulate formulae if the entire formula can be stored inside the main memory of the computer. This is the only limit to the size of formulae. The symbolic manipulation program FORM [128, 188, 225, 227] has been designed to deal with formulae of virtually infinite size (q.v., [226]). On the other hand, the size of the set of instructions in FORM is somewhat limited.

Magma [23, 24, 25, 43] is the successor of Cayley [39, 42], released in 1994, and designed around the algebraic notions of structure and morphism. Its aim is to support computation in algebra, number theory, geometry and algebraic combinatorics. This is achieved through the provision of extensive machinery for groups, rings, modules, algebras, geometric structures and finite incidence structures (designs, codes, graphs). Two basic design principles are the strong typing scheme derived from modern algebra whereby

types correspond to algebraic structures and the integration of algorithmic and database knowledge.

MuPAD [88, 100] stands for Multi Processing Algebra Data Tool. It is a system for symbolic and numeric computation, parallel mathematical programming and mathematical visualization. MuPAD is freely distributed for educational and non-commercial use from the website www.mupad.de. It is still in active development at the University of Paderborn in cooperation with SciFace Software GmbH & Co. KG and its mathematical contents is growing.

A portable system for parallel symbolic computation through Maple exists as well: it is called `||MAPLE||` (speak: parallel MAPLE) [209]. The system is built as an interface between the parallel declarative language Strand [84] and the sequential computer algebra system Maple, thus providing the elegance of Strand and the powerfulness of the existing sequential algorithms in Maple. More recently, a Java-based package for writing parallel programs in Maple and executing them on networks of computers has been developed at RISC Linz. It is called ‘Distributed Maple’ [208].

Last (but not least) in the row is AXIOM [70, 71, 140, 219]. It is a powerful general purpose system developed in the eighties at the IBM Thomas J. Watson Research Laboratory under the name of “Scratchpad.” In contrast to most other general purpose systems, which only allow calculations in a specific algebraic domain, e.g., the field of rational numbers or the ring of polynomials over the integers, AXIOM allows its users to define and handle distinct types of algebraic structures. But alas, this computer algebra system died in 2001, when NAG released the last version 2.3 to existing customers.

1.3 Some Properties of Computer Algebra Systems

Computer algebra systems differ from one another, but they share many properties. We shall illustrate common properties with examples from Maple.

Computer algebra systems are *interactive* programs that, in contrast to numerical computer programs, allow mathematical computations with *symbolic* expressions. Typically, the user enters a mathematical expression or an instruction, which the computer algebra system subsequently tries to execute. Given the result of the computation the user may enter a new instruction. This may lead to a fruitful computer algebra session. As an easy example of the use of Maple, we shall compute the stationary point of the real function

$$x \mapsto \arctan \left(\frac{2x^2 - 1}{2x^2 + 1} \right),$$

as well as the value of the function at this point. As we shall see later on in this section, Maple can compute the minimum on its own. Here, we only use the example to show some of the characteristics of computer algebra systems. Below is a screen dump of a complete work session with Maple 8, on a PC running the worksheet interface.

Let us take a closer look at this example. When Maple is started by choosing the Maple command from the appropriate menu or by double-clicking the corresponding icon on the desktop, an empty worksheet appears, except that the system prints the greater-than symbol “>” on the first line in order to prompt the user to enter an instruction. The symbol “>” is called the *Maple prompt*.

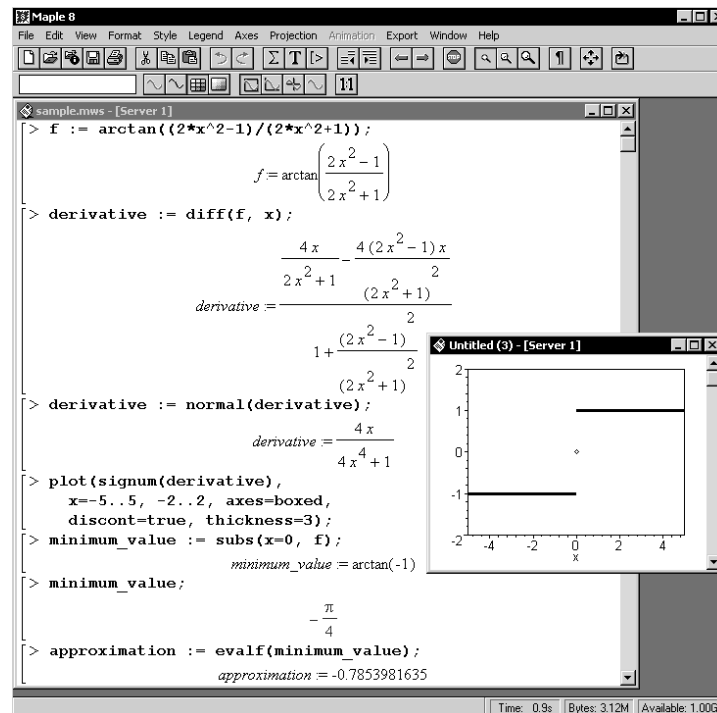


Figure 1.1. The Maple environment with a worksheet.

In the first command we enter the formula f , ending the input line with a semicolon, and pressing the RETURN key. The last two key strokes signal Maple to start to work. In this case, the formula is shown in two-dimensional mathematical notation of textbook quality. What strikes one most is that the system allows the use of symbols like x . In most numerical programming languages this would immediately cause an error; but not in systems for *symbolic* computations!

Each time Maple has executed an instruction, it prints the prompt and waits for another command. We decide to consider f as a function, **differen**tiate it, and assign the result to the variable called *derivative*. Maple's answer is a rather complicated expression. So, we **normal**ize the rational function. The answer is a simple expression of which the sign can be plotted. From this plot we immediately conclude that the original function has a minimum at $x = 0$. The minimum value $-\frac{1}{4}\pi$ is obtained by **subst**itution of $x = 0$ in f . We obtain an approximate floating-point result by use of the procedure **evalf**. The name **evalf** — short for “**eval**uate using **f**loating-point arithmetic” — is already the fourth example that shows Maple's general philosophy in choosing names: Use a short, easy-to-remember name for a procedure that describes its functionality. In addition to this, we have given meaningful names to variables, which describe their use.

We see that Maple leaves it to us to find our way through the computation. We must decide, on the basis of the second result, to try and find a less complicated formula for the derivative. One may wonder why the system itself does not perform this more or less obvious simplification. But remember, it is not always clear when and how to simplify. In many cases more than one simplification is possible and it is the mathematical context that actually determines which simplification is appropriate. For example, the rational expression

$$\frac{(x^2 - 1)(x^2 - x + 1)(x^2 + x + 1)}{(x - 1)^6}$$

can be transformed into the compact expression

$$\frac{x^6 - 1}{(x - 1)^6},$$

but also into a form suitable for integration, viz.,

$$1 + \frac{6}{(x - 1)^5} + \frac{15}{(x - 1)^4} + \frac{20}{(x - 1)^3} + \frac{15}{(x - 1)^2} + \frac{6}{x - 1}.$$

Another problem with automatic simplification is that in many computations one cannot predict the size and shape of the results and therefore must be able to intervene at any time. A procedure that works fine in one case might be a bad choice in another case. For example, one might think that it is always a good idea to factorize an expression. For example, the factorization of

$$x^8 + 8x^7 + 28x^6 + 56x^5 + 70x^4 + 56x^3 + 28x^2 + 8x + 1$$

is

$$(x + 1)^8.$$

However (surprise!) apart from being expensive, the factorization of the relatively simple

$$x^{26} + x^{13} + 1$$

yields

$$(x^{24} - x^{23} + x^{21} - x^{20} + x^{18} - x^{17} + x^{15} - x^{14} + x^{12} - x^{10} + x^9 - x^7 + x^6 - x^4 + x^3 - x + 1)(x^2 + x + 1).$$

For these reasons, Maple only applies automatic simplification rules when there is no doubt about which expression is simpler: $x + 0$ should be simplified to x , $3x$ is simpler than $x + x + x$, x^3 is better than $x \times x \times x$, and $\sin(\pi)$ should be simplified to 0. Any other simplification is left to the user's control; Maple only provides the tools for such jobs.

