A GUIDE TO MAPLE

Springer Science+Business Media, LLC

ERNIC KAMERICH

A GUIDE TO MAPLE

With 41 Illustrations

Springer

Ernic Kamerich
Katholieke Universiteit Nijmegen
Toernooiveld 1
Nijmegen, 6525 ED
The Netherlands
ernic@sci.kun.nl

Library of Congress Cataloging-in-Publication Data
Kamerich, Ernic.
A guide to Maple / Ernic Kamerich.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-4612-6436-1 ISBN 978-1-4419-8556-9 (eBook)
DOI 10.1007/978-1-4419-8556-9
1. Maple (Computer file). 2. Mathematics—Data processing.
I. Title.
QA76.95.K355 1998
510'.285'53—dc21 98-30559

Printed on acid-free paper.

Maple is a registered trademark of Waterloo Maple, Inc.

Production managed by Anthony K. Guardiola; manufacturing supervised by Nancy Wu.
Camera-ready copy prepared using the author's TeX files.

9 8 7 6 5 4 3 2 1

ISBN 978-1-4612-6436-1 SPIN 10122799

Short Reference List

This list contains only the most basic Maple commands. All commands explained in this book can be found in the *Catchword Index.*

`Elements of commands that must be typed in literally are printed in a typewriter font, as in the present line, or in a bold font.`
Elements of commands that stand for Maple expressions to be chosen by the user are printed in an italic font, as in the present line.

1 special characters

`;`	terminator for commands
`:`	the same, but without printing results to the screen
`:=`	used in assignments
`=`	used in equations, not in assignments
`%1`	abbreviation (in printed results)
`%, %%, %%%`	refer to last, second last, and third last result in Release V.5 (in command lines)
`", "", """`	refer to last, second last, and third last result in releases before V.5
`" "`	a pair of double quotes makes the sequence of characters to a string in Release V.5
`` ` ` ``	a pair of single back quotes makes the sequence of characters to a symbol in Release V.5 and to a string in earlier releases
`' '`	a pair of single forward quotes prevents evaluation
`.`	used for concatenation of strings or in floating point numbers
`..`	range
`*`	obligatory multiplication sign
`!`	faculty
`&`	prefix for special operators such as `&*`
~	postfix, indicating a property of a variable (in printed results)

`->`	in function definitions, where the expression after `->` is taken literally
`@`	composition of functions
`@@`	repeated composition of a function
`( )`	parentheses as usual in mathematics, also for arguments to functions
`[ ]`	list
`{ }`	set
`,`	separator in sequences, lists, sets, and between arguments of functions
`_`	underscore, used for internal names
`$`	repeat
`::`	for type testing of arguments to procedures
`?`	on-line help

2 general

constants	e is entered as `exp(1)`, π as `Pi`, i as `I`
`eval(`*expr*`,1)`	evaluates *expr* only one step
`assume(`*expr*`,`*prop*`)`	assumes the property *prop* for the —otherwise undefined—variable or expression *expr*
expr `mod` *n*	calculates *expr* modulo *n*
`sqrt(`*expr*`)`	square root of *expr*
`with`	loads package
`readlib`	reads a procedure from the standard library

3 substitution, subexpressions

`subs(`*old*`=`*new*`,`*expr*`)`	substitutes *new* for *old* in *expr*. If *expr* is a name referring to a procedure or matrix, etc., apply `subs(`*old*`=`*new*`,eval(`*expr*`))` For simultaneous substitution, use a list of equations.
`eval(`*expr*`,` *old*`=`*new*`)`	the same, but without replacing formal parameters, such as x in diff(f(x),x)
`simplify(`*expr*`,` *equation*`)`	simplify *expr* to side relation *equation*, possibly with optional third argument *[<vars>]*, indicating to which variables the expression should be reduced
`op(`*expr*`)`	yields the operands of *expr*; cannot be applied to a sequence
`op(`*n*`,`*expr*`)`	yields the *n*th operand of *expr*; cannot be applied to a sequence
select from sequence	by indexing: if `sols` is a sequence, then `sols[2]` yields the second element of the sequence `sols`
`subsop(`*n*`=`*expr1*`,`*expr2*`)`	substitutes *expr1* for the *n*th operand of *expr2*
`map(`*proced*`,`*expr*`)`	applies the procedure *proced* to all the operands of *expr* (possibly a list or set), composing the results according to the structure of *expr*
`map2(`*expr1*`,`*proced*`,`*expr2*`)`	applies the procedure *proced* to *expr1* as the first argument and the operands of *expr2* as second argument, composing the results according to the structure of *expr2*
`coeff(`*poly*`,`*x*`,`*n*`)`	gives the coefficient of x^n in the polynomial *poly*
`numer`	yields the numerator of a quotient
`denom`	yields the denominator of a quotient
`lhs,rhs`	yield the left- and right-hand side of an equation

4 manipulating numbers and formulas

`evalc(`*expr*`)`	tries to convert a complex algebraic expression *expr* into something of the form $a + b\,I$, where a and b are real expressions, supposing that all names in *expr* refer to real numbers
`convert(`*expr*`,polar)`	converts a complex expression *expr* into a polar form of type $\mathrm{polar}(a, \phi)$, to be interpreted as $ae^{\phi i}$
`collect(`*poly*`,`*x*`)`	describe *poly* as a polynomial in *x*
`expand(`*expr*`)`	expands powers, multiplications, and function calls in *expr*
`factor(`*expr*`)`	factors *expr* over the rational numbers
`normal(`*expr*`)`	converts an algebraic expression *expr* into one quotient and applies some elementary simplification rules
`combine(`*expr*`,`*option*`)`	acts contrary to expansion of functions, if the type of combination is given as `option`. Possible options: `exp`, `ln`, `power`, `trig`, `Psi`, `radical`,`abs`, `signum`, `plus`, `atatsign`, `conjugate`, `plot`, `product`, `range`, `polylog`, `cmbplus`, `cmbtms`, `cmbpwr`, `polylog`
`simplify(`*expr*`,`*option*`)`	simplifies *expr*, according to the rules of the given *option*. Possible options: `power`, `commonpow`, `radical`, `RootOf`, `polar`, `infinity`, `max`, `atsign`, `atatsign` collections of functions: `trig`, `arctrig`, or the name of a function such as `exp`, `ln`, `Dirac`, `hypergeom`, etc., for composition of functions: `atsign`, for repeated compositions: `atatsign`
`convert(`*expr*`,`*option*`)`	action depends on option; many options are available; consult the on-line help
`convert(`*expr*`,RootOf)`	converts radicals in *expr* into `RootOf` expressions

`convert(`*expr*`,radical)`	converts each `RootOf` expression in *expr* into a radical by choosing one of the values that is found with `allvalues`, if possible
`convert(`*expr*`,rational)`	converts all the floating-point numbers occurring in *expr*, into rational numbers; optional third option `exact`
`testeq(`*expr1*`=`*expr2*`)`	executes a numerical test on the given equality. Not to be used with radicals and `RootOf` expressions.

5 graphics and numerical calculations

graphics	for two-dimensional plots use `plot`, for three-dimensional plots use `plot3d`. Additional procedures in the packages `plots` and `DEtools`.
`plot(`*expr*`,`*x*`=`*a*` .. `*b*`)`	plots a graph of an algebraic expression *expr* as a function in *x*, where *x* ranges from *a* to *b*. Many options and variations possible.
`plot3d(`*expr*`,`*x*`=`*a*` .. `*b*`,`*y*`=`*c*` .. `*d*`)`	the same for three-dimensional plots
`plots[display]({ })`	combine several plots into one picture
`evalf(`*expr*`,`*n*`)`	approximate all the real numbers in *expr* by floating-point numbers. Calculations are executed to *n* digits.
`Digits`	variable that determines accuracy of floating-point calculations when not specified explicitly

6 solving equations

`solve(`*equa*`,`*x*`)`	tries to solve *equa* for the variable *x* exactly
`solve({`*equa1*`,`*equa2*`},{`*x*`,`*y*`})`	tries to solve the system {equa1,equa2} for the variables *x* and *y* exactly (also possible with larger systems)

`fsolve(`*equa*`,`*x*`)`	tries to find an approximation to one of the solutions of *equa* for *x* (in case of a polynomial: to all solutions), also to be used for systems of equations
`fsolve(`*equa*`,`*x*`,`*x*`=`*a*`..`*b*`)`	tries to find one approximation to a solution of *equa* for *x* within the segment [*a*,*b*]
`lhs,rhs`	yield the left and right-hand side of an equation
`allvalues(`*expr*`)`	yields all possible values if the `RootOf` expressions contained in *expr* are solved as polynomial equations; it replaces equal `RootOf` expressions on several places by the same value if not an option `independent` is added
`dsolve`	solves differential equations: see calculus

7 calculus

`diff(`*f(x)*`,`*x*`)`	differentiates the expression *f(x)* in respect to the variable *x* with result: an expression
`D(`*f*`)`	differentiates the function *f* with result: a function
`diff(`*f(x,y)*`,`*x*`,`*y*`,`*y*`)`	calculates $\frac{\partial^3}{\partial x\,\partial y^2}\,\mathrm{f}(x,y)$
D[1,2,2](*f*)	calculates the function $(x,y) \mapsto \frac{\partial^3}{\partial x\,\partial y^2}\,\mathrm{f}(x,y)$
`int(`*expr*`,`*x*`)`	$\int expr\,dx$
`int(`*expr*`,`*x*`=`*a*`..`*b*`)`	$\int_a^b expr\,dx$
`sum(`*expr*`,`*k*`=`*a*`..`*b*`)`	$\sum_a^b expr$
`product(`*expr*`,`*k*`=`*a*`..`*b*`)`	$\prod_a^b expr$
infinity	is denoted as `infinity` in Maple
numerical integration	`evalf(Int(`*expr*`,`*x*`=`*a*`..`*b*`))` calculates a numerical approximation to $\int_a^b expr\,dx$ without trying to calculate the integral symbolically

`series(`*expr*`,`*x*`=`*c*`,`*n*`)`	calculates a series expansion of *expr*, perceived as a function in *x*, around *x*=*c*. The order can be determined with the aid of the last argument, *n* in this case, but may be lower than this number. The result is of type series, usually.
`convert(`*sr*`,polynom)`	converts the expression *sr*, supposed to be of type `series`, into a (generalized) polynomial, omitting the order term
`Order`	variable that determines the order of series calculations, when the order is not specified otherwise
`limit(`*expr*`,`*x*`=`*a*`)`	tries to calculate $\lim_{x=a} expr$; *a* may be `infinity` or `-infinity`; an option may be added: `left`, `right`, `real`, or `complex`
`dsolve`	tries to solve a differential equation or set of differential equations
differential equation	is denoted in terms like `diff(f(x),x,x)=-f(x)` or `D(D(f))(x)=-f(x)`
initial conditions	are given like `f(0)=1`, `D(f)(0)=a`, etc.
`->`	(arrow: combination of "minus" and "greater than"), used for construction of functions, for instance `x->x^2`, `(x,y)->x*cos(y)` The expression at the right-hand side of the arrow is taken literally. For functions of more than one variable, parentheses around the parameters are obligatory.
`unapply(`*expr*`,`*vars*`)`	constructs a function in *vars* described by the result of evaluating *expr*
`?inifcns`	yields a listing of all mathematical functions known to Maple at start-up
`piecewise`	yields piecewise-defined expression

8 linear algebra

start linear algebra	by reading the `linalg` package: `with(linalg):`
create	matrix or column vector, for example `matrix( [[1,2,3][4,5,6]] )` `vector( [1,2,3] )`
diagonal matrix	for example `diag(a,b,c)`
evaluate	matrix A (or vector) with `eval(A)` or, if it contains assigned names, `map(eval,A)`
matrix arithmetic	requires always `evalm`, for instance `evalm(5*A^2+B&*C)` Use `&*` for multiplying a matrix with a matrix or vector, use `*` for scalar multiplications.
transpose	vector or matrix with `transpose`
`dotprod(`v`,`w`)`	dot product of v and w
`copy(`A`)`	creates a new matrix or vector object in memory with the same entries as A
`det(`A`)`	determinant of A
`eigenvals, eigenvects`	calculate eigenvalues and eigenvectors of a square matrix
`colspace, rowspace`	calculate a basis of the linear space spanned by columns, or rows, of a matrix
`kernel`	calculates the kernel of a matrix
`concat, stack`	glue two matrices together side by side, or bottom to top

How to use this book

If you have little or no experience with Maple, you are advised to read the first five chapters (70 pages) carefully, possibly skipping sections indicated with a star before the number. It is advantageous if you can read with Maple running on a computer close at hand; try out examples and experiment a little, but don't waste time if you get stuck on something. Probably you will learn about that problem by reading on.