Automatic simplification sometimes introduces loss of mathematical correctness. For example, the automatic simplification of $0 \times f(1)$ to 0 is not always correct. An exception is the case $f(1)$ is undefined or infinity. The automatic simplification is only wanted if $f(1)$ is finite but difficult to compute. In cases like this, designers of computer algebra systems have to choose between rigorous mathematical correctness and usability/efficiency of their systems (q.v., [77]). In Maple and many other systems the scales sometimes tip to simplifications that are not 100% safe in every case.

Another remarkable fact in the first example is that Maple computes the exact value $-\frac{1}{4}\pi$ for the function in the origin and does not return an approximate value like 0.785398. This is a second aspect of computer algebra systems: the emphasis on *exact arithmetic*. In addition, computer algebra systems can carry out floating-point arithmetic in a *user-defined precision*. For example, in Maple the square $\tan^2(\pi/12)$ can be computed exactly, but the numerical approximation in 25-digits floating-point notation can also be obtained.

```
> real_number := tan(Pi/12)^2;
      real_number := tan( $\frac{\pi}{12}$ )^2
> real_number := convert(real_number, radical);
      real_number := (2 -  $\sqrt{3}$ )^2
> real_number := expand(real_number);
      real_number := 7 - 4 $\sqrt{3}$ 
> approximation := evalf[25](real_number);
      approximation := .071796769724490825890216
```

Computer algebra systems like Maple contain a substantial amount of built-in mathematical knowledge. This makes them good mathematical assistants. In calculus they differentiate functions, compute limits, and

compute series expansions. Integration (both definite and indefinite), is one of the highlights in computer calculus. Maple uses non-classical algorithms such as the Risch algorithm for integrating elementary functions [31, 98, 203], instead of the heuristic integration methods that are described in most mathematics textbooks.

With the available calculus tools, one can easily explore mathematical functions. In the Maple session below we shall explore the previously defined function f . The *sharp symbol* `#` followed by text is one of Maple's ways of allowing comments during a session; in combination with the names of variables and Maple procedures, this should explain the script sufficiently. If an input line is ended by a colon instead of a semicolon, then Maple does not print its results. Spaces in input commands are optional, but at several places we have inserted them to make the input more readable.

```
> f := arctan((2*x^2-1)/(2*x^2+1)); # enter formula

$$f := \arctan\left(\frac{2x^2 - 1}{2x^2 + 1}\right)$$

> df := normal(diff(f, x)); # differentiate f

$$df := 4 \frac{x}{4x^4 + 1}$$

> F := integrate(df, x); # integrate derivative

$$F := \arctan(2x^2)$$

> normal(diff(F-f, x)); # verify F = f + Pi/4

$$0$$

> eval(F-f, x=0);

$$\frac{\pi}{4}$$

> extrema(f, {}, x, stationary_points);

$$\{\arctan(-1)\}$$

> %; # extra evaluation

$$\left\{\frac{\pi}{4}\right\}$$

> stationary_points;

$$\{\{x = 0\}\}$$

> # this value was assigned by the call of 'extrema'
> solve(f=0, x); # compute the zero's of f

$$\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}$$

> series(f, x=0, 15);
```

```


$$-\frac{\pi}{4} + 2x^2 - \frac{8}{3}x^6 + \frac{32}{5}x^{10} - \frac{128}{7}x^{14} + O(x^{15})$$

> with(numapprox):
> pade(f, x, [6,4]);

$$\frac{-15\pi + 120x^2 - 36\pi x^4 + 128x^6}{12(5 + 12x^4)}$$

> chebpade(f, x, [2,2]);

$$\frac{-0.007904471007 T(0, x) + .4715125862 T(2, x)}{T(0, x) + .4089934686 T(2, x)}$$

> subs(simplify({T(0,x)=ChebyshevT(0,x),
> T(2,x)=ChebyshevT(2,x)}), %);

$$\frac{-.4794170572 + .9430251724 x^2}{.5910065314 + .8179869372 x^2}$$

> limit(f, x=infinity);

$$\frac{\pi}{4}$$

> series(f, x=infinity);

$$\frac{\pi}{4} - \frac{1}{2x^2} + O\left(\frac{1}{x^6}\right)$$

> plot([f, df], x=-5..5, linestyle=[0, 4], color=black,
> legend=["f", "f'"], title="graph of f and f'");

```

The graph is shown in Figure 1.2.

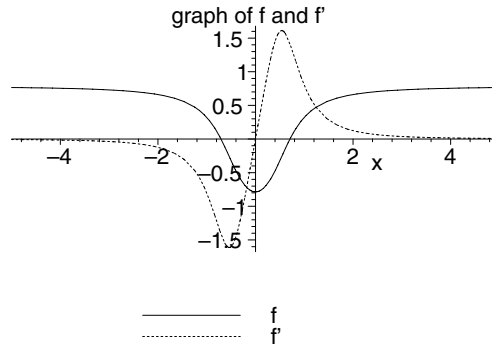


Figure 1.2. Graph of $(x, y) \mapsto \arctan\left(\frac{2x^2-1}{2x^2+1}\right)$ and its derivative.

Other impressive areas of computer algebra are polynomial calculus, the solution of systems of linear and nonlinear equations, the solution of recurrence equations and differential equations, calculations on matrices with numerical and symbolic coefficients, and tensor calculus. Various tools for manipulation of formulae are present: selection and substitution of

parts of expressions, restricted simplification, simplification rules, pattern matching, and so on. We may call computer algebra systems *mathematical expert systems* with which mathematical problems can be solved in a more productive and accurate way than with pencil and paper.

In addition to functioning as symbolic and algebraic calculators, most computer algebra systems can be used as *programming languages* for implementing new mathematical algorithms. By way of illustration we write a Maple program that computes the Bessel polynomials $y_n(x)$. Recall [116] that they can be recursively defined by

$$\begin{aligned} y_0(x) &= 1, \\ y_1(x) &= x + 1, \\ y_n(x) &= (2n - 1)x y_{n-1}(x) + y_{n-2}(x), \quad \text{for } n > 1. \end{aligned}$$

```
> Y := proc( n::nonnegint, x::name )
>   if n=0 then 1
>   elif n = 1 then x+1
>   else Y(n,x) := expand( (2*n-1)*x*Y(n-1,x) + Y(n-2,x) )
>   end if
> end proc:
> Y(5,z);
```

$$945 z^5 + 945 z^4 + 420 z^3 + 105 z^2 + 15 z + 1$$

The Maple programming language is reminiscent of Algol68 without declarations, but also includes several functional programming paradigms.

1.4 Advantages of Computer Algebra

The long-term goal of computer algebra is to automate as much as possible the mathematical problem solving process. Although present computer algebra systems are far from being automatic problem solvers, they are already useful, if not indispensable, tools in research and education. Of course, it takes time to familiarize oneself with a computer algebra system, but this time is well-spent. In this section, some of the more important reasons for learning and using a computer algebra system will be illustrated with Maple examples, a few of which are rather advanced mathematically. All computations will be carried out with Maple 8, on a PC running Windows 2000, with a 1.7 Ghz Pentium 4 processor having 512 MB main memory. This does not imply that the same results could not have been obtained on a much smaller machine, but the timings would be different.

The main advantage of a computer algebra system is its ability to carry out large algebraic computations. Although many calculations are straightforward standard manipulations that can be calculated with pencil and paper, the larger the formulae, the harder the work and the less the chance

of success. For this kind of computation a computer algebra system is an excellent tool. The next three examples demonstrate this.