If you already have some experience with Maple, reading the first five chapters may help you to use Maple more efficiently and to understand the basic ideas.

If you have read the first five chapters or already know the basics of Maple well, you can proceed with the chapters related to your specific interests. Some of these chapters are directed at a mathematical field, others at a symbolic manipulation field. Each chapter has a preface, where you can see what you can expect in that chapter. At the end of the book you can find appendices on some rather special subjects.

When working with Maple and encountering problems, you can consult this book with the aid of the contents and the *index on catchwords*. Each reference to a catchword corresponds to a bold printed word or a section heading in the text. It is no problem if you have not read preceding parts of the book; the many cross references make it easy to find additional information when necessary. Moreover, there is an *index on error messages*, demonstrated and explained in examples in this book.

If you are not a daily user, you might forget Maple commands. The most used commands are summarized in the *Short Reference List* at the start of the book. There is room to write your own extension of that list.

It is a good habit not to switch off your mind when you switch on the computer, even when you are to use such a powerful tool as Maple. In many examples in this book you can see how common mathematical sense can help considerably in using Maple.

An essential aspect of using computer systems for calculation is the question of reliability of calculated results. This aspect is discussed throughout this book where relevant, and ways of checking and/or testing are shown. However, even where a calculation is said to be reliable, nobody can be sure that unknown bugs will not appear. Testing and checking results is a good habit in general, and more so when using such a complicated system.

The present book is based on Maple V Release 5, but differences from releases 3 and 4 are indicated where relevant. Maple output is printed in the style of output generated by Maple versions for windowing systems, such as for MS-Windows, Mac, X-terminals and -workstations, up to the choice of fonts and line breaking.

More on Maple

You can obtain general information on Maple, additional software (new packages in the Maple Share Library), demos, and Maple support information from the Maple Info Server on http://www.maplesoft.com/home.html. On this same site you can find titles of books on using Maple.

Here you can also download patches for bugs; as is the case with most software, new releases tend to have bugs, these are attended to by the development team; bug fixes become available from this www-site as patch files.

A vivid discussion on all aspects of Maple is going on continuously in the Maple User Group on electronic mail, where people ask simple as well as advanced questions or bring Maple aspects to debate, and others reply. You can subscribe to this group by sending an email message to `majordomo@daisy.uwaterloo.ca`, with the message body containing the line: `subscribe maple-list [<address>]`.

Acknowledgments

First I would like to thank the development team of Maple and the many contributors of mathematical algorithms and Maple source outside that team for creating such a powerful and enjoyable companion in mathematical calculations.

I would like to thank the Mathematical Institute of Catholic University Nijmegen, which offered me the opportunity for writing this book and the necessary support, and the computer department for their kind and patient help.

For the writing of this book, I owe many thanks to the people whom I could introduce to Maple; by their asking questions and showing me their difficulties they have helped me considerably in teaching the use of Maple. Another source of problems and ideas has been the discussions in the Maple User Group, whose contributors I would like to thank as well. I would also like to thank Drs. A. Heck of Computer Algebra Nederland for the productive arguments we often had on Maple and on teaching its use, and I would like to thank Professor Dr. A.H.M. Levelt of Nijmegen University for his incentive to write this book.

I would like to thank Drs. J.M.G. Ingelaat, Professor Dr. A.G.M. Janner at Nijmegen University, R.M. Corless, D. Redfern, B. Barber, and all the other people, who read the book or parts of it and commented it, for their helpful comments. Also I would like to thank

This book has been written in TEX. I would like to thank Dr. V. Eijkhout for introducing me to TEX and helping me with some problems, and Yunliang Yu, who created a powerful Maple package for converting Maple expressions into TEX; I have adapted this package for simulating screen output of Maple commands in this book.

At last I would like to thank my friends for encouraging me to write this book.

Ernic Kamerich

In spite of all efforts and help, undoubtedly there is room for improvements. So, if you have suggestions, corrections, or other remarks concerning this book, I will very much welcome your comments at the following address:

Dr. B.N.P. Kamerich
Fac. of Math. and Comp. Sci.
Catholic University Nijmegen
Toernooiveld 1
6525 ED Nijmegen
The Netherlands

fax: 0031-243652140

electronic mail: `ernic@sci.kun.nl`

Contents

chapter 1

Basic elements in the use of Maple

The first three sections of this chapter introduce you to Maple with some basic examples in solving equations, both numerically and symbolically. At the same time, you see some general aspects of using Maple, such as giving commands and assigning values to variables. These are discussed systematically in the subsequent sections, where some common users' mistakes are also demonstrated. Moreover, these sections show some more basic examples of calculations and the on-line help system of Maple.

1.1 Meeting Maple: symbolic calculations

When you start Maple on your computer, you can see the cursor waiting for your input at the right of the input prompt, usually " > ".
If you see a cursor with a question mark, Maple is in the state of "math input", where commands can be entered with the aid of the mouse and the palettes of symbols. You can change to the usual input from the keyboard by entering Control-J or by choosing `execution group` from the `Insert` menu of the toolbar at the top of the window.
Suppose that we are interested in a function in (positive) x, described by the formula

$$\frac{16x^2 - 24x + 121}{32x^2 - 48x + 34}$$

This formula can be entered in Maple at the right of the prompt by typing

```
(16*x^2 - 24*x + 121) / (32*x^2 - 48*x + 34)
```

Please note that the spaces are optional, but can be useful for readability.

Now we type a semicolon (;) and press the Enter key. (In the state of math input the semicolon is omitted.) Then the screen looks like

```
> (16*x^2 - 24*x + 121) / (32*x^2 - 48*x + 34);
```

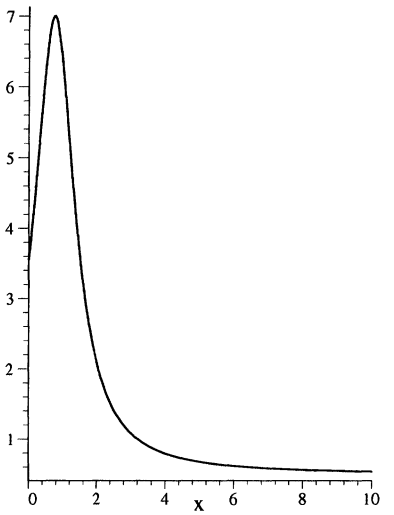
$$\frac{16\,x^2 - 24\,x + 121}{32\,x^2 - 48\,x + 34}$$

Although the formula has to be entered in a simple linear way, Maple presents it in the usual, much more readable fashion. If Maple has been started in a tty (characters only) system (MS-DOS or Unix for instance), formulas are still presented in the same style, but there are some restrictions in the presentation of special symbols. All screen parts presented in this book are printed in the style of Maple on windowing systems (such as Ms Windows), but line breaking may be different and the left brackets in the Maple command screen are omitted.

E. Kamerich, *A Guide to Maple*

First, let's make a graph. In the next command the percentage sign % is used for referring to the previous formula.

```
> plot( % , x = 0 .. 10 );
```



Maple has read the last command as if we had typed

```
plot((16*x^2-24*x+121)/(32*x^2-48*x+34),x = 0 .. 10);
```

In Maple the **percentage sign %** is called the *ditto*. In releases before Maple V.5, the **double quote "** is used for the ditto.
It is a good idea to assign the formula to a name, say `peak`. For this purpose, we have to ditto the formula again. That is possible with the *twofold* ditto, as the formula is the *second last* result.

```
> peak := %% ;
```

$$peak := \frac{16x^2 - 24x + 121}{32x^2 - 48x + 34}$$

From now on, Maple reads `peak` as that formula.

Let's suppose that we intend to compare the previous expression with another expression in x,

```
> (476*x+891)/(484*x+53);
```

$$\frac{476x + 891}{484x + 53}$$

and that we want to know the area of the region where the first expression exceeds the second one.

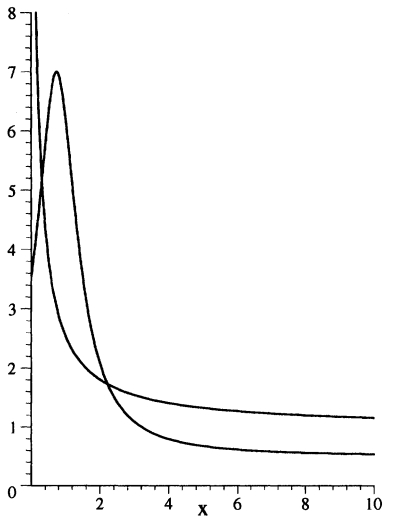
Let's assign this second formula to a name as well:

```
> fall := % ;
```

$$fall := \frac{476\,x + 891}{484\,x + 53}$$

In order to compare both, we can plot them together first. Because of the vertical asymptote of `fall`, we must restrict the vertical range of the plot, say to the range $0 \ldots 8$.

```
> plot( { peak , fall } , x=0..10 , 0..8 );
```



Braces are used as they are used in mathematics: for denoting a set. The last argument restricts the range of the second coordinate to $[0, 8]$.

Our intention is to compute the area between the curves where the value of *peak* is larger than the value of *fall*. First, let's find the x for which both are equal. If you are using a windowing version of Maple, you can get approximations to the coordinates of the intersection points by pointing with the mouse and clicking, but you can also ask Maple to solve the following equation:

```
> peak = fall;
```

$$\frac{16\,x^2 - 24\,x + 121}{32\,x^2 - 48\,x + 34} = \frac{476\,x + 891}{484\,x + 53}$$

```
> solve( % , x );
```

$$\frac{11}{36}, -\frac{5}{4} + 2\sqrt{3}, -\frac{5}{4} - 2\sqrt{3}$$

As could be guessed from the equation, we obtain three solutions. Generally, solving third-degree equations yields much longer results. This is no problem for Maple, but this example has been chosen in order to find a more readable result.

The present solutions are expressed in an exact way, using rational numbers and

square roots. This is typical for a symbolic calculator such as Maple. Approximations are available as well by applying the procedure `evalf` ("evaluate to a floating-point number").

```
> evalf( [%] );
```

$$[.3055555556,\ 2.214101616,\ -4.714101616]$$

Square brackets are used for denoting a list. The solutions are bundled here into a list because `evalf` can approximate only one object, and this object may be an expression or a list of expressions, etc., but not a sequence of expressions; a more detailed explanation is given in section 10.3 on page 127.

In order to compute the area of the region where the first function exceeds the second one, the difference should be integrated from the first to the second solution. A solution can be selected from the sequence of solutions by indexing, for instance:

```
> %%[1];
```

$$\frac{11}{36}$$

So the desired integral can be calculated by

```
> int( peak-fall , x = %%%[1] .. %%%[2] );
```

$$\frac{91}{121} - \frac{117}{121}\sqrt{3} + \frac{13}{4}\sqrt{2}\arctan\left(\left(-2+2\sqrt{3}\right)\sqrt{2}\right) + \frac{25376}{14641}\ln(2) - \frac{25376}{14641}\ln\left(-69+121\sqrt{3}\right) + \frac{13}{4}\sqrt{2}\arctan\left(\frac{4}{9}\sqrt{2}\right) + \frac{25376}{14641}\ln(113) - \frac{50752}{14641}\ln(3)$$

Again, Maple has calculated this in an exact way. Let's approximate this to 40 digits:

```
> evalf( % , 40 );
```

$$3.822742863575994928443933759352133657199$$

You may have wondered if we could not tell Maple beforehand that we want only positive solutions of the equation $peak = fall$. However, if a symbolic solution is a very complicated expression, it may take some clever, complicated, and time-consuming manipulations in order to see in a *symbolic* way if this expression is really a positive number or not; see Chapter 12, *Manipulating and converting numbers*, and Chapter 15, *Manipulating algebraic expressions*. Automatic manipulations without the help of a user may even fail to prove that a number is positive. Therefore, the decision is left to the user. Usually, a floating-point approximation, possibly with a high level of `Digits`, is an effective tool for such a decision, although symbolic manipulation may be necessary if a strict mathematical proof is required.