The first example is one of the problems posed by R. Pavelle [194] as a challenge for computer algebra systems. The object is to prove that

$$\frac{\sin\left(\frac{nz\sqrt{x^2+y^2+z^2}}{\sqrt{y^2+z^2}}\right)}{\sqrt{x^2+y^2+z^2}}$$

is a solution of the fourth order partial differential equation

$$\left(\frac{\partial^2}{\partial x^2}\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right) + n^2\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)f = 0.$$

The simplification procedures of Maple are powerful enough to solve this problem within a second. We shall use the procedure **radnormal** (radical normalization) to simplify the expression, which contains radicals.

```
> settime := time(): # start timing
> f := sin( n*z*sqrt(x^2+y^2+z^2)/sqrt(y^2+z^2) ) /
> sqrt(x^2+y^2+z^2);
```

$$f := \frac{\sin\left(\frac{nz\sqrt{x^2+y^2+z^2}}{\sqrt{y^2+z^2}}\right)}{\sqrt{x^2+y^2+z^2}}$$

```
> radnormal(diff(diff(f,x$2) + diff(f,y$2) + diff(f,z$2),
> x$2) + n^2*(diff(f,x$2) + diff(f,y$2)));
0
> cpu_time = (time()-settime)*second; # computing time
cpu_time = 0.671 second
```

In the second example, the objective is find the generating function for dimensions of representations of the Lie group of type G_2 (q.v., [57, 58]). So, the attempt is made to find a rational function $F(x, y)$ such that

$$F(x, y) = \sum_{k, l \geq 0} G2(k, l) x^k y^l,$$

where $G2(k, l)$ is the following polynomial expression.

```
> G2 := (k, l) -> 1/5!*(k+1)*(l+1)*(k+l+2)*(k+2*l+3)*
> (k+3*l+4)*(2*k+3*l+5);
```

$$G2 := (k, l) \rightarrow \frac{(k+1)(l+1)(k+l+2)(k+2l+3)(k+3l+4)(2k+3l+5)}{5!}$$

Here, we have used Maple's arrow notation for functional operators. In this way $G2$ is a function with values defined in terms of its arguments instead

of just a formula. Maple has a package called `genfunc` for manipulating rational generating functions. We use it to solve our problem.

```
> with(genfunc):      # load genfunc package
> settime := time(): # start timing
> F := rgf_encode(rgf_encode(G2(k,l), k, x), l, y):
> F := sort(factor(F));

F := (x^4 y^4 + 8 x^4 y^3 + x^3 y^4 + 8 x^4 y^2 - 26 x^3 y^3 + x^4 y
- 41 x^3 y^2 + 15 x^2 y^3 - 6 x^3 y + 78 x^2 y^2 - 6 x y^3 + 15 x^2 y - 41 x y^2
+ y^3 - 26 x y + 8 y^2 + x + 8 y + 1)/((x - 1)^6 (y - 1)^6)
> cpu_time = (time()-settime)*second; # computing time

cpu_time = .501 second
```

An example taken from scientific life where Maple could have played the role of mathematical assistant can be found in [240]. In this paper the Laplace-Beltrami operator Δ in hyperspherical coordinates is wanted. To this end, the Jacobian of the coordinate mapping, the metric tensor, and its inverse are calculated. Following are quotations from the paper:

“It is not difficult to compute $\frac{\partial \mathbf{Y}}{\partial q_i}$, and it is not difficult, but tedious, to compute the traces in Eq.(32B). After quite some algebra we find, ...”

“It is also tedious to invert \mathbf{g} . After several pages of computation of minors we find, ...”

These remarks are indeed true when one carries out these calculations with pencil and paper; but not if one lets Maple carry out the computations! Below is the Maple calculation, as much as possible in the notation of [240]. Don't worry if you do not fully understand individual commands: details will be provided later in this book.

The first step in the computation is to define the coordinate mapping Y and to build the metric tensor G . This turns out to be the most time-consuming step in the computation.

```
> settime := time(): # start timing
> with(LinearAlgebra): # load the linear algebra package
> R[z] := x -> <<cos(x)| -sin(x)| 0>,
> <sin(x)| cos(x)| 0>, <0| 0| 1>>:
> 'R[z](phi)' = R[z](phi);

R_z(phi) = 
$$\begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

> R[y] := x -> <<cos(x)| 0| -sin(x)>, <0| 1| 0>,
> <sin(x)| 0| cos(x)>>:
> 'R[y](phi)' = R[y](phi);
```

$$R_y(\phi) = \begin{bmatrix} \cos(\phi) & 0 & -\sin(\phi) \\ 0 & 1 & 0 \\ \sin(\phi) & 0 & \cos(\phi) \end{bmatrix}$$

```

> T := x -> <<cos(x)+sin(x)| 0| 0>,
> <0| cos(x)-sin(x)| 0>, <0| 0| 0>>:
> 'T(phi)' = T(phi);

T(phi) = \begin{bmatrix} \cos(\phi) + \sin(\phi) & 0 & 0 \\ 0 & \cos(\phi) - \sin(\phi) & 0 \\ 0 & 0 & 0 \end{bmatrix}

> macro(a=alpha, b=beta, c=gamma, f=phi, t=theta):
> Y := ScalarMultiply(R[z](a) . R[y](b) . R[z](c/2)
> . T(t/2) . R[z](f/2), r/sqrt(2))
> Y1 := map(diff, Y, r): Y2 := map(diff, Y, a):
> Y3 := map(diff, Y, b): Y4 := map(diff, Y, c):
> Y5 := map(diff, Y, t): Y6 := map(diff, Y, f):
> G := Matrix(6, 6, shape=symmetric):
> for i to 6 do for j from i to 6 do
>   G[i,j] := simplify(Trace(Transpose(Y||i) . Y||j))
> end do end do:
> intermediate_cpu_time = (time() - settime)*seconds;

```

intermediate_cpu_time = 2.394 seconds

Now, we apply some simplification procedures to obtain the formulae in [240]. To shorten the output of the session, we continue to suppress most results. We also show a slightly polished session, admittedly not the first interactive session when the problem was solved.

```

> G := subs(cos(t/2)^2=1/2+1/2*cos(t),
>   cos(c/2)^2=1/2+1/2*cos(c), sin(t/2)=sin(t)/(2*cos(t/2))
>   sin(c/2)=sin(c)/(2*cos(c/2)), G):
> G[2,2] := normal(subs(cos(c)*sin(t)=(cos(b)^2+sin(b)^2
>   *cos(c)*sin(t), normal(G[2,2]))):
> G[3,3] := normal(G[3,3]):
> G; # this the formula in the paper!

```

$$\begin{aligned}
& [1, 0, 0, 0, 0, 0] \\
& \left[0, \frac{1}{2} r^2 (1 - \sin(\theta) \cos(\gamma) \sin(\beta)^2 + \cos(\beta)^2), \right. \\
& \left. \frac{1}{2} r^2 \sin(\beta) \sin(\gamma) \sin(\theta), \frac{1}{2} r^2 \cos(\beta), 0, \frac{1}{2} r^2 \cos(\beta) \cos(\theta) \right] \\
& \left[0, \frac{1}{2} r^2 \sin(\beta) \sin(\gamma) \sin(\theta), \frac{1}{2} r^2 (\cos(\gamma) \sin(\theta) + 1), 0, 0, 0 \right] \\
& \left[0, \frac{1}{2} r^2 \cos(\beta), 0, \frac{r^2}{4}, 0, \frac{1}{4} r^2 \cos(\theta) \right] \\
& \left[0, 0, 0, 0, \frac{r^2}{4}, 0 \right]
\end{aligned}$$

$$\left[0, \frac{1}{2} r^2 \cos(\beta) \cos(\theta), 0, \frac{1}{4} r^2 \cos(\theta), 0, \frac{r^2}{4}\right]$$

Due to the screen width and the length of expressions Maple was not able to produce a nice layout, but it can translate the formulae automatically into a format suitable for text processing programs like L^AT_EX [159].

```
> latex(G, "metric_tensor"):
```

The L^AT_EX code is not shown, but the result after typesetting is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{r^2(\cos(\beta)^2+1-\sin(\theta)\cos(\gamma)\sin(\beta)^2)}{2} & \frac{r^2\sin(\gamma)\sin(\beta)\sin(\theta)}{2} & \frac{r^2\cos(\beta)}{2} & 0 & \frac{r^2\cos(\beta)\cos(\theta)}{2} \\ 0 & \frac{r^2\sin(\gamma)\sin(\beta)\sin(\theta)}{2} & \frac{r^2(1+\sin(\theta)\cos(\gamma))}{2} & 0 & 0 & 0 \\ 0 & \frac{r^2\cos(\beta)}{2} & 0 & \frac{r^2}{4} & 0 & \frac{r^2\cos(\theta)}{4} \\ 0 & 0 & 0 & 0 & \frac{r^2}{4} & 0 \\ 0 & \frac{r^2\cos(\beta)\cos(\theta)}{2} & 0 & \frac{r^2\cos(\theta)}{4} & 0 & \frac{r^2}{4} \end{bmatrix}$$

Let us compute the Jacobian.

```
> determinant := simplify(Determinant(G)):
> determinant := normal(subs(cos(b)^2=1-sin(b)^2,
>   determinant)):
> determinant := normal(subs(cos(t)^2=1--sin(t)^2,
>   cos(t)=sin(2*t)/(2*sin(t)), determinant)):
> Jacobian := sqrt(determinant) assuming 0<=r, 0<=t,
>   t<=Pi/2, 0<=b, b<=Pi;
```

$$Jacobian := \frac{1}{32} r^5 \sin(2\theta) \sin(\beta)$$

This is formula (33) in the paper. The assumptions on the hyperspherical coordinates were necessary to have the square root expression automatically simplified by Maple.

Next, we compute the inverse of the metric tensor.

```
> GINV := map(simplify, MatrixInverse(G)):
> GINV := subs(cos(t)^2=1-sin(t)^2,
>   cos(b)^2=1-sin(b)^2, GINV):
> cpu_time = (time()-settime)*seconds; # computing time
```

$$cpu_time = 3.425 \text{ seconds}$$

We do not show the inverse metric tensor, but all entries except GINV[4,4] are in the shape of formula (34) in the paper. Casting GINV[4,4] into good shape is not difficult; we skip this last part of the computation. Anyway, the calculation can easily be done in about 4 seconds of computing time and paper-ready formulae are obtained too!

Another example from science where computer algebra enables the researcher to finish off a proof of a mathematical result that requires a lot of straightforward but tedious computations can be found in [59]. There, a

purely algebraic problem, related to proving the existence of a certain finite subgroup of a Lie group, could be reduced to the problem of solving a set of linear equations in 240 variables with coefficients from the field of 1831 elements. This system of linear equations was easily shown to have a unique solution by computer. By pencil and paper this is almost impossible, but by computer this is quite feasible. The role of Maple in this computation is described in [167].

In the last two worked-out examples we have used Maple to determine a generating function and to compute a metric tensor and its inverse, respectively. Now, one may not think much of mathematical results obtained by a computer program, but remember that there are many independent ways to check the answers. One may even use Maple itself to check the answers or to enlarge one's confidence in the results: e.g., one can compute the first terms of the Taylor series and compare them with the original coefficients, and one can multiply the metric tensor and its inverse as an extra check on the answers.

Symbolic and algebraic computation often precedes numerical computation. Mathematical formulae are first manipulated to cast them into good shape for final numerical computation. For this reason, it is important that a computer algebra system provides a good interface between these two types of computation. Maple can generate C, FORTRAN, and Java expressions from Maple expressions. Double precision arithmetic and code optimization are optional. For example, the submatrix of the metric tensor G in the previous example consisting of the first two rows can be converted into FORTRAN as shown below. Two technicalities play a role: You have to get rid of the link between Euler's constant and the Greek character γ in Maple, e.g., by using the name `Gamma`, and you must use (named) arrays instead of nested lists for matrices.

```
> # recognition of Euler's constant
> CodeGeneration[Fortran](gamma);

cg = 0.5772156649D0

> H := SubMatrix(G, 2..3, 1..6):
> H := eval(%, gamma=Gamma); # replace gamma by Gamma

H :=

$$\begin{bmatrix} 0, \frac{1}{2} r^2 (1 - \sin(\theta) \cos(\Gamma) \sin(\beta)^2 + \cos(\beta)^2), \\ \frac{1}{2} r^2 \sin(\beta) \sin(\Gamma) \sin(\theta), \frac{1}{2} r^2 \cos(\beta), 0, \frac{1}{2} r^2 \cos(\beta) \cos(\theta) \\ 0, \frac{1}{2} r^2 \sin(\beta) \sin(\Gamma) \sin(\theta), \frac{1}{2} r^2 (\cos(\Gamma) \sin(\theta) + 1), 0, 0, 0 \end{bmatrix}$$

> CodeGeneration[Fortran](H, optimize=true,
>   functionprecision=double, coercetypes=false);
```



```

t1 = r ** 2
t2 = dsin(theta)
t3 = dcos(Gamma)
t4 = t2 * t3
t5 = dsin(beta)
t6 = t5 ** 2
t8 = dcos(beta)
t9 = t8 ** 2
t14 = dsin(Gamma)
t17 = t1 * t5 * t14 * t2 / 2
t18 = t1 * t8
t20 = dcos(theta)
cg(1,1) = 0
cg(1,2) = t1 * (1 - t4 * t6 + t9) / 2
cg(1,3) = t17
cg(1,4) = t18 / 2
cg(1,5) = 0
cg(1,6) = t18 * t20 / 2
cg(2,1) = 0
cg(2,2) = t17
cg(2,3) = t1 * (t4 + 1) / 2
cg(2,4) = 0
cg(2,5) = 0
cg(2,6) = 0

```

The answers obtained with a computer algebra system are either exact or in a user-defined precision. They can be more accurate than hand calculations [146]; and they can lead to many corrections to integral tables. Below, we give two examples of integrals incorrectly tabulated in one of the most popular tables, viz., Gradshteyn–Ryzhik [109]:

1. Formula 2.269

$$\int \frac{1}{x\sqrt{(bx+cx^2)^3}} dx = \frac{2}{3} \left(-\frac{1}{bx} + \frac{4c}{b^2} - \frac{8c^2x}{b^3} \right) \frac{1}{\sqrt{bx+cx^2}}.$$

2. Formula 3.828(19)

$$\int_0^\infty \frac{\sin^2(ax) \sin^2(bx) \sin(2cx)}{x} dx =$$

$$\frac{\pi}{16} (1 + \operatorname{sgn}(c-a+b) + \operatorname{sgn}(c-b+a) - 2\operatorname{sgn}(c-a) - 2\operatorname{sgn}(c-b)),$$

where $a, b, c > 0$.

As shown below Maple does not only give correct answers but also more information on conditions of the parameters. In the example we shall use the percentage symbol to refer to the previous result.

```
> Integrate(1/(x*sqrt((b*x+c*x^2)^3)), x);
```

$$\int \frac{1}{x\sqrt{(bx+cx^2)^3}} dx$$

> value(%);

$$\frac{2(-\sqrt{bx+cx^2}b^2 + 4\sqrt{bx+cx^2}bcx + 5\sqrt{bx+cx^2}c^2x^2 + 3c^2\sqrt{x(b+cx)x^2})\sqrt{x(b+cx)}}{(3xb^3\sqrt{x^3(b+cx)^3})}$$

> factor(%);

$$\frac{2(b+cx)(-b^2+4bcx+8c^2x^2)}{3b^3\sqrt{x^3(b+cx)^3}}$$

Assume that all variables are positive and simplify the above result under this condition.

> simplify(%) assuming positive;

$$\frac{2(-b^2+4bcx+8c^2x^2)}{3b^3x^{(3/2)}\sqrt{b+cx}}$$

It is a bit more work to get a look-alike of the tabulated result.

> subs(b+c*x=y/x, %);

$$\frac{2(-b^2+4bcx+8c^2x^2)}{3b^3x^{(3/2)}\sqrt{\frac{y}{x}}}$$

> simplify(%) assuming positive;

$$\frac{2(-b^2+4bcx+8c^2x^2)}{3b^3x\sqrt{y}}$$

> expand(%);

$$-\frac{2}{3bx\sqrt{y}} + \frac{8c}{3b^2\sqrt{y}} + \frac{16xc^2}{3b^3\sqrt{y}}$$

> collect(3/2*%, sqrt(y));

$$\frac{-\frac{1}{bx} + \frac{4c}{b^2} + \frac{8xc^2}{b^3}}{\sqrt{y}}$$

> subs(y=b*x+c*x^2, 2/3*%);

$$\frac{2(-\frac{1}{bx} + \frac{4c}{b^2} + \frac{8xc^2}{b^3})}{3\sqrt{bx+cx^2}}$$

The mistake in the tabulated result is an incorrect minus sign. In the fourth edition of Gradshteyn–Ryzhik [109] in 1980 this has been corrected. Anyway, the result in an integral table you can only believe or not; in Maple you can verify the result by differentiation.