1.2 Meeting Maple: numerical calculations

In the previous section, exact results have been found and these have been approximated. It is also possible, and often necessary, to calculate numeric results directly. For instance, in the previous example, Maple had to solve a polynomial equation of degree 3. But there is no method for solving all polynomial equations, although Maple always tries to solve a high-degree polynomial equation by special tricks. If this is not successful, you may want to revert to numerical calculations. Here you see how the previous problem can be handled by the numerical equation solver of Maple, the procedure `fsolve` ("floating-point solve").

```
> fsolve( peak=fall , x );
```

$$.3055555556$$

We achieve just one solution. The graph indicates that there are two more solutions, one between 0 and 1 and the other between 2 and 3. We can ask for them by specifying these ranges.

```
> fsolve( peak=fall , x , x=0..1 );
```

$$.3055555556$$

However, this equation can be handled in a much more elegant way by converting it to a polynomial equation as follows. First we convert *peak* − *fall* into one quotient with **normal**:

```
> normal(peak-fall);
```

$$-\frac{13}{2}\,\frac{576\,x^3+1264\,x^2-6452\,x+1837}{(16\,x^2-24\,x+17)\,(484\,x+53)}$$

Now let's solve for which x the numerator is zero.

```
> numer(%);
```

$$-7488\,x^3-16432\,x^2+83876\,x-23881$$

```
> fsolve( %=0 , x );
```

$$-4.714101615,\ .3055555556,\ 2.214101615$$

Maple uses a specific numerical method for solving polynomial equations, yielding all real solutions at once. All solutions of such a polynomial equation, including complex ones, can be found by the command `fsolve( ,x,complex)`.

It is easy to obtain solutions with a higher accuracy by changing the value of **Digits**. If we want solutions to 50 digits, we have to set the variable `Digits` to 50.

```
> Digits := 50:
```

Maple did not print a result on the screen for the last command because the command ended with a colon (:) instead of a semicolon (;).

```
> fsolve( %%% , x );
```

$$-4.7141016151377545870548926830117447338856105076208,$$
$$.30555555555555555555555555555555555555555555555556,$$
$$2.2141016151377545870548926830117447338856105076208$$

Let's reset `Digits` immediately to its usual value.

```
> Digits := 10:
```

Maple has many powerful numerical algorithms. For instance, we can calculate the earlier integral in a numerical way as well. A very convenient aspect of using Maple is the combination of both symbolic and numerical calculations (and plotting) in the same environment. However, if you want to take symbolic results over to specialized numerical routines in Fortran or C, Maple offers translators for this purpose; see section 9.5 on page 123.

1.3 Meeting Maple: symbolic calculations again

Often, numerical methods are not really satisfactory, especially when using unknown parameters. Here is an equation containing a parameter p:

```
> 18*p^2*x^4 + 36*p^3*x^2 + 36*p*x^3 + 72*p^2*x
  + 63*x^2 + 27 = 27*p^2*x^2 + 54*x^4 +
  108*p*x^2 + 54*p*x + 36*p;
```

$$18p^2x^4 + 36p^3x^2 + 36px^3 + 72p^2x + 63x^2 + 27 =$$
$$27p^2x^2 + 54x^4 + 108px^2 + 54px + 36p$$

As you can see, it is easy to enter large formulas: Maple continues on the next line(s) until all is entered, then you can terminate it with a semicolon. You can force a new line by pressing the Enter key; in windowing systems you then get a warning:

```
Warning, incomplete statement or missing semicolon
```

but that is only a reminder.

Let's assign this equation to a name. In Maple a name can refer to a number, an expression, an equation, a procedure and many other things.

```
> eq := % ;
```

$$eq := 18p^2x^4 + 36p^3x^2 + 36px^3 + 72p^2x + 63x^2 +$$
$$27 = 27p^2x^2 + 54x^4 + 108px^2 + 54px + 36p$$

Let's solve for x. A numerical solver would be useless because p is contained in the equation, so we use **solve**.

```
> solve( % , x );
```

$$\frac{1}{2}\sqrt{-8p+6},\ -\frac{1}{2}\sqrt{-8p+6},\ \frac{1}{2}\frac{-2p+2\sqrt{2p^2-3}}{p^2-3},$$
$$\frac{1}{2}\frac{-2p-2\sqrt{2p^2-3}}{p^2-3}$$

It is important to realize that the equation is solved only for a "general" p. Let's have a closer look at this aspect by following the steps in this solving process. Maple has solved a fourth-degree polynomial equation. Let's have a look at this polynomial. First take the difference of the left-hand side and the right-hand side of the equation with `lhs` and `rhs`:

```
> poly := lhs(eq)-rhs(eq);
```

$$poly := 18p^2x^4 + 36p^3x^2 + 36px^3 + 72p^2x + 63x^2 + 27 - 27p^2x^2 - 54x^4 - 108px^2 - 54px - 36p$$

The result is assigned to the name `poly`. Let's write this as a polynomial in x.

```
> collect( % , x );
```

$$\left(-54+18p^2\right)x^4+36px^3+\left(63-108p-27p^2+36p^3\right)x^2+\left(72p^2-54p\right)x+27-36p$$

For special values of p (viz. $\sqrt{3}$ and $-\sqrt{3}$), the coefficient of x^4 is zero. Consequently, the general solution given by Maple cannot be applied to the cases where $p=\sqrt{3}$ or $p=-\sqrt{3}$.

> Names are interpreted as general abstract objects in Maple.
> Substituting special values for names in the result of a command
> does not always yield a correct result
> for the corresponding special mathematical problem.

This special problem can be solved simply by substituting these special values and solving the corresponding special problem separately. So let's substitute $p=\sqrt{3}$ in the original equation and solve the resulting equation.

```
> subs( p=sqrt(3) , eq );
```

$$54x^4+108\sqrt{3}x^2+36\sqrt{3}x^3+216x+63x^2+27 = 81x^2+54x^4+108\sqrt{3}x^2+54\sqrt{3}x+36\sqrt{3}$$

```
> solve( % , x );
```

$$\frac{1}{2}\sqrt{6-8\sqrt{3}},\ -\frac{1}{2}\sqrt{6-8\sqrt{3}},\ \frac{1}{6}\sqrt{3}$$

The third element of this solution is not contained in the solution for general p. The same can be done for $p = -\sqrt{3}$.

> Special cases for special values of parameters
> have to be looked for by the user, possibly with Maple as a tool.

More on this subject can be found in R.M. Corless and D.J. Jeffrey: Well ... It Isn't Quite That Simple, Sigsam Bulletin 26 (1992).
You have seen enough examples for now. It is time to look more systematically at some details.

1.4 Spaces and asterisks

Generally, spaces are neglected by Maple.

```
> a*b + p   *     q;
```

$$a\,b + p\,q$$

However, spaces can improve readability; thus the frequent use of spaces in Maple commands printed in this book.

In printing output, Maple represents multiplication by inserting spaces between the factors, which might be tempting you to use spaces to denote multiplication in the input. This, however, is not accepted by Maple; Maple reports a syntax error, indicating the first offending character by the cursor, in this case the character b:

```
> a b;
Syntax error, missing operator or ';'
```

Because Maple can use names of more than one character, the product of `a` and `b` cannot be entered as `ab` either. Maple requires the **asterisk** (*) for each multiplication.

1.5 Terminating commands with semicolons or colons

Each Maple command must be terminated with a **semicolon** (;) or with a **colon** (:). (This is not true if you have changed the Input Display form Maple Notation to Standard Math in the Options menu: then no terminator is expected.) Pressing the Enter key makes the present line enter the Maple system, but if no terminator is detected at the end of the command, the line is supposed to be incomplete and it must be continued on the next line. In windowing versions of Maple the user gets a warning message `incomplete statement or missing semicolon`.
It can easily happen that a user forgets the obligatory terminator:

```
> arcsin(1/2)
>
Warning, premature end of input
```

After that warning, you might be tempted to enter the line again, adding a semicolon. But then Maple gets really cross and issues a syntax error; from the first command you can see the following on the screen:

```
> arcsin(1/2)
> arcsin(1/2);
Syntax error, missing operator or ';'
```

In a windowing version you can see a bracket in the left indicating that both lines are taken together as one command. The transition to the second line is interpreted as a space: `arcsin(1/2) arcsin(1/2);` . (These brackets are omitted in this book, but here such a bracket can help to see what has happened.)
Maple puts the cursor on the second line, which seems to be a perfect line, but entering this again by pressing the Enter key does not work, because both lines are still read together. The remedy is simple:

> If you have forgotten the terminating semicolon,
> type this semicolon on the next line and press the Enter key again,
> or move the cursor back to the previous line,
> add the semicolon, and press Enter.

For instance, in the previous example:

```
> arcsin(1/2)
> ;
```

$$\frac{1}{6}\pi$$

If you are using a Maple version for text-only command screens such as MS-DOS Maple and Maple for Unix systems, you must be careful about missing (semi)colons, and it might be worthwhile to look at Appendix C, *The user interface for text-only versions.*

It is possible to give more than one command in one line. Here for instance the roots of the derivative of an expression are calculated:

```
> x^3 + 11*x^2 - 16*x; diff(%,x); solve( % , x );
```

$$x^3 + 11\,x^2 - 16\,x$$

$$3\,x^2 + 22\,x - 16$$

$$-8, \frac{2}{3}$$

The results from the commands on that one line are printed on subsequent lines.

Sometimes it is more efficient if Maple does not print the result of a command to the screen. For instance, after the command to elaborate $(x + \frac{y}{3})^{12}$

```
> expand( (x + y/3)^12 );
```

$$x^{12}+4\,y\,x^{11}+\frac{22}{3}\,y^2\,x^{10}+\frac{220}{27}\,y^3\,x^9+\frac{55}{9}\,y^4\,x^8+\frac{88}{27}\,y^5\,x^7+$$
$$\frac{308}{243}\,y^6\,x^6+\frac{88}{243}\,y^7\,x^5+\frac{55}{729}\,y^8\,x^4+\frac{220}{19683}\,y^9\,x^3+$$
$$\frac{22}{19683}\,y^{10}\,x^2+\frac{4}{59049}\,y^{11}\,x+\frac{1}{531441}\,y^{12}$$

you might wish to preserve the result by assigning it to a name. It is useless to see this formula again, so we terminate the command with a **colon**; this prevents Maple from printing the result on the screen.

```
> pow := % :
```

Maple has executed the command and has stored the resulting expression for calling with the ditto. However, you do not see the result because of the colon at the end of the command.

1.6 Names and assignments

The combination of colon and equal sign (**:=**) is used to **assign** a Maple object to a name:

```
> y := a*x^2-1;
```

$$y := a\,x^2 - 1$$

From now on, in almost any instance where Maple encounters y, it reads this as $a\,x^2 - 1$. A mathematician could say: "y is a formula in the free variables x and a." In terms of Maple: "y, x, and a are **names**, where y is a name that **refers to** an expression, while x and a do not refer to anything."

Be careful about capitals in names: Maple distinguishes between lower case and upper case:

```
> Y - y;
```

$$Y - a\,x^2 + 1$$

In mathematics, variables usually have one-letter names, but in Maple, words can be used. In section 3.5 on page 37 you can read more about combinations of characters that can or cannot be used as a name.

A typical mistake is the use of `=` or `:` instead of `:=` in an assignment. Here is an example where both mistakes are shown.

```
> x1 = 10;
```

$$x1 = 10$$

```
> x2 : 20;
```

$$20$$

```
> x1 + x2;
```

$$x1 + x2$$

The first line yields an equation. The second line consists of two Maple commands: the first asks Maple to evaluate x2, but Maple does not print the result to the screen as the command is terminated with a colon. The second command on this line asks Maple to yield 20. So both x1 and x2 stay unassigned and the last line yields simply x1 + x2.

1.7 Referring to previous results with the ditto

As you have seen, it is possible to refer to the previous result with the **ditto** (%). In order to refer to results one or two steps earlier, you can use a two- or threefold ditto.