> simplify(diff(%,x) - 1/(x*sqrt((b*x+c*x^2)^3)))
> assuming positive;

0

The second example is more intriguing: the tabulated result is wrong and Maple finds a more general answer.

```
> Integrate(sin(a*x)^2*sin(b*x)^2*sin(2*c*x)/x,
> x=0..infinity);
```

$$\int_0^\infty \frac{\sin(ax)^2 \sin(bx)^2 \sin(2cx)}{x} dx$$

```
> value(%);
```

$$\begin{aligned} & \frac{1}{8} \operatorname{csgn}(c) \pi + \frac{1}{16} \operatorname{csgn}(-2c + 2b) \pi - \frac{1}{16} \operatorname{csgn}(2c + 2b) \pi \\ & + \frac{1}{16} \operatorname{csgn}(-2c + 2a) \pi - \frac{1}{16} \operatorname{csgn}(2c + 2a) \pi \\ & + \frac{1}{32} \operatorname{csgn}(-2b + 2c + 2a) \pi + \frac{1}{32} \operatorname{csgn}(2b + 2c + 2a) \pi \\ & - \frac{1}{32} \operatorname{csgn}(-2b - 2c + 2a) \pi - \frac{1}{32} \operatorname{csgn}(2b - 2c + 2a) \pi \end{aligned}$$

Here, Maple gives the solution for general a , b , and c . You can specialize to the case of positive variables, which is the only case tabulated in [109].

```
> % assuming positive; # all variables > 0
```

$$\begin{aligned} & \frac{\pi}{32} + \frac{1}{16} \operatorname{signum}(-2c + 2b) \pi + \frac{1}{16} \operatorname{signum}(-2c + 2a) \pi \\ & + \frac{1}{32} \operatorname{signum}(-2b + 2c + 2a) \pi - \frac{1}{32} \operatorname{signum}(-2b - 2c + 2a) \pi \\ & - \frac{1}{32} \operatorname{signum}(2b - 2c + 2a) \pi \end{aligned}$$

Let us get rid of the 2's in the signs and factorize.

```
> factor(eval(%), signum=(signum@primpart)));
```

$$\begin{aligned} & \frac{1}{32} \pi (1 + 2 \operatorname{signum}(b - c) + 2 \operatorname{signum}(-c + a) + \operatorname{signum}(-b + c + a) \\ & - \operatorname{signum}(-b - c + a) - \operatorname{signum}(b - c + a)) \end{aligned}$$

So, a constant was wrong and a term was missing in the tabulated result.

In many cases of integration, conditions on parameters are important to obtain an exact result. Below is the example of the definite integral

$$\int_0^\infty \frac{t^{1/3} \ln(at)}{(b + 2t^2)^2} dx, \quad a, b > 0.$$

```
> assume(a>0, b>0):
> normal(integrate(t^(1/3)*ln(a*t)/(b+2*t^2)^2,
> t=0..infinity));
```

$$\frac{1}{36} \frac{\pi 2^{(1/3)} (2 \ln(a^{\sim}) \sqrt{3} + \pi - 3 \sqrt{3} - \sqrt{3} \ln(2) + \sqrt{3} \ln(b^{\sim}))}{b^{\sim(4/3)}}$$

The tildes after a and b in the above result indicate that these variables have certain assumed properties. Maple can inform its user about these properties.

```
> about(a);

Originally a, renamed a~:
is assumed to be: RealRange(Open(0),infinity)
```

Integration is also a good illustration of another advantage of computer algebra: it provides easy access to advanced mathematical techniques and algorithms. When input in Maple, the following two integrals

$$\int x^2 \exp(x^3) dx \quad \text{and} \quad \int x \exp(x^3) dx$$

give different kinds of response:

$$\frac{1}{3} \exp(x^3) \quad \text{and} \quad \int x \exp(x^3) dx.$$

This means more than just the system's incompetence to deal with the latter integral. Maple can decide, via the Risch algorithm [31, 98, 203], that for this integral no closed form in terms of elementary functions exists. This contrasts with the heuristic methods usually applied when one tries to find an answer in closed form. In the heuristic approach one is never sure whether indeed no closed form exists or that it is just one's mathematical incompetence. The Risch algorithm however is a complicated algorithm that is based on rather deep mathematical results and algorithms and that involves many steps that cannot be done so easily by pencil and paper. Computer algebra enables its user to apply such advanced mathematical results and methods without knowing all details.

Using a computer algebra system, one can concentrate on the analysis of a mathematical problem, leaving the computational details to the computer. Computer algebra systems also invite one to do "mathematical experiments". They make it easy to test mathematical conjectures and to propose new ones on the basis of calculations (q.v., [20, 165]). As an example of such a mathematical experiment, we shall conjecture a formula for the determinant of the $n \times n$ matrix A_n defined by

$$A_n(i, j) := x^{\gcd(i, j)}.$$

The first thing to do is to compute a few determinants, and look and see.

```
> numex := 7: # number of experiments
> dets := Vector(numex):
> macro(det=LinearAlgebra[Determinant]):
> for n to numex do
>   A[n] := Matrix(n, n, shape=symmetric):
```

```

>   for i to n do
>     for j to i do
>       A[n][i,j] := x^igcd(i,j)
>     end do
>   end do:
>   dets[n] := factor(det(A[n])):
>   print(dets[n])
> end do:

```

$$\begin{aligned}
& x \\
& x^2(x-1) \\
& x^3(x+1)(x-1)^2 \\
& x^5(x+1)^2(x-1)^3 \\
& x^6(x^2+1)(x+1)^3(x-1)^4 \\
& x^7(x^2+1)(x^3+x-1)(x+1)^4(x-1)^5 \\
& x^8(x^2+x+1)(x^2-x+1)(x^2+1)(x^3+x-1)(x+1)^5(x-1)^6
\end{aligned}$$

At first sight, it has not been a success. But, look at the quotients of successive determinants.

```

>   for i from 2 to numex do
>     quo(dets[i], dets[i-1], x)
>   end do;

```

$$\begin{aligned}
& x^2 - x \\
& x^3 - x \\
& x^4 - x^2 \\
& x^5 - x \\
& x^6 - x^3 - x^2 + x \\
& x^7 - x
\end{aligned}$$

In the n^{th} polynomial only powers of x appear that have divisors of n as exponents. Thinking of Möbius-like formulae and playing a little more with Maple, the following conjecture comes readily to mind.

Conjecture. $\det A_n = \prod_{j=1}^n \phi_j(x)$, where the polynomials $\phi_j(x)$ are defined by $x^n = \sum_{d|n} \phi_d(x)$.

Some Properties.

(i) If p is a prime number, then

$$\phi_p(x) = x^p - x,$$

and for any natural number r ,

$$\phi_{p^r}(x) = \phi_p(x^{p^{r-1}}).$$

(ii) If p is a prime number, not dividing n , then

$$\phi_{pn}(x) = \phi_n(x^p) - \phi_n(x).$$

(iii) Let $n = p_1^{r_1} \dots p_s^{r_s}$ be a natural number with its prime factorization, then

$$\phi_n(x) = \phi_{p_1 \dots p_s}(x^{p_1^{r_1-1} \dots p_s^{r_s-1}}).$$

(iv) We have

$$\phi_n(x) = \sum_{d|n} \mu\left(\frac{n}{d}\right) x^d = \sum_{d|n} \mu(d) x^{(\frac{n}{d})},$$

where μ is the Möbius function such that $\mu(1) = 1$, $\mu(p_1 \dots p_s) = (-1)^s$ if p_1, \dots, p_s are distinct primes, and $\mu(m) = 0$ if m is divisible by the square of some prime number.

For the interested reader: $\frac{1}{d}\phi_d(q)$ is equal to the number of monic irreducible polynomials of degree d in one variable and with coefficients in a finite field with q elements [176].