Here is an example. First the antiderivative (indefinite integral) of an expression is calculated.

```
> 1/x/(a*x^2+b*x)^(3/2);
```

$$\frac{1}{x\,(a\,x^2+b\,x)^{(3/2)}}$$

```
> integrate( % , x );
```

$$-\frac{2}{3}\,\frac{1}{b\,x\,\sqrt{a\,x^2+b\,x}}+\frac{8}{3}\,\frac{a\,(2\,a\,x+b)}{b^3\,\sqrt{a\,x^2+b\,x}}$$

```
> normal( % );
```

$$\frac{2}{3}\,\frac{-b^2+8\,a^2\,x^2+4\,a\,x\,b}{b^3\,x\,\sqrt{a\,x^2+b\,x}}$$

Now the result of `integrate` can be checked easily by comparing its derivative with the original expression by using the two- and threefold ditto.

```
> normal( diff(%%,x) - %%% );
```

$$0$$

In the last line the twofold ditto calls the second last result, the result of `integrate( % , x );` and the threefold ditto calls the third last result, the original expression.

One might be tempted to enter something like:

```
> newvar;
```

$$newvar$$

```
> % := 100;
```

$$\% := 100$$

If the second command was meant to assign 100 to `newvar`, it was not successful. Maple seems to be willing, but the output of the last command already indicates that something else happened: only the ditto is made to refer to 100, not `newvar`:

```
> newvar;
```

$$newvar$$

If you are using the ditto, remember that the result of a command terminated by a colon is put on the ditto stack as well, although the result is not printed on the screen. For example,

```
> 7^20:
> % / 7^18;
```

$$49$$

Some commands do not yield any result as far as the ditto is concerned. For instance, the procedures **print** and **lprint**.

```
> lprint(x^2/3);
1/3*x^2
> %;
```

$$49$$

The ditto does not refer to the last command line, but to the last result. Remember that more than one command can be given on one line, often generating more than one result.

If you are using a windowing version, it is possible to execute any previous command again by moving the cursor to that line and pressing the Enter key. You can also change the command by editing before issuing. However, this command will use the present values of variables, and dittos will refer to the present history of results. If this command yields a result, this is the last result at that moment. A ditto in the command next issued will refer to that very result, which may not be the result of the command *above* it in your worksheet.

It is not possible to refer to earlier results than the third last result with the ditto facility.

If you are used to the double quote as the ditto in earlier releases, you might do things such as:

```
> a+Pi;
```

$$a + \pi$$

```
> (cos(")+1)^2;
Warning, incomplete string; use " to end the string
(note that the ditto operator is now % instead of ")
```

The best thing you can do is edit the input line, replacing " with %. When you follow the advice and enter a double quote, Maple is still not content:

```
> ";
; unexpected
```

This is caused by the parentheses in the start of the input: `(cos(` . In a windowing version of Maple it will be necessary to enter these extra parentheses; correct the input line into:

```
> "));
Error, cos expects its 1st argument, x, to be of type algebraic,
but received ")+1)^2;\n"
```

After this error, Maple is ready for new input.
In a text-only version, don't bother about it: Maple reports a syntax error and is ready for new input.

1.8 Referring to previous results with other facilities

In windowing versions you can select a previous result by pointing to that result with the mouse and then pressing Control with the left mouse button or by triple clicking with the left button. Then press Control-C to copy it to the clipboard, put the cursor on the correct input line, and press Control-V to paste it there. Don't forget to add a terminator. You can edit the expression before pressing the Enter key. If the **Output Display** is in Editable Math Notation (see the Options menu), the default at start-up, then you can select syntactically valid portions of an output expression by pressing the left mouse button, keeping it pressed, and dragging the mouse over that part.

Some versions of release V.5 (MS Windows and Mac) offer a spreadsheet facility. If you are experimenting with the input for a calculation over several command lines, this can be a very comfortable tool: for the calculation in one cell you can refer to the content of other cells, and changes in one cell will cause recalculation in all the other cells referring to this cell.

An old, rather primitive facility is **showtime**. After the command `showtime()`, results are stored by assigning them successively to the names `O1`, `O2`, etc. (the character O, followed by a number). Moreover, for each command, processor time and workspace are shown.

1.9 Using procedures

Let's have a close look at what happens when a procedure is applied. Here is an example:

```
> form := (10*x^2-15*x)/(2*x^3+2*x-3*x^2-3) + 1;
```

$$form := \frac{10\,x^2 - 15\,x}{2\,x^3 + 2\,x - 3\,x^2 - 3} + 1$$

```
> normal(form);
```

$$\frac{x^2 + 5\,x + 1}{x^2 + 1}$$

What has happened due to the second command is

- the argument `form` has been evaluated to the formula to which it refers
- the procedure `normal` has converted this formula into one ratio and it has divided out the common factor $2x - 3$ from numerator and denominator
- the result has been printed to the screen
- and it has been 'stacked up' by Maple so that a **ditto** in the next command can recall this result.

By applying `normal` to `form` we have *not* changed the reference of `form`:

```
> form;
```

$$\frac{10\,x^2 - 15\,x}{2\,x^3 + 2\,x - 3\,x^2 - 3} + 1$$

The procedure `normal` could not have changed the value of `form` because `form` was evaluated (its value was looked up) before `normal` came into action.

This example illustrates the usual action of procedures: the arguments are evaluated first, then the procedure uses the results of these evaluations to calculate the next result.

1.10 Procedures that seem to do nothing

Sometimes a procedure seems to do nothing:

```
> normel(form);
```

$$\mathrm{normel}\left(\frac{10\,x^2 - 15\,x}{2\,x^3 + 2\,x - 3\,x^2 - 3} + 1\right)$$

Here Maple finds an unknown name of a procedure. For our eyes it may seem to be a mistyping of `normal`. Maple does not protest, but prints the unevaluated procedure call on the screen. The idea behind it is the possibility that a procedure `normel` might be defined later.

In the next command, the procedure `sin` is applied to 1; this procedure tries to simplify the expression $\sin(1)$, but no better expression is found, so the function call itself is returned.

```
> sin(1);
```

$$\sin(1)$$

This is called an **unevaluated function call.** Generally, if asked for, Maple can supply an approximation in such a case.

```
> evalf( % , 30 );
```

$$.841470984807896506652502321630$$

Here we have asked for an approximation of $\sin(1)$ to 30 decimals.

Sometimes, applying a procedure can yield an error message, or an unexpected result, possibly just silence. Then you can ask Maple for some elucidation about its activities by using `infolevel`. For instance, if we ask for the exact solutions of the equation $\sin(x^2) = x^3$ in x, we cannot expect Maple to find them:

```
> sin(x^2)=x^3;
```

$$\sin(x^2) = x^3$$

```
> solve( % , x );
>
```

It is clear that there are solutions: 0 is a solution, but no result or message is printed. We can ask for more information: by setting the `infolevel` for `solve`:

```
> infolevel[solve] := 3:
> solve( %% , x );
solve/rec2:   solving for linear equation in   _S03
solve/rec2:   solving for linear equation in   _S01
solve/rec/RootOf:   RootOfs substitution   _S04 = RootOf(-_S03^3+_Z^2)
solve/rec2:   solving for linear equation in   _S04
solve:   Warning: no solutions found
solve:   Warning: solutions may have been lost
```

Now we can see that Maple gives it a try with the aid of many tricks, but has to give up in the end and then warns that solutions may have been lost.
The value of the `infolevel` of a procedure can be set to 0, 1, 2, 3, 4, or 5. If the value is set to 0, then no additional information is given. Generally, the value 1 gives sufficient information.
It is a good idea to reset things immediately:

```
> infolevel[solve] := 0:
```

1.11 The sign % for abbreviations in output

For complicated results, Maple can make results more readable by using abbreviations. For instance,

```
> x^3 - a*x - 1;
```

$$x^3 - a\,x - 1$$

```
> solve( % , x );
```

$$\frac{1}{6}\ \%1^{(1/3)} + 2\,\frac{a}{\%1^{(1/3)}},$$

$$-\frac{1}{12}\ \%1^{(1/3)} - \frac{a}{\%1^{(1/3)}} + \frac{1}{2}\,I\sqrt{3}\left(\frac{1}{6}\ \%1^{(1/3)} - 2\,\frac{a}{\%1^{(1/3)}}\right),$$

$$-\frac{1}{12}\ \%1^{(1/3)} - \frac{a}{\%1^{(1/3)}} - \frac{1}{2}\,I\sqrt{3}\left(\frac{1}{6}\ \%1^{(1/3)} - 2\,\frac{a}{\%1^{(1/3)}}\right)$$

$$\%1 := 108 + 12\sqrt{-12\,a^3 + 81}$$

The result of `solve` is a sequence of three numbers. This is printed with the use of an **abbreviation** `%1` for a common subexpression. The last line of the output describes the meaning of this abbreviation.

The abbreviation system is standard in releases before V.5 and in text-only versions of release V.5, but in windowing versions of release V.5 abbreviations are used only if you change **Output Display** from Editable Math Notation (the default) to Typeset Notation (or one of the other two choices) in the Options menu. For improving readability, all the output in this book is printed in Typeset Notation with these abbreviations.

The present abbreviation can also be called by the user as long as Maple has not cleared it out:

```
> %1;
```

$$108 + 12\sqrt{-12\,a^3 + 81}$$

If we ask for the first solution, Maple does not use this abbreviation, because, in each case, Maple decides separately which abbreviations are useful, if any. For instance, let's ask for the first solution only by taking the first element of the second last result:

```
> %%[1];
```

$$\frac{1}{6}\left(108 + 12\sqrt{-12\,a^3 + 81}\right)^{(1/3)} + 2\,\frac{a}{\left(108 + 12\sqrt{-12\,a^3 + 81}\right)^{(1/3)}}$$

1.12 On-line help

Maple offers a simple and comprehensive guide to its facilities by its on-line help. The `Help` menu activates a hypercard system (only from Release 4) for a search for help on the word at the cursor (possibly the first characters of a word) or a search by topic or a search for all places in the help texts for a word. In a help page you can click with the mouse on an underscored word for getting specific information on that subject.

For instance, we can ask for help on substitution. We choose `Topic Search` from the Help menu. As soon as we have typed one or more letters, we see a list of topics, the first letters of which are equal to the letters entered. In this way, we can ask for help on the procedure `subs`. In all versions of Maple, also in text-only (tty) versions of Maple, there is another method of getting on-line help: type a question mark and then the catchword.

Now let's suppose that we ask for help on the topic `subs`. In the following, the help text given by Maple is printed in parts with some explanations thereafter:

First, the purpose of `subs` and the syntactically correct way to use `subs` is described in a formal way.

Function: subs - substitute subexpressions into an expression

Then you can see how `subs` can be used:

Calling Sequence:

subs(x=a,expr)

subs(s1,...,sn,expr)

Up to now, only the first way of using `subs` has been demonstrated.
Then you can see what type of parameters can be used. Types will be explained in section 11.3 on page 139 and section A.1 on page 275.
Now a description of the action of `subs` is given:

Description:

- The first form of the `subs` command substitutes `a` for `x` in the expression `expr`.

The remaining part of the description is omitted here; these remarks require explanations given later in this book.

Often, the fastest way to get an idea of the use of a command is reading the "Examples" at the end of the help text. If you had forgotten the way `subs` should be used, the first example might suffice.

Examples:

```
> subs( x=2, x^2+x+1 );
```

The remaining examples are left out here.
This information can be retrieved separately by the command `???subs`.

At the very end of the help text, Maple refers to related subjects. This can be very helpful, for instance, when you do not know the name of a procedure that can do the job; if you know a name of a procedure that does something related, you can ask for help for the known procedure and look at the end.

See Also: op, subsop, eval, algsubs, limit

This list of related topics can be retrieved separately by the command `related(subs)`. Each of the underlined words can be clicked on in order to get information about this topic.

The help system can do more than just explain Maple procedures. For example, you can get a description of the changes made in new releases of Maple under the topic `updates`.

If you want to search the on-line help system systematically on a text-only system, enter `?index` to see the categories available.

chapter 2

Numbers and algebraic operators

In this chapter, various types of numbers are discussed together with their algebraic operators: rational numbers, radicals, special real numbers such as π, complex numbers, floating-point approximations, integers, and Z modulo n.

Tools for manipulating and converting numbers are discussed in Chapter 12, Manipulating and converting numbers, *but the basic ideas for manipulating radicals and complex numbers are demonstrated in the present chapter.*

2.1 Algebraic operators

In Maple, the main algebraic operators for sum, difference, product, quotient, and power are entered successively as `+`, `-`, `*`, `/` and `^`. Parentheses are entered as usual in mathematics:

```
> p*(2*a-b)^5/(c+7);
```

$$\frac{p\,(2\,a-b)^5}{c+7}$$

The results found by Maple are presented in the customary two-dimensional way, but a formula must be entered in a linear way. For copy purposes, you can print a formula in a linear way with `lprint`:

```
> lprint(%);
p*(2*a-b)^5/(c+7)
```

Maple does not interpret `ax` as the product of `a` and `x`, but as a two-letter name `ax`.

> Each multiplication is to be entered with an asterisk.

In printing output, multiplications are represented by spaces. However, this method of notation is not accepted as input.

It is tempting to make mistakes such as the following:

```
> a(b+c);
```

$$\mathrm{a}(b+c)$$

At first glance, it may not be obvious that the result is *not* the product of `a` and `b+c`. Actually, Maple supposes that `a` is a procedure (maybe a function) that is applied

to the sum of b and c. If you are using Maple in a windowing version, you can see that a is printed *upright*, while the other characters are *slanted*, indicating that Maple supposes that a is a function or procedure acting on $b + c$.

At first sight, the following result might look strange:

```
> solve( x-3 = 7(p+q) , x );
```

$$10$$

Maple has interpreted 7(p+q) as application of the constant function 7 to the parameter p+q, with the obvious result 7. This interpretation would have come out by Maple's screen output, if the equation had been entered *before* applying `solve`:

```
> x-3 = 7(p+q); solve( % , x );
```

$$x - 3 = 7$$

$$10$$

Any user can make mistakes in typing formulas. Advice:

> Before applying a procedure to a formula
> that has to be typed in,
> it is better to enter this formula first,
> check the result printed by Maple,
> and then, if this is correct, apply the procedure.

In mathematics, the **dot** can be used in denoting multiplication, but not so in Maple: it is used in floating-point numbers and for gluing strings together, i.e., concatenation:

```
> toget.her;
```

together

2.2 Parentheses and precedence rules

Maple respects the standard **precedence rules** of arithmetic operations. Please check if the following two results are what you expect.

```
> 100 - b + 2*3^2;
```

$$118 - b$$

This is interpreted as

$$(100 - b) + (2 * (3^2))$$

```
> 1 / p*q / r-1 ;
```

$$\frac{q}{pr} - 1$$

This is interpreted as

$$\frac{\frac{1}{p}q}{r} - 1$$

This last example may be another argument for entering a formula first, checking the result on the screen, and only then applying a procedure.

Contrary to some other computer languages, Maple conforms to mathematics in the handling of minus a power. For example

```
> -3 ^ 4 ;
```

$$-81$$

The layout of the input in the last example might be misleading, the spaces suggesting a different interpretation. If you want the fourth power of -3, do not forget to use parentheses:

```
> (-3) ^ 4 ;
```

$$81$$

Sometimes Maple demands more parentheses:

```
> 3 * |-4 ;
Syntax error, `-` unexpected
```

Correct input is:

```
> 3 * (-4);
```

$$-12$$

The general rule in Maple:

> Never enter an arithmetic operator side by side with minus: they should be separated by a parenthesis.

Moreover, due to the nonassociativity of the operator ^ , Maple demands parentheses in the following nad indicates an error with de cursor at the second ^:

```
> 2 ^ 3 |^ 2;
Syntax error, `^` unexpected
```

Correct input is:

```
> 2 ^ (3 ^ 2);
```

$$512$$

2.3 Rational numbers

When Maple divides two integers, where the second is not 0, it yields a rational number after automatic dividing out the gcd of numerator and denominator.

```
> 26 / (-8);
```

$$\frac{-13}{4}$$

Approximation by a floating-point number can be obtained by the command `evalf`, as discussed later in this chapter.

Be careful when entering **decimal fractions**. For example, let's calculate 1.45/3.

```
> 1.45/3 ;
```

$$.4833333333$$

Maple has calculated an approximation, as it interprets the floating-point number 1.45 as an approximation already. If you want to obtain an exact result for dividing the rational number 1.45 by 3, enter 145/100 instead of 1.45.

```
> 145/100/3;
```

$$\frac{29}{60}$$

See the section *Floating-point numbers, approximations* of the present chapter for more details.

An important Maple rule is the **autosimplification of rational numbers**: the operators +, -, *, /, and ^ between rational numbers are activated immediately, with one exception: a power with a noninteger exponent is not simplified.

2.4 Real constants

Some special mathematical constants are known to Maple:

- π is to be entered as **Pi**, but is printed as π in windowing versions.
- **Catalan** := $\sum_{i=0}^{\infty} \frac{(-1)^i}{(2i+1)^2}$.
- γ := the Euler constant := $\lim_{n\to\infty}((\sum_{i=1}^{n} \frac{1}{i}) - \ln(n))$, printed as γ.

These names are unassigned, but several procedures can handle them, for example,

```
> sin(100000000/3*Pi);
```

$$-\frac{1}{2}\sqrt{3}$$

In releases before V.4, there is another constant **E**, denoting the base of the natural logarithm, in mathematics generally denoted as e.

2.5 Complex numbers

The **complex** number i is represented in Maple as **I**:

```
> I^2;
```

$$-1$$

In fact `I` is an 'alias' for `(-1)^(1/2)`:

```
> (-1)^(1/2);
```

$$I$$

The `alias` construction is discussed in section B.1 on page 285. There you will also see that you can choose another name for $\sqrt{-1}$.

Here are some calculations with complex numbers.

```
> (2 + 3*I)*(1 - I);
```

$$5 + I$$

```
> 1/(3-4*I);
```

$$\frac{3}{25} + \frac{4}{25}I$$

```
> (1+I)^6;
```

$$-8I$$

In other cases you must ask Maple to perform calculations with the procedure **evalc** ("evaluate as a complex number"):

```
> (1+I)*(x+I);
```

$$(1 + I)(x + I)$$

```
> evalc(%);
```

$$x - 1 + I(1 + x)$$

The procedure `evalc` tries to convert a complex number into the form $a + bI$, where a and b are real numbers. Be careful:

The procedure `evalc` assumes
that all names not referring to some object
are real variables.

```
> evalc( exp(a*I) );
```

$$\cos(a) + I\,\sin(a)$$

This is a correct splitting into a sum of the real and the imaginary component of $\exp(ai)$ only if a is real. Maple does not expect the user to substitute, for instance, $2 * I$ for a.

Numbers of the type $a + b\sqrt{-1}$, where a and b are rational, form a special class of numbers in Maple; these are called the **rational complex numbers.**

2.6 Radicals

A radical can be generated as a power. For instance, $\sqrt[4]{50} + \sqrt[6]{72}$ is entered as:

```
> 50 ^ (1/4) + 72 ^ (1/6);
```

$$50^{(1/4)} + 72^{(1/6)}$$

This can be converted to standard form by the procedure **simplify**:

```
> simplify( % , radical );
```

$$2^{(1/4)}\,25^{(1/4)} + \sqrt{2}\,9^{(1/6)}$$

If the second argument `radical` is omitted, the same result is found here; but if large expressions are to be handled, it is more prudent to specify the type of simplification in order to avoid unwanted effects.

A **square root** can be denoted with the procedure **sqrt**, too. This has built-in simplification:

```
> sqrt(48/25);
```

$$\frac{4}{5}\sqrt{3}$$

```
> sqrt(8-sqrt(15));
```

$$\frac{1}{2}\sqrt{30} - \frac{1}{2}\sqrt{2}$$

For radicals of complex numbers, `evalc` can be used in order to get the standard form:

```
> (-I)^(1/4);
```

$$(-I)^{(1/4)}$$

```
> evalc(%);
```

$$\frac{1}{2}\sqrt{2+\sqrt{2}} - \frac{1}{2}I\sqrt{2-\sqrt{2}}$$

The procedure `evalc` interprets a power of a complex number z^a by conceiving z as $|z|\, e^{i\phi}$ with $-\pi < \phi \leq \pi$ and calculating $|z|^a\, e^{ia\phi}$. In particular, it takes the principal branch of $x \to x^{1/n}$:

```
> (-625)^(1/4); evalc(%);
```

$$(-625)^{(1/4)}$$

$$\frac{1}{2}625^{(1/4)}\sqrt{2} + \frac{1}{2}I\,625^{(1/4)}\sqrt{2}$$

```
> simplify(%,radical);
```

$$\frac{5}{2}\sqrt{2} + \frac{5}{2}I\sqrt{2}$$

Automatic simplification of the nth root is offered by **root[n]( )**:

```
> root[4](-625);
```

$$5\,(-1)^{(1/4)}$$

Another interpretation of the nth root is available. In calculations with real numbers you want to find a negative real number for an odd root of a negative number. This can be found by **surd**, for instance, $\sqrt[3]{-125}$:

```
> surd(-125,3);
```

$$-5$$

For complex numbers, `surd` can be used as well, generally not yielding the principal value, but a root with nearest argument to that of the original (useful for deviations in numeric calculations).

2.7 Manipulating radicals and complex numbers—an example

As an introduction to manipulating radicals and complex numbers, here is an example:

```
> evalc(I^(1/4)) ;
```

$$\frac{1}{2}\sqrt{2+\sqrt{2}} + \frac{1}{2}I\sqrt{2-\sqrt{2}}$$

Now let's try to convert the fourth power of this to standard complex form in order to get I again.

```
> % ^ 4;
```

$$\left(\frac{1}{2}\sqrt{2+\sqrt{2}}+\frac{1}{2}I\sqrt{2-\sqrt{2}}\right)^4$$

This power is not elaborated automatically, so we must tell Maple to do that:

```
> expand( % );
```

$$\frac{1}{2}I\sqrt{2+\sqrt{2}}\sqrt{2-\sqrt{2}}\sqrt{2}$$

The last result contains a product of square roots, where `I` is interpreted as a square root by Maple as well. These four roots can be combined into the square root of just one expression with the procedure `combine`:

```
> combine(%);
```

$$I$$

The procedure `combine` could have been given an option `radical` in this case in order to restrict its actions.

2.8 Floating-point numbers, approximations

As shown previously, there is an essential difference between a decimal fraction and a floating-point number:

```
> sqrt( 13/10 ) , sqrt( 1.3) ;
```

$$\frac{1}{10}\sqrt{130},\ 1.140175425$$

In the second expression, Maple assumes that the user wants an approximation as the result, because it considers `1.3` as an approximation.

> In any case where Maple encounters
> a floating-point approximation in an expression,
> Maple supposes that the user wants an approximation as the result.

This should be kept in mind, for instance, when a square root expression is wanted and not an approximation:

```
> 80^0.5, 80^(1/2);
```

$$8.944271910,\ \sqrt{80}$$

The general format for a floating-point number is scientific notation, containing a factor 10^n:

```
> 1.234e+50 , 0.98e-7;
```

$$.1234\,10^{51},\ .98\,10^{-7}$$

The standard procedure available for **numerical approximation** is **evalf** (evaluate to a floating-point number):

```
> evalf( ln(2) , 20 );
```

$$.69314718055994530942$$

The second argument specifies the number of digits in the calculation. When the number of digits is not specified, calculations with floating-point numbers are executed with a fixed number of digits; the standard number is 10. This default is set by the value of the name **Digits**. The value can be changed by assigning another number to `Digits`, with a maximum as high as 500,000 (student version: 100).

Maple does not claim that all digits are correct in numerical computations; in this respect it acts like most numerical packages and calculators. But eventually, Maple has an advantage: it can execute numerical calculations with very large numbers of digits.

Sometimes, `evalf` yields a relatively small number that can better be neglected. For instance, it is possible that you expect a real value in a calculation, but get a small imaginary component when you apply `evalf`. In such cases you can use **fnormal**

```
> convert(tan(1),exp);
```

$$-\frac{I\left(\left(\mathbf{e}^{(I)}\right)^2-1\right)}{\left(\mathbf{e}^{(I)}\right)^2+1}$$

```
> evalf(%);
```

$$1.557407725 + .1678572783\,10^{-9}\,I$$

```
> fnormal(%)
```

$$1.557407725$$

For **fast numerical calculations**, Maple can use fast floating-point procedures (double precision) from the C-library on the system. The procedure for this purpose is **evalhf**:

```
> appr := evalhf( ln(2) );
```

$$appr := .6931471805599453$$

Here the number of digits cannot be specified; it is determined by the actual system.