However much mathematical knowledge has been included in a computer algebra system, it is still important that an experienced user can enhance the system by writing procedures for personal interest. The author implemented in Maple several algorithms for inversion of polynomial mappings according to the methods developed in [76]. With these programs it is possible to determine whether a polynomial mapping has an inverse that is itself a polynomial mapping, and if so, to compute the inverse. We show an example of an invertible mapping in three unknowns.

```
> read "invpol.m"; # load user-defined package
> P := [ x^4 + 2*(y+z)*x^3 + (y+z)^2*x^2 + (y+1)*x
>       + y^2 + y*z, x^3 + (y+z)*x^2 + y, x + y + z ];
```

$$P := [x^4 + 2(y+z)x^3 + (y+z)^2x^2 + (y+1)x + y^2 + yz, \\ x^3 + (y+z)x^2 + y, x + y + z]$$

```
> settime := time(): # start timing
> invpol(P, [x,y,z]); # compute inverse mapping
```

$$[x - yz, y - x^2z + 2yz^2x - y^2z^3, z - x - y + yz + x^2z - 2yz^2x + y^2z^3]$$

```
> cpu_time = (time()-settime)*second; # computing time
```

$$cpu_time = .110 \text{ second}$$

Computing the inverse of a polynomial mapping with pencil and paper is almost impossible; one needs a computer algebra system for the symbol

crunching. However, one cannot expect designers of computer algebra systems to anticipate all of the needs of their users. One can expect good programming facilities to implement new algorithms oneself.

1.5 Limitations of Computer Algebra

What has been said about computer algebra systems so far may have given the impression that these systems offer unlimited opportunities, and that they are a universal remedy for solving mathematical problems. But this impression is too rosy. A few warnings beforehand are not out of place.

Computer algebra systems often make great demands on computers because of their tendency to use up much memory space and computing time. The price one pays for exact arithmetic is often the exponential increase in size of expressions and the appearance of huge numbers. This may even happen in cases where the final answer is simply “yes” or “no.” For example, it is well-known that Euclid’s algorithm yields the greatest common divisor (gcd) of two polynomials. However, this “naive” algorithm does not perform very well. Look at the polynomials

```
> f[1] := 7*x^7 + 2*x^6 - 3*x^5 - 3*x^3 + x + 5;
```

$$f_1 := 7x^7 + 2x^6 - 3x^5 - 3x^3 + x + 5$$

```
> f[2] := 9*x^5 - 3*x^4 - 4*x^2 + 7*x + 7;
```

$$f_2 := 9x^5 - 3x^4 - 4x^2 + 7x + 7$$

We want to compute the $\gcd(f_1, f_2)$ over the rational numbers by Euclid’s algorithm. In the first division step we construct polynomials q_2 and f_3 , such that $f_1 = q_2 f_2 + f_3$, where $\deg(f_3) < \deg(f_2)$ or $f_3 = 0$. This is done by long division.

```
> f[3] := sort(rem(f[1], f[2], x, q[2]));
```

$$f_3 := \frac{70}{27}x^4 - \frac{176}{27}x^3 - \frac{770}{81}x^2 - \frac{94}{81}x + \frac{503}{81}$$

```
> q[2];
```

$$\frac{7}{9}x^2 + \frac{13}{27}x - \frac{14}{81}$$

Next, polynomial q_3 and f_4 are computed such that $f_2 = q_3 f_3 + f_4$, where $\deg(f_4) < \deg(f_3)$ or $f_4 = 0$, etc. until $f_n = 0$; then $\gcd(f_1, f_2) = f_{n-1}$. The following Maple program computes the polynomial remainder sequence.

```

> Euclid_gcd := proc(f::polynom, g::polynom, x::name)
>   local r;
>   if g = 0 then sort(f)
>   else
>     r := sort(rem(f, g, x));
>     if r <> 0 then print(r) end if;
>     Euclid_gcd(g, r, x)
>   end if
> end proc;
> Euclid_gcd(f[1], f[2], x):

```

$$\begin{aligned}
& \frac{70}{27}x^4 - \frac{176}{27}x^3 - \frac{770}{81}x^2 - \frac{94}{81}x + \frac{503}{81} \\
& \frac{100881}{1225}x^3 + 72x^2 - \frac{14139}{2450}x - \frac{98037}{2450} \\
& - \frac{16726864175}{10176976161}x^2 - \frac{5255280625}{10176976161}x + \frac{19754564375}{10176976161} \\
& \frac{35171085032244648729}{456796710414528050}x + \frac{6605604895087335357}{456796710414528050} \\
& \frac{240681431042721245661011901925}{121549387831506345564025862481}
\end{aligned}$$

The conclusion is that the polynomials f_1 and f_2 are relatively prime, because their greatest common divisor is a unit. But look at the tremendous growth in the size of the coefficients from 1-digit integers to rational numbers with thirty digits (even though the rational coefficients are always simplified). In [98, 148] one can read about more sophisticated algorithms for gcd computations that avoid blowup of coefficients as much as possible.

The phenomenon of tremendous growth of expressions in intermediate calculations turns up frequently in computer algebra calculations and is known as *intermediate expression swell*. Although it is usually difficult to estimate the computer time and memory space required for a computation, one should always do one's very best to optimize calculations, both by using good mathematical models and by efficient programming. It is worth the effort. For example, with the **LinearAlgebra** package, Maple computes the inverse of the 9×9 matrix with (i, j) -entry equal to $(iu + x + y + z)^j$ in 535 seconds requiring memory allocation of 6 Mbytes on a PC running Windows 2000, with a 1.7 Ghz Pentium 4 processor having 512 MB main memory. The obvious substitution $x + y + z \rightarrow v$ reduces the computer time to 1 second and the memory requirements to about 0.5 Mbyte.

A second problem in using a computer algebra system is psychological: how many lines of computer output can one grasp? And when one is faced with large expressions, how can one get enough insight to simplify them? For example, merely watching the computer screen it would be difficult to

recover the polynomial composition

$$f(x, y) = g(u(x, y), v(x, y)),$$

where

$$\begin{aligned} g(u, v) &= u^3v + uv^2 + uv + 5, \\ u(x, y) &= x^3y + xy^2 + x^2 + y + 1, \\ v(x, y) &= y^3 + x^2y + x, \end{aligned}$$

from its expanded form.

While it is true that one can do numerical computations with a computer algebra system in any precision one likes, there is also a negative side of this feature. Because one uses software floating-point arithmetic instead of hardware arithmetic, numerical computation with a computer algebra system is 100 to 1000 times slower than numerical computation in a programming language like FORTRAN. Hence, for numerical problems, one must always ask oneself “Is exact arithmetic with rational numbers or high-precision floating-point arithmetic really needed, or is a numerical programming language preferable?”

In an enthusiastic mood we characterized computer algebra systems as mathematical expert systems. How impressive the amount of built-in mathematical knowledge may be, it is only a small fraction of mathematics known today. There are many mathematical areas where computer algebra is not of much help yet, and where more research is required: partial differential equations, indefinite and definite integration involving non-elementary functions like Bessel functions, contour integration, surface integration, calculus of special functions, and non-commutative algebra are just a few examples.

Another serious problem and perhaps the trickiest, is the wish to specify at an abstract level the number domain in which one wants to calculate. For example, there are an infinite number of fields, and one may want to write algorithms in a computer algebra system using the arithmetic operations of a field, without the need to say what particular field you work with. Moreover, one may want to define one’s own mathematical structures. AXIOM [70, 71, 140, 219] was the first computer algebra system making steps in this direction. In Maple, the **Domains** package allows its user to create domains in a similar way as AXIOM does.

As far as the syntax and the semantics are concerned, the use of a computer algebra system as a programming language is more complicated than programming in a numerical language like FORTRAN. In a computer algebra system one is faced with many built-in functions that may lead to unexpected results. One must have an idea of how the system works, how data are represented, how to keep data of manageable size, and so on.

Efficiency of algorithms, both with respect to computing time and memory space, requires a thorough analysis of mathematical problems and a careful implementation of algorithms. For example, if we had forgotten to expand intermediate results in the procedure for calculating Bessel polynomials as defined in §1.3, it would have resulted in unnecessarily large expressions. The rather unsuccessful implementation of **Y** would compute $Y_5(z)$ as follows.

```
> Y := proc(n::nonnegint, x::name)
>   if n=0 then 1
>   elif n=1 then x+1
>   else Y(n,x) := (2*n-1)*x*Y(n-1,x) + Y(n-2,x)
>   end if
> end proc:
> Y(5,z);
```

```
9 z (7 z (5 z (3 z (z + 1) + 1) + z + 1) + 3 z (z + 1) + 1) + 5 z (3 z (z + 1) + 1)
+ z + 1
```

Familiarity with the basic features of a computer algebra system, such as elementary data structures and built-in facilities, makes it easier to program efficiently and to foresee some of the many pitfalls in symbolical computation. A good understanding of the computer algebra systems that one has at one's disposal is also prerequisite to making the right choice for the system to use when studying a particular problem. For example, the formula manipulation system FORM [225] is better suited than Maple for doing computations in most non-commutative algebras because, in this system, non-commutative objects are of basic type and pattern matching is an amply supported basic operation. See [60] for a comparison of Maple and FORM in computing with quaternions.

Finally, we mention the technical difficulties in using computer algebra systems. Sometimes, input and output formats and the reuse of expressions are confusing and too far removed from standard mathematical notation. In comparison to global manipulations, local manipulations in formulae are often difficult to carry out. Interfaces with other programming languages are frequently absent or inadequate. And last but not least, bugs are a major issue. See [215] for an entertaining discussion on this issue. It is hoped that some of these remarks will soon be outdated. But systems like Maple will always surprise you, for the good or the bad. We end this section with two surprising results in Maple calculations where you have to assist the system a bit.

The first example: an easy integral that can be done without using a computer algebra system.

```
> Integrate(2*x*(x^2+1)^24, x);
```

$$\int 2x(x^2 + 1)^{24} dx$$

The answer is clear, but what does Maple give?

```
> value(%);
```

$$92x^6 + 12x^4 + x^2 + \frac{653752}{5}x^{30} + 178296x^{28} + 81719x^{18} + 178296x^{22} \\ + \frac{653752}{5}x^{20} + 208012x^{24} + 208012x^{26} + 506x^{42} + \frac{10626}{5}x^{40} \\ + \frac{10626}{5}x^{10} + 506x^8 + 7084x^{12} + 19228x^{14} + 43263x^{16} + \frac{1}{25}x^{50} \\ + x^{48} + 12x^{46} + 92x^{44} + 7084x^{38} + 19228x^{36} + 43263x^{34} + 81719x^{32}$$

Surprise! You may think that factorization helps.

```
> factor(%);
```

$$x^2(x^8 + 5x^6 + 10x^4 + 10x^2 + 5)(x^{40} + 20x^{38} + 190x^{36} + 1140x^{34} \\ + 4845x^{32} + 15505x^{30} + 38775x^{28} + 77625x^{26} + 126425x^{24} \\ + 169325x^{22} + 187760x^{20} + 172975x^{18} + 132450x^{16} + 84075x^{14} \\ + 43975x^{12} + 18760x^{10} + 6425x^8 + 1725x^6 + 350x^4 + 50x^2 + 5)/25$$

The problem is, as you may have guessed, in the choice of the integration constant.

```
> factor(% + 1/25);
```

$$\frac{(x^2 + 1)^{25}}{25}$$

You are expecting too much if you think that such problems can always be avoided.

The second example comes from a study in optics [19] and is about integrating expressions consisting only of terms

$$x^n \sin^i x \cos^j x \cosh^k x \sinh^l x,$$

where $i, j, k, l, n \in \mathbb{N}$. It can easily be shown that any such integral can be expressed completely in the same kind of terms. An example of what Maple does:

```
> Integrate(x*sin(x)^2*cos(x)*sinh(x)^2*cosh(x), x);
```

$$\int x \sin(x)^2 \cos(x) \sinh(x)^2 \cosh(x) dx$$

```
> value(%);
```

$$\frac{1}{32} \left(\frac{3x}{10} - \frac{2}{25} \right) e^{(3x)} \cos(x) - \frac{1}{32} \left(-\frac{x}{10} + \frac{3}{50} \right) e^{(3x)} \sin(x) \\ - \frac{1}{192} x e^{(3x)} \cos(3x) + \frac{1}{32} \left(-\frac{x}{6} + \frac{1}{18} \right) e^{(3x)} \sin(3x) - \frac{1}{64} x e^x \cos(x) \\ + \frac{1}{32} \left(-\frac{x}{2} + \frac{1}{2} \right) e^x \sin(x) + \frac{1}{32} \left(\frac{x}{10} + \frac{2}{25} \right) e^x \cos(3x)$$

$$\begin{aligned}
& -\frac{1}{32} \left(-\frac{3x}{10} + \frac{3}{50}\right) e^x \sin(3x) + \frac{1}{64} x e^{(-x)} \cos(x) \\
& + \frac{1}{32} \left(-\frac{x}{2} - \frac{1}{2}\right) e^{(-x)} \sin(x) + \frac{1}{32} \left(-\frac{x}{10} + \frac{2}{25}\right) e^{(-x)} \cos(3x) \\
& - \frac{1}{32} \left(-\frac{3x}{10} - \frac{3}{50}\right) e^{(-x)} \sin(3x) + \frac{1}{32} \left(-\frac{3x}{10} - \frac{2}{25}\right) e^{(-3x)} \cos(x) \\
& - \frac{1}{32} \left(-\frac{x}{10} - \frac{3}{50}\right) e^{(-3x)} \sin(x) + \frac{1}{192} x e^{(-3x)} \cos(3x) \\
& + \frac{1}{32} \left(-\frac{x}{6} - \frac{1}{18}\right) e^{(-3x)} \sin(3x)
\end{aligned}$$

To get the formula in the requested form you have to write the exponentials in terms of hyperbolic sines and cosines and work out the intermediate expression.

```
> convert(%,'trig'); # exp -> trigonometric function
> expand(%);
```

$$\begin{aligned}
& \frac{1}{5} \cos(x) x \sinh(x) \cosh(x)^2 - \frac{1}{6} x \sinh(x) \cosh(x)^2 \cos(x)^3 \\
& - \frac{1}{6} x \cosh(x)^3 \sin(x) \cos(x)^2 + \frac{1}{5} x \cosh(x) \sin(x) \cos(x)^2 \\
& + \frac{1}{18} \sinh(x) \cosh(x)^2 \sin(x) \cos(x)^2 - \frac{1}{10} \cos(x) x \sinh(x) \\
& - \frac{13}{450} \sin(x) \sinh(x) \cosh(x)^2 - \frac{1}{50} \cos(x) \cosh(x)^3 \\
& - \frac{1}{10} \sin(x) x \cosh(x) + \frac{1}{15} \sin(x) x \cosh(x)^3 + \frac{1}{15} x \sinh(x) \cos(x)^3 \\
& - \frac{13}{450} \sinh(x) \sin(x) \cos(x)^2 + \frac{19}{450} \sin(x) \sinh(x) + \frac{1}{50} \cosh(x) \cos(x)^3
\end{aligned}$$

The final step might be to combine the commands into a procedure for further usage.

```
> trivalent := proc()
>   integrate(args);
>   convert(%,'trig');
>   expand(%);
> end proc;
> trivalent(x*sin(x)^2*cos(x)^3, x);
```

$$\begin{aligned}
& \frac{1}{15} x \sin(x) \cos(x)^2 + \frac{2}{15} x \sin(x) + \frac{1}{45} \cos(x)^3 + \frac{2}{15} \cos(x) \\
& - \frac{1}{5} x \cos(x)^4 \sin(x) - \frac{1}{25} \cos(x)^5
\end{aligned}$$

```
> trivalent(x*sin(x)*cos(x)^2, x=0..Pi);
```

$$\frac{\pi}{3}$$

This kind of adapting Maple to one's needs happens fairly often. Therefore, this book contains many examples of how to assist the system. Some of them are of a rather sophisticated level. They have been added because merely explaining Maple commands and showing elementary examples does not make you proficient enough at the system for when it comes to real work.