2.9 Some effects of automatic simplification of floating-point numbers

If Maple switches over to approximation automatically, the number of digits it uses in calculations is determined by the value of `Digits`. It does not help if you use `evalf` in such a case:

```
> evalf( 80^0.5 , 40 );
```

$$8.944271910$$

It is obvious that this does not work. Evaluation, including automatic simplification, is applied on the argument before the procedure comes into action. So, Maple has approximated the power before `evalf` comes into action, and then it is too late for `evalf` to get a result to 40 digits. The remedy would be assigning 40 to `Digits`, before this calculation.

Using floating-point numbers in calculations can generate unexpected results:

```
> evalf( Pi , 30 );
```

$$3.14159265358979323846264338328$$

```
> evalf( Pi , 50 );
```

$$3.1415926535897932384626433832795028841971693993751$$

```
> % - %% ;
```

$$0$$

Maple yields zero for the difference because the last command has been executed as a numerical calculation with 10 digits precision. When we set `Digits` to 50, we get what we want:

```
> Digits := 50:
> evalf( Pi , 30 ) - evalf( Pi , 50 );
```

$$.4971158028306006249\,10^{-30}$$

More on this subject can be found in Chapter 9, *Numerical calculations with Maple*.

2.10 Calculations with integers

Maple can use very large integers:

```
> 7 ^ 1000;
```

```
125325663996571831810755483238273420616498507508098617l4\
   6349500752097059631738l1643244883905435152076319861591\
   1955159407668582898946726302276179083827085457983001 5\
   1112466612039846243589298325716157180147040963056680 9\
   7507613273663023226895250541385927158426088684494082 4\
   1676861770818959228693603992231112568371921504668915 6\
   7383525901372415545101858559645499275754932473911325 4\
   8534378497978806084951085874202011836362315727420109 5\
   5478298879153008828971184455050023048563841318994713 2\
   1422439473341992593007356224929374194536500614903021 0\
   5127920314430401636855677549136337481321811349678427 0\
   7609143734504539933734861126116805592935540299282319 2\
   4911903600270361122831809358727752145174640131782746 5\
   7100736321564606838252739601156414628445543663144696 0\
   5065016081262181432706266619517270178020028664502382 3\
   0831859280613713103008292840711412077312806000 01
```

There is a limit, but a very generous one. Generally, Maple can manage integer numbers of more than half a million digits. If Maple foresees that an object cannot be handled, calculation is stopped:

```
> 7 ^ 1000000;
Error, object too large
```

For **factorial**, the operator **!** can be used:

```
> 5! ;
```

$$120$$

Division of an integer by an integer with the operator **/**, generally, yields a rational number. Integer division with remainder can be executed by **iquo** and **irem**:

```
> iquo( 705 , 7 ), irem( 705 , 7 );
```

$$100, 5$$

Both results can be computed at the same time as follows:

```
> iquo( 705 , 7 , 't' );
```

$$100$$

```
> t;
```

$$5$$

The mechanism behind this construction is explained in section 5.4 on page 67.

For more information on calculations with integers and on using Maple in number theory, consult on-line help about `binomial`, `ifactor`, `ifactors`, `igcd`, `igcdex`, `ilcm`, `issqr`, `nextprime`, `prevprime`, `ithprime`, `combinat`, and `numtheory`.

2.11 Integers modulo an integer

Calculations modulo an integer can be carried out with **mod**. The arithmetic operators +, -, and * can be used in conjunction with `mod` an integer, and even division, if possible, can be carried out with / .

```
> 20*x^5 - 12*x - 2 mod 3;
```

$$2x^5 + 1$$

```
> 1/2 mod 9;
```

$$5$$

Calculation of powers in conjunction with `mod` can be considerably faster with `&^`. See also the on-line help about `Power` and `Powmod`.

```
> 7 &^ 1000000 mod 3;
```

$$1$$

These constructions with `mod` can be applied in cases with variables as well, for instance on polynomials over Z mod n. For univariate and bivariate polynomials over the integers modulo a prime p, `modp1` and `modp2` are available. In section 18.20 on page 273, matrices with elements in Z mod n are demonstrated. For more information on this subject, consult the on-line help about `mod` and `inert`.

For the Chinese remainder algorithm, the procedure `chrem` is available.

2.12 Algebraic extensions and general rings

Calculations in Maple using algebraic extensions of the rational field are discussed in section 14.10 on page 183 and in Chapter 15, *Manipulating algebraic expressions*. You can also consult the on-line help about `evala`.

For the creation of an abstract **group** or an abstract **operator** the procedure `define` can be used. Moreover, Maple has special packages for **peadic numbers** `padic`, **Gaussian integers** `GaussInt` and for **Galois fields** `GF` (which can also be created in `Domains`), and the procedure `Berlekamp`. These subjects are beyond the scope of this book.

chapter 3

Names and evaluation 1: mathematical variables

Maple uses numbers, mathematical variables, algebraic expressions, mathematical functions, matrices, and many other types of objects. This chapter discusses the mechanisms in Maple for handling names that refer to these objects.

The Maple commands in this chapter make up an unbroken Maple session, as it is the case with each chapter.

More about names and evaluation can be found in Chapter 5, Names and evaluation 2: applying procedures, *and in Appendix B,* Names and evaluation 3: some special features.

3.1 Assigning names to objects and evaluating names to objects

In Maple, a name can be used without a value, emulating an undetermined mathematical variable; for instance, `X` and `a` in the following expression:

```
> 3*X^2 + a;
```

$$3X^2 + a$$

Please pay attention to the use of *upper-case* and *lower-case* letters: both can be used in names, and Maple distinguishes between them. For instance, Maple perceives `x` and `X` as different names.

Any name in Maple can *refer* to another Maple object. For instance,

```
> a := 1000;
```

$$a := 1000$$

We say: "the number `1000` has been **assigned** to the name `a`."

From now on, generally, Maple reads `1000` for each occurrence of `a`. We say: "the name `a` **refers** to the number `1000`":

```
> a^2*t - 2*a - 1;
```

$$1000000\,t - 2001$$

We say: "Maple has **evaluated** `a^2t - 2 a - 1`." The term *evaluation* is given several meanings in various computer languages. Strictly speaking, Maple evaluation is finding the *values* of the variables (by the process of searching the memory

for references of names), and does not imply any calculations. In Maple terminology, a calculation is called a **simplification**. Generally, simplifications must be requested by the user, but some basic simplifications are executed automatically, such as calculating the square of 1000 and the combination of -2000 and -1 in the last example. So Maple has calculated the result in that example by

- evaluating a to 1000
- **autosimplification** of the resulting expression
- sorting subexpressions in the resulting expression according to the internal order in the Maple memory.

Evaluation is always followed by this autosimplification and sorting. Therefore, this whole process also is called **evaluation** usually and so we do in this book.

3.2 Assigning names and expressions to a name

Now let's see what happens when we assign the first expression to a name.

```
> expr := 3*X^2 + a;
```

$$expr := 3\,X^2 + 1000$$

What Maple has done is the following:

- The right-hand side has been evaluated to $3X^2 + 1000$.
- This result has been put on the 'stack' for reference by the **ditto** (%).
- The left-hand side, the name `expr`, is made to refer to this expression.

This last action is represented by Maple's response. If you had assigned something to a previously and would have forgotten about that, you would not have expected this result. Therefore, you better check the result. Advice:

> When you enter an assignment, check its effect by viewing the result that is printed to the screen.

The name a has been used in assigning a value to `expr`. But a has been evaluated *before* the assignment. This makes `expr` refer to something that has nothing to do with a. Therefore, if we now change the value of a, the value of `expr` is not affected.

```
> a := 777 ;
```

$$a := 777$$

```
> expr ;
```

$$3\,X^2 + 1000$$

Now let's assign something to X and then ask for `expr`.

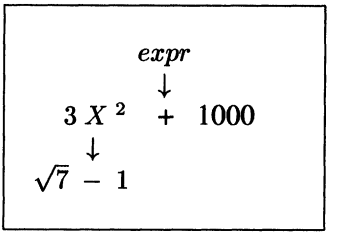
```
> X := sqrt(7)-1;
```

$$X := \sqrt{7} - 1$$

```
> expr;
```

$$3\left(\sqrt{7}-1\right)^2 + 1000$$

After one step of evaluation of the name `expr`, Maple got $3X^2 + 1000$, but now `X` refers to something, so Maple takes another step in evaluating `X`. The situation is shown in the following diagram:



You can ask Maple to evaluate exactly one step or a number of steps by the procedure **eval**.

```
> eval(expr,1);
```

$$3X^2 + 1000$$

```
> eval(expr,2);
```

$$3\left(\sqrt{7}-1\right)^2 + 1000$$

The standard in Maple is **full evaluation**, with a few exceptions, discussed in Appendix B, *Names and evaluation 3: some special features*.

It may be clear that changing the value of `X` changes the result of evaluating `expr`.

```
> X := 5*Pi;
```

$$X := 5\pi$$

```
> expr;
```

$$75\pi^2 + 1000$$

It is essential to consider present references of names involved in assignments, because the result of evaluating the right-hand side is assigned to the left-hand side. You can test whether you understand this mechanism by predicting the results of the following commands, then checking your predictions by entering the lines.

```
> t1 := (s^2 - 1);
```

$$t1 := s^2 - 1$$

```
> s := 10:
> t2 := (s^2 - 1);
```

$$t2 := 99$$

```
> t1 , t2 ;
```

$$99, 99$$

```
> eval(t1,1) , eval(t2,1);
```

$$s^2 - 1, 99$$

```
> s:= 100: t1 , t2 ;
```

$$9999, 99$$

3.3 Unassigning

A name can be unassigned by the aid of **single forward quotes.**

```
> a := 'a';
```

$$a := a$$

As usual in assignments, the right-hand side is evaluated first. Evaluating something between forward quotes prevents Maple from looking in the memory for references of the names between these forward quotes; instead of this, Maple peels off that pair of forward quotes. Here the right-hand side evaluates to the name `a` and the assignment determines the name `a` to once again have no reference. The command may suggest that `a` would refer to itself in a recursive way, but this is prevented by Maple. See the section *Recursive definitions of names* at the end of this chapter.

Often, problems arise when the user forgets about an assignment earlier in the session, and thinks that a name is unassigned.

> It is wise to unassign all names
> as soon as their values become useless.

Let's act on this advice and unassign `X` and `expr`, previously assigned:

```
> X:='X' :  expr:='expr' :
```

We could have unassigned these names also with the procedure `unassign` as follows:

```
> unassign('X, expr');
```

Do not forget the single quotes, otherwise the arguments are evaluated before they can be unassigned.

If you want to unassign all names used, including all procedures and tables read from the library, you can enter **restart**. This brings the Maple engine nearly to the state of start-up, but does not erase your worksheet.

3.4 Names and properties

Maple can give a name or an expression a property with the aid of the procedure **assume**:

```
> assume( p < 0 );
> assume( x > 1 );
```

Many procedures can use these assumptions:

```
> abs( x*p );
```

$$-p\sim\ x\sim$$

Maple informs the user that p and x have properties by printing their names with a trailing **tilde** ($\sim$). (In windowing versions you can make other choices.)

Maple can use this property for calculations, for instance in calculating $\lim_{n\to\infty} x^n$ and $\lim_{n\to\infty}(\frac{1}{x})^n$.

```
> limit( x^n , n=infinity );
```

$$\infty$$

```
> limit( (1/x)^n , n=infinity );
```

$$0$$

If we had not made such an assumption about x, Maple would have returned the command unevaluated.
However, not all procedures take such properties into account. For instance, in the following command, **solve** does not take into account that x has been assumed to be greater than 10:

```
> solve( x^2=400 , x );
```

$$20, -20$$

The assumptions about x can be omitted in the same way as x can be unassigned.

```
> x := 'x'; p := 'p';
```

$$x := x$$

$$p := p$$

The assume facility is explained in a more comprehensive way in Appendix A, *Types, properties, and domains.*

3.5 Combinations of characters that can be accepted as names

Maple accepts almost any combinations of lower-case letters (a, . . . , z), upper-case letters (A, . . . , Z), digits, and the underscore character (_) as a name, if the first character is not a digit.

However, the following **keywords** of the Maple language cannot be used as names in a simple, direct way.

and	by	do	done	elif
else	end	fi	for	from
if	in	intersect	local	minus
mod	not	od	option	options
or	proc	quit	read	save
stop	then	to	union	while

For example,

```
> in := 5 ;
Syntax error, ':=' unexpected
```

Moreover, many names used by Maple, for instance names of procedures, are **protected**; if you try to assign something to such a name, Maple protests.

```
> abs := 3;
Error, attempting to assign to 'abs' which is protected
```

This protection can be removed by the procedure `unprotect`.

In practice, there can be more restrictions in the use of combinations of characters as names. For example,

```
> I := 20;
Error, Illegal use of an object as a name
```

This rather mysterious message of Maple arises from the fact that `I` is an **alias** of `sqrt(-1)`, so Maple reads the command as "`sqrt(-1):=20;`" and does not accept this. More about `alias` can be found in Appendix B, *Names and evaluation 3: some special features.*

Moreover, a name used in an option should not be assigned.

Although a name can have the **underscore** (_) as its first character, a not very advanced user should not use such a name, because Maple uses this category of names for internal purposes. For example, the procedure `integrate` substitutes the name `_X` for the integration parameter before it tries to calculate a result. Then, if Maple finds a result, this `_X` is exchanged back for the original parameter.

A special class of names is formed by the **environment variables** of Maple, containing especially `Digits`, `Order`, `printlevel`, `mod`, `Normalizer` and `Testzero`. Maple prevents an assignment to one of these if it makes no sense. For instance, the number of digits in a floating-point approximation should be a natural number, so the following command is rejected:

```
> Digits := sqrt(101) ;
Error, invalid assignment to Digits
```

Another special class of names is formed by the **constants**:

```
> constants;
```

$$false, \gamma, \infty, true, Catalan, FAIL, \pi$$

```
> Pi := 3.14;
Error, may not assign to a system constant
```

The name **Pi** does not refer to anything else, though the procedure `evalf` can find floating-point approximations, but it is not possible to assign a value to this name because it is contained in the sequence of constants.

3.6 Greek letter names

In windowing versions of Maple, the names of the Greek letters are printed as Greek letters.

```
> phi, Phi;
```

$$\phi, \Phi$$

Be careful with the name `Pi`: it is printed by Maple as π, exactly in the same way as the name `pi` is printed. But Maple takes `Pi` as the mathematical constant π, while `pi` is a name like other names and can be used as a variable.

```
> sin(pi), sin(Pi);
```

$$\sin(\pi), 0$$

```
> 5*Pi+7*pi ;
```

$$5\pi + 7\pi$$

In the last result, both terms contain π, but the two are different, so the terms cannot be combined into 12π. You can see this by applying **lprint**:

```
> lprint(%);
5*Pi+7*pi
```

More possible causes for confusion of names can be found in Chapter 6, *Creating and using mathematical functions* in the section *Denotation of the functions* `exp`, `Gamma`, *and* `Zeta`.

3.7 Names with an index

A name with an index can be created with square brackets:

```
> A[n];
```

$$A_n$$

In fact, Maple reads `A[n]` as an element of a *table* with the name `A`; see section 10.10 on page 135.
Suppose, for instance, that you want to create a truncated power series with undefined coefficients. You can do that as follows:

```
> ser := sum( c[i]*x^i , i=0..8 );
```

$$ser := c_0 + c_1\,x + c_2\,x^2 + c_3\,x^3 + c_4\,x^4 + c_5\,x^5 + c_6\,x^6 + c_7\,x^7 + c_8\,x^8$$

More on this subject can be found in section B.3 on page 287.
As an alternative you can use

```
> ser := sum( 'c.i'*x^i , i=0..8 );
```

$$ser := c0 + c1\,x + c2\,x^2 + c3\,x^3 + c4\,x^4 + c5\,x^5 + c6\,x^6 + c7\,x^7 + c8\,x^8$$

The period between the characters `a` and `i` is Maple's **concatenation** operator: it evaluates `i` at its right side and then glues the result to the symbol `a` at its left side. It works, but the result looks less attractive. Moreover, the **forward quotes** are necessary here, because, generally, the arguments to a procedure are evaluated before they are handed over to the procedure; see Chapter 6, *Creating and using mathematical functions*. So, if we had not used forward quotes, the concatenation would have been executed before the procedure `sum` could have substituted the successive values for `i`:

```
> misser := sum( c.i*x^i , i=0..8 );
```

$$misser := ci + ci\,x + ci\,x^2 + ci\,x^3 + ci\,x^4 + ci\,x^5 + ci\,x^6 + ci\,x^7 + ci\,x^8$$

3.8 Single back quotes

If you want to use a name that is not acceptable to Maple, you can make it acceptable by enclosing it in back quotes.
For instance, here are names containing spaces, exclamation, and question marks:

```
> `the largest number!!` := 100000000000000000;
```

$$`the\ largest\ number!!` := 100000000000000000$$

```
> `a still larger number?` := `the largest number!!`+1;
```

$$`a\ still\ larger\ number?` := 100000000000000001$$

In this way, a keyword can be used, too.

```
> `quit` := 0;
```

$$`quit` := 0$$

Making a sequence of characters acceptable as a name is the only effect of a pair of back quotes. For instance,

```
> t:=10: `t`;
```

$$10$$

The pair of back quotes around `t` here is superfluous, as the name `t` is acceptable. So Maple neglects this pair of back quotes. For the same reason, back quotes cannot change the behavior of an alias.

```
> `I` := 3 ;
Error, Illegal use of an object as a name
```

The role of a pair of **back quotes** is to make the command line interpreter accept a combination of characters as a whole symbol, a name that can refer to other objects. A pair of back quotes does not prevent evaluation.

Be careful not to enter unpaired back quotes. If you do, you will see strange warnings:

```
> cos(`new var);
Warning, incomplete quoted name;  use ` to end the name
```

You might think that entering the second back quote could satisfy Maple's syntax checker, but that is not so:

```
> `|;
Syntax error, `;` unexpected
```

Here Maple expects you to enter the closing parentheses, so make the correction

```
> `);
```

```
cos('new var');
  )
```

Now Maple is kindly waiting for new input.

3.9 The concepts of name, symbol, and string in Maple

In many computer languages, such as Pascal and Basic, a string is constructed by enclosing characters between a pair of single or double quotes. In Maple V release 5, such a string can be constructed by enclosing it between a pair of **double quotes**. (In previous releases the double quote is used for the ditto). But a name is something quite different: a name can refer to something else and a name can act as a variable. A pair of single back quotes urges Maple to accept the sequence of characters between them as a name.

Likewise, enclosing a name between a pair of **single forward quotes** may resemble the idea of creating a string in other computer languages, but again, it is quite different: these forward quotes preserve the content from looking up its value, as the evaluation peels off only this pair of forward quotes.

```
> 'a' + ''b'' + '''c''';
```

$$a + 'b' + ''c''$$

Forward quotes do *not* withhold Maple from elementary calculations:

```
> '6 + ((a^2)^3) + 4';
```

$$10 + a^6$$

3.10 Recursive definitions of names

There is one more pitfall in assignment.

```
> k := k+3;
Warning, recursive definition of name
```

$$k := k + 3$$

Maple detects a loop in the reference of k and warns the user. You can see this loop by evaluating k only one step.

```
> eval(k,1);
```

$$k + 3$$

If you don't give the command

```
> k := 'k';
```

$$k := k$$

but let Maple evaluate k, you get a message

```
Error, too many levels of recursion
```

or Maple crashes.

In more complicated cases, Maple may not detect such recursive definitions in time to warn the user.

If Maple issues a message such as the above or if Maple crashes, in most cases Maple has tried to evaluate a name that refers to itself by a more or less complicated detour.

chapter 4

Elementary calculus

This chapter deals with calculating derivatives, antiderivatives (or primitive functions, indefinite integrals) and definite integrals, as well as calculating sums and products. Although Maple is very powerful in calculating antiderivatives, sometimes it can be useful to help it a little bit; this chapter pays attention to several ways to do so, demonstrated by simple examples. Moreover, reliability and methods for checking are discussed.

Calculus is based on the concept of functions. It is often more convenient to use functions than expressions in calculations. In Chapter 6, Creating and using mathematical functions, *the use of functions is discussed, especially in connection with calculus.*

Applying series approximations, for instance for antiderivatives, is shown in Chapter 8, Taylor or Laurent expansion and limits. *In that chapter limits are dealt with, too.*

4.1 Differentiation

An expression interpreted as a function in one variable can be differentiated with the procedure **diff**. In order to check the correctness of the command before it is evaluated, you can enter it between a pair of forward quotes:

```
> 'diff( exp(-a*x^2) , x )';
```

$$\frac{\partial}{\partial x}\,\mathbf{e}^{(-a\,x^2)}$$

The name `diff` is evaluated in the next step, using the ditto, and so the corresponding procedure becomes active:

```
> %;
```

$$-2\,a\,x\,\mathbf{e}^{(-a\,x^2)}$$

The second argument for `diff` must be a name that does not refer to something else.

Higher order and partial differentiation can be obtained by adding more arguments to `diff`. For instance, $\frac{\partial}{\partial a}\,\frac{\partial}{\partial x}\,\exp(-a\,x^2)$ can be calculated by

```
> diff( exp(-a*x^2) , x, a);
```

$$-2\,x\,\mathrm{e}^{(-a\,x^2)} + 2\,a\,x^3\,\mathrm{e}^{(-a\,x^2)}$$

In the same way $\frac{\partial^3}{\partial x^3}\exp(-a\,x^2)$ can be asked for:

```
> diff( exp(-a*x^2) , x, x, x );
```

$$12\,a^2\,x\,\mathrm{e}^{(-a\,x^2)} - 8\,a^3\,x^3\,\mathrm{e}^{(-a\,x^2)}$$

This last command can be abbreviated to `diff( exp(-a*x^2) , x$3 )`. The operator **$** is used for generating a sequence of equal objects, for instance,

```
> again $ 7;
```

again, *again*, *again*, *again*, *again*, *again*, *again*

Undefined functions can be used as well:

```
> diff( cos(f(x)) , x );
```

$$-\sin(\mathrm{f}(x))\left(\frac{\partial}{\partial x}\,\mathrm{f}(x)\right)$$

```
> diff( f(cos(x)) , x );
```

$$-\,\mathrm{D}(f)\,(\cos(x))\sin(x)$$

In the last result `D(f)` stands for the derivative function of `f`. The procedure `D` is discussed in section 6.9 on page 79.

It is not possible to differentiate with respect to an expression such as $u(t)$:

```
> diff( exp(u(t)) , u(t) );
Error, wrong number (or type) of parameters in function diff
```

Only a name can be accepted as a variable in respect to which an expression is to be differentiated. In a case like this, you can first substitute `u` for `u(t)`, ask for differentiation to `u`, and then back substitute `u(t)` for `u`.

Differentiation is a straightforward algorithm and does not generate **reliability** problems in principle.

4.2 The derivative at a point

There are two ways to find the derivative at a point.

A. The obvious way is by evaluating the derivative:

```
> exp(-x^2+x);
```

$$e^{(-x^2+x)}$$

```
> diff(%,x);
```

$$(-2x+1)\,e^{(-x^2+x)}$$

```
> eval(%,x=1);
```

$$-1$$

In releases before V.5 `eval` cannot be used in this way. In this case, substitution with the procedure `subs` could do the same job. However, there is a subtle difference between evaluation at a point and substitution. This can be illustrated in the following example:

```
> diff( cos(f(x)) , x );
```

$$-\sin(f(x))\left(\frac{\partial}{\partial x}\,f(x)\right)$$

```
> subs( x=0 , % );
```

$$-\sin(f(0))\,\mathrm{diff}(f(0),0)$$

This looks weird; evaluating it yields an error:

```
> %;
Error, wrong number (or type) of parameters in function diff
```

The procedure `subs` substitutes immediately, but `eval` knows that `x` in `diff(cos(f(x)),x)` must be interpreted as a still unknown expression in `x`. As soon as this expression can be calculated, the actual substitution can be executed:

```
> eval( %% , x=0 );
```

$$-\sin(f(0))\left(\frac{\partial}{\partial x}\,f(x)\right)\Bigg|_{\{x=0\}}$$

```
> subs( f=exp , % ); %;
```

$$-\sin(1)\left(\frac{\partial}{\partial x}\,e^{x}\right)\Bigg|_{\{x=0\}}$$

B. The second method uses the procedure `D` that calculates the derivative of a function. For the example at the start of this section, where the derivative of the function $x \mapsto \exp(-x^2+x)$ at $x=1$ is calculated, we can use also:

```
> D( x->exp(-x^2+x) )(a);
```

$$(-2a+1)\,e^{(-a^2+a)}$$

The procedure `D` is discussed in section 6.9 on page 79.