1.6 Design of Maple

The name Maple is not an acronym of **m**athematical **p**leasure — great fun as it is to use a computer algebra system — but was chosen to draw attention to its Canadian origin. Since 1980 a lot of work has gone into the development of the Maple system by the Symbolic Computation Group of the University of Waterloo and at ETH Zürich. Since 1992 it has been further developed and marketed by Waterloo Maple Software (since 1995, Waterloo Maple Inc.) in collaboration with the original developers.

Maple is a computer algebra system open to the public and fit to run on a variety of computers, from a mainframe computer down to desktop computers like Macintosh and PC. The user accessibility shows up best on a mainframe with a time-sharing operating system, where several users of Maple can work simultaneously without being a nuisance to each other or other software users, and without excessive load put on the computer. This is possible because of the modular design of Maple. It consists of several parts: the user interface called the *Iris*, the basic algebraic engine or *kernel*, the external *library*, and optionally the so-called *share library* with contributions of Maple users or other private libraries.

The *Iris* and the kernel form the smaller part of the system, which has been written in the programming language C; they are loaded when a Maple session is started. The *Iris* handles input of mathematical expressions (parsing and notification of errors), display of expressions (“prettyprinting”), plotting of functions, and support of other user communication with the system. There are special user interfaces called *worksheets* for the X Window System, Macintosh, and MS-Windows. *Maplets*, which are launched from a Maple session, provide customized graphical user interfaces, e.g., for setting kernel options or for building plots in an interactive way.

In a Maple worksheet you can combine Maple input and output, graphics, and text in one document. Further features of the worksheet interface are that it provides hypertext facilities inside and between documents, that it allows embedding of multi-media objects (on some platforms), that it uses typeset mathematics, and that subexpressions of mathematical output can be selected for further processing. A Maple worksheet consists of a hierarchy

of regions. Regions can be grouped into sections and subsections. Sections and subsections can be “closed” to hide regions inside. A worksheet can be exported as RTF (Rich Text format), HTML (with MathML), XML, or \LaTeX format. You are referred to the Maple documentation for precise details about the worksheet interface.

The Maple kernel interprets the user input and carries out the basic algebraic operations such as rational arithmetic and elementary polynomial arithmetic. It also contains certain algebraic routines that are so often used that they must be present in the lower-level systems language, for efficiency reasons. To the latter category belong routines for manipulation of polynomials like **degree**, **coeff**, and **expand**. The kernel also deals with storage management. A very important feature of Maple is that the system keeps only one copy of each expression or subexpression within an entire session. In this way testing for equality of expressions is an extremely inexpensive operation, viz., one machine instruction. Subexpressions are reused instead of being recomputed over and over again.

Most of the mathematical knowledge of Maple has been coded in the Maple programming language and resides as functions in the external library. When a user needs a library function, Maple is smart enough to load the routine itself. Maple must be informed about the use of separate packages like the linear algebra, number theory, and statistics packages, to name a few. All this makes Maple a compact, easy-to-use system. But what is more important, in this setting Maple uses memory space only for essential things and not for facilities in which a user is not interested. This is why many people can use Maple on one computer simultaneously, and why Maple runs on computers with rather limited memory. Table 1.1 summarizes the design of the Maple system as just described.

Part	Function
Iris	parser display of expressions (“prettyprinting”) graphics special user interfaces
Kernel	interpreter memory management basic & time-critical procedures for computations in \mathbb{Z} , \mathbb{Q} , \mathbb{R} , \mathbb{C} , \mathbb{Z}_n , $\mathbb{Q}[x]$, etc.
Library	library functions application packages on-line help
Private Libraries	contributions of Maple users

Table 1.1. Components of the Maple system.

The Maple language is a well-structured, comprehensible, high-level programming language. It supports a large collection of data structures: functions, sequences, sets, lists, arrays, tables, etc. There are also plenty of easy-to-use operations on these data structures like type-testing, selection and composition of data structures, and so on. These are the ingredients of the programming language in which almost all mathematical algorithms of Maple are implemented, and which is the same language users will employ in interactive calculator mode. Furthermore, anyone who is interested in the algorithms that Maple uses and who is interested in the way these algorithms are implemented can look at the Maple code in the library; the procedures are available in readable form or can be reproduced as such inside a Maple session. If desired, a user can enlarge the library with self-written programs and packages. The *Maple Application Center* (URL: www.mapleapps.com) contains many user contributions, sample worksheets, and documentation. Maple also provides its users with facilities to keep track of the execution of programs, whether self-made or not.

The last advantage of Maple we shall mention is its user-friendly design. Many computer algebra systems require a long apprenticeship or knowledge of a low-level systems language if one really wants to understand what is going on in computations. In many systems one has to leaf through the manual to find the right settings of programming flags and keywords. Nothing of the kind in Maple. First, there is the help facility, which is basically the on-line manual for Maple procedures. Secondly, the secret to why Maple is so easy to use is the hybrid algorithmic structure of the computer algebra system, where the system itself can decide which algorithm is favorable. As an example we look at some invocations of the Maple procedure **simplify**, which does what its name suggests.

```
> trig_formula := cos(x)^6 + sin(x)^6
> + 3*sin(x)^2*cos(x)^2:
> exp_ln_formula := exp(a+1/2*ln(b)):
> radical_formula := (x-2)^(3/2)/(x^2-4*x+4)^(1/4):
> trig_formula = simplify(trig_formula);
```

$$\cos(x)^6 + \sin(x)^6 + 3 \sin(x)^2 \cos(x)^2 = 1$$

```
> exp_ln_formula = simplify(exp_ln_formula);
```

$$e^{(a+1/2 \ln(b))} = e^a \sqrt{b}$$

```
> radical_formula = simplify(radical_formula);
```

$$\frac{(x-2)^{(3/2)}}{(x^2-4x+4)^{(1/4)}} = \frac{(x-2)^{(3/2)}}{((x-2)^2)^{(1/4)}}$$

Here, Maple does not assume that $x \geq 2$ and cannot simplify the square root. The generic simplification of the square root can be forced by adding the keyword **symbolic**.

```
> radical_formula = simplify(radical_formula, symbolic);
```

$$\frac{(x-2)^{(3/2)}}{(x^2-4x+4)^{(1/4)}} = x-2$$

But the key point is that just one procedure, viz., **simplify**, carries out distinct types of simplifications: trigonometric simplification, simplifications of logarithms and exponential functions, and simplification of powers with rational exponentials. On the other hand, the concept of pattern matching and transformation rules (rewrite rules) is underdeveloped in Maple. It is rather difficult to program mathematical transformations that can be applied globally.

At many places Maple makes decisions which way to go; we mention four examples. Computation of the determinant of a matrix is done by the method of minor expansion for a small matrix, otherwise Gaussian elimination is used. Maple has essentially six numerical methods at its disposal to compute definite integrals over a finite interval: In the default case (no particular method specified), the integration problem is first passed to NAG integration routines if **Digits** is not too large. If the NAG routines cannot perform the integration, then some singularity handling may be performed and control may pass back to the NAG routines with a modified problem. The default hybrid numeric-symbolic integration method is Clenshaw-Curtis quadrature, but when convergence is slow (due to nearby singularities) the system tries to remove the singularities or switches to an adaptive double-exponential quadrature method. An adaptive Newton-Cotes method is available when low precision (e.g., **Digits** ≤ 15) suffices. Other numerical integration methods are the adaptive Gaussian quadrature method and the sinc quadrature method. By the way, generalized series expansions and variable transformations are two of the techniques used in the Maple procedure **evalf/int** to deal with singularities in an analytic integrand. The interested reader is referred to [93, 97]. At present, there are six algorithms coded into the Maple procedure **fsolve**: Newton, Secant, Dichotomic, inverse parabolic interpolation, a method based on approximating the Jacobian (for systems), and a method of partial substitutions (for systems again). As a user of Maple you normally do not need to know about or act on these details; the system finds its own way. This approach turns Maple into an easy-to-learn and easy-to-use system for mathematical computations on computers.



<http://www.springer.com/978-0-387-00230-9>

Introduction to Maple

HECK, A.

2003, XVI, 828 p., Hardcover

ISBN: 978-0-387-00230-9