4.3 Some more tools in differential calculus

For differentiation of **implicitly defined functions** you can use the procedure `implicitdiff`, which must be read from the library with `readlib(implicitdiff)` before it can be used. In releases before Release 4, you can use series approximations of solutions of equations; see section 16.9 on page 227.

There are procedures available for finding **extrema**: `extrema`, `minimize`, and `maximize`. These are not always reliable.

For calculations in differential geometry, Maple offers a package **difforms**; this can handle **differential forms** with wedge products, exterior derivatives, etc.

4.4 Antiderivatives

You can ask for the antiderivative (or primitive, indefinite integral) of an expression, e.g.,

$$\int \frac{x^2}{x^3+x^2+x+1}\,dx$$

with the procedure **integrate**, which can be abbreviated to **int**:

```
> x^2/(x^3+x^2+x+1);
```

$$\frac{x^2}{x^3+x^2+x+1}$$

```
> integrate( % , x ) ;
```

$$\frac{1}{2}\ln(x+1)+\frac{1}{4}\ln(x^2+1)-\frac{1}{2}\arctan(x)$$

```
> int( %% , x );
```

$$\frac{1}{2}\ln(x+1)+\frac{1}{4}\ln(x^2+1)-\frac{1}{2}\arctan(x)$$

The second argument to `int` should be a name not referring to something else. The second argument cannot be something like `x(t)`. This is analogous to the procedure `diff`; see section 4.1 on page 43.

If you want to see an antiderivative as a function and want to apply this function, use **eval** or `useintat`:

```
> int(f(x),x);
```

$$\int \mathrm{f}(x)\,dx$$

```
> eval(%,x=a);
```

$$\left(\int \mathrm{f}(x)\,dx\right)\Big|_{\{x=a\}}$$

Of course, this expression is not defined mathematically, only differences are defined in the case in which f is continuous; see section 11.6 on page 143. Essentially the same result can be found in one step by the use of the procedure **intat**, but more elegantly and more accessible for manipulation (see the on-line help for `intat` and `PDEtools[dchange]`).

4.5 Special elements appearing in the results of the procedure `int`

Maple has been equipped with a comprehensive set of mathematical functions, several of which are defined as an antiderivative. These can be used for expressing other antiderivatives. For instance,

```
> 'int(sqrt(1-k^2*sin(x)^2),x)';
```

$$\int \mathrm{sqrt}\left(1-k^2\,\sin(x)^2\right)\,dx$$

```
> %;
```

$$\frac{\sqrt{\cos(x)^2}\ \mathrm{EllipticE}(\sin(x),\mathrm{csgn}(k)\,k)}{\cos(x)}$$

The result contains a standard elliptic integral. The on-line help can tell you that `EllipticE(z,k)` is defined as

$$\int_0^z \frac{\sqrt{1-k^2\,t^2}}{\sqrt{1-t^2}}\,dt$$

The integration methods used by Maple in the previous cases are more or less the same as the usual methods applied in calculations done by hand. When Maple cannot find results with these, it launches the attack with a laborious but very powerful set of Risch algorithms. In order to see what methods are attempted, enter

```
> infolevel[int] := 1;
```

$$infolevel_{int} := 1$$

This **infolevel** can be set to 2, 3, 4, or 5 for more details.

Here is an example of a calculation based on a Risch algorithm with a rather cryptic result, which will be explained in the following.

```
> 'int(1/(x^6+x^3+1),x)';
int/indef1:   first-stage indefinite integration
int/ratpoly:   rational function integration
int/rischnorm:   enter Risch-Norman integrator
int/rischnorm:   exit Risch-Norman integrator
int/risch:   enter Risch integration
int/risch:   exit Risch integration
```

$$\int \frac{1}{x^6+x^3+1}\,dx$$

```
> % ;
```

$$\sum_{\_R=\%1} \_R \ln(x - 729\,\_R^4)$$

$$\%1 := \mathrm{RootOf}(19683\,\_Z^6 + 243\,\_Z^3 + 1)$$

Before analyzing this result, we reset the `infolevel` to prevent such additional information:

```
> infolevel[int] := 0;
```

$$infolevel_{int} := 0$$

The last result of the procedure `int` may look mysterious; it should be read as follows. The expression

```
RootOf(19683 _Z^6 + 243 _Z^3 + 1)
```

is the set of all six complex roots of the equation in $\_Z$:

$$19683\_Z^6 + 243\_Z^3 + 1 = 0$$

If these are denoted as $z_1 \ldots z_6$, then the antiderivative expression found by Maple can be written as

$$\sum_{i=1}^{6} z_i \ln(x - 729 z_i)$$

The appearance of this **RootOf** is caused by the action of `solve` within this Risch algorithm. These `RootOf` expressions are explained in Chapter 14, *Polynomial equations and factoring polynomials.*

Maple can make the result more concrete with the aid of **allvalues**. This procedure replaces a `RootOf` expression with all the roots of the argument of that `RootOf`. In the following, you can see how that can be done. Don't bother to understand the details before you have read Chapter 10, *Manipulating several objects at once.*

In the following we substitute the list of these roots for the `RootOf` expression (the second subexpression of the second subexpression of `result`), at the same time we convert the sum into a sequence and replace the internal name `_R` with `j`.

```
> subs(%1 = [allvalues(%1)] , sum=seq, _R=j, result );
```

$$\frac{1}{18}\,\%2^{(1/3)}\,\ln\left(x-\frac{1}{144}\,\%2^{(4/3)}\right),\left(-\frac{1}{36}\,\%2^{(1/3)}+\frac{1}{36}\,I\sqrt{3}\,\%2^{(1/3)}\right)\ln\left(x-729\left(-\frac{1}{36}\,\%2^{(1/3)}+\frac{1}{36}\,I\sqrt{3}\,\%2^{(1/3)}\right)^4\right),\left(-\frac{1}{36}\,\%2^{(1/3)}-\frac{1}{36}\,I\sqrt{3}\,\%2^{(1/3)}\right)\ln\left(x-729\left(-\frac{1}{36}\,\%2^{(1/3)}-\frac{1}{36}\,I\sqrt{3}\,\%2^{(1/3)}\right)^4\right),\frac{1}{18}\,\%1^{(1/3)}\,\ln\left(x-\frac{1}{144}\,\%1^{(4/3)}\right),\left(-\frac{1}{36}\,\%1^{(1/3)}-\frac{1}{36}\,I\sqrt{3}\,\%1^{(1/3)}\right)\ln\left(x-729\left(-\frac{1}{36}\,\%1^{(1/3)}-\frac{1}{36}\,I\sqrt{3}\,\%1^{(1/3)}\right)^4\right),\left(-\frac{1}{36}\,\%1^{(1/3)}+\frac{1}{36}\,I\sqrt{3}\,\%1^{(1/3)}\right)\ln\left(x-729\left(-\frac{1}{36}\,\%1^{(1/3)}+\frac{1}{36}\,I\sqrt{3}\,\%1^{(1/3)}\right)^4\right)$$

$$\%1 := -36-12\,I\sqrt{3}$$

$$\%2 := -36+12\,I\sqrt{3}$$

The result is an expression that would evaluate to a sequence of expressions, obtained by substituting six complex numbers for j in $j\ln(x-729j^4)$. This expression can be simplified with the tools of Chapter 15, *Manipulating algebraic expressions*, but we will not do so.

Now we can compute the result as the sum of these six expressions.

```
> sum(%[i] , i=1..nops([%])):
```

(Printing the result is suppressed by using the semicolon.)

4.6 Definite integrals

A definite integral can be requested as in the following example:

```
> int( x^3 , x = a .. b);
```

$$\frac{1}{4}b^4 - \frac{1}{4}a^4$$

Here `a .. b` is the Maple denotation of the **range** from a to b.

```
> 'int( x^(-2) , x=-1..1)';
```

$$\int_{-1}^{1} \frac{1}{x^2}\,dx$$

```
> %;
```

$$\infty$$

```
> 'int( exp(-x) , x=0..infinity)';
```

$$\int_{0}^{\infty} \mathrm{e}^{(-x)}\,dx$$

```
> %;
```

$$1$$

4.7 Helping Maple to find a definite integral by restricting the domain of a parameter

In the following example, Maple gets stuck in an undefined limit:

```
> 'int( exp(-a^2*x) , x=0..infinity)';
```

$$\int_{0}^{\infty} \mathrm{e}^{(-a^2 x)}\,dx$$

```
> %;
Definite integration: Can't determine if the integral is convergent.
Need to know the sign of --> a^2
Will now try indefinite integration and then take limits.
```

$$\lim_{x\to\infty} -\frac{\mathrm{e}^{(-a^2 x)} - 1}{a^2}$$

This is because Maple knows nothing about a and only supposes it to be a complex number. In a future release Maple may ask questions to the user in such a case, sometimes under the control of an option `ask`, but now we can take the initiative and tell that `a^2` represents a positive number. We can say so with **assume**. See section 3.4 on page 36 or section A.4 on page 278.

```
> assume(a^2, positive);
> int( exp(-a^2*x) , x=0..infinity);
```

$$\frac{1}{a\sim^2}$$

The tilde character ($\sim$) that follows the name a in the result means that a has properties, in this case, that a is assumed to be positive.

4.8 Helping Maple to find an antiderivative by conversion to RootOf

In many cases, Maple prefers `RootOf` expressions above the corresponding radical expressions. See section 14.4 on page 175. For instance,

```
> integrand:=sqrt(x^6+x^3+1)/x^4;
```

$$\frac{\mathrm{sqrt}(x^6+x^3+1)}{x^4}$$

```
> int(%,x) ;
```

$$-\frac{1}{3}\frac{\sqrt{x^6+x^3+1}}{x^3}+\int\frac{1}{2}\frac{2x^3+1}{x\sqrt{x^6+x^3+1}}\,dx$$

This does not yield a desired result. Conversion to `RootOf` helps Maple out of distress:

```
> convert( % , RootOf );
```

$$-\frac{1}{3}\frac{\mathrm{RootOf}(\_Z^2-x^6-x^3-1)}{x^3}+\int\frac{1}{2}\frac{2x^3+1}{x\ \mathrm{RootOf}(\_Z^2-x^6-x^3-1)}\,dx$$

The expression $\mathrm{RootOf}(\_Z^2-x^6-x^3-1)$ symbolizes a root of the equation in $\_Z$: $\_Z^2-x^6-x^3-1=0$, in fact $\pm\sqrt{x^6+x^3+1}$.
After the conversion no evaluation is performed, so we have to ask for evaluation by entering the converted form:

```
> % ;
```

$$-\frac{1}{3}\frac{\%1}{x^3}+\frac{1}{6}\ln\left(-\frac{-2+3x^3+8x^9+8\%1x^6-\ln 4\%1x^3+2\%1}{x^3}\right)$$

$$\%1:=\mathrm{RootOf}(\_Z^2-x^6-x^3-1)$$

The `RootOf`s in this result can be translated into radicals by

```
> convert( % , radical );
```

$$-\frac{1}{3}\frac{\%1}{x^3}+\frac{1}{6}\ln\left(-\frac{-2+3x^3+8x^9+8\%1x^6-\ln 4\%1x^3+2\%1}{x^3}\right)$$

$$\%1 := \sqrt{x^6 + x^3 + 1}$$

This result should be checked, especially here after the two conversions. That can be done in the standard way with `diff` and `normal`:

```
> normal(diff(%,x)-integrand);
```

$$0$$

4.9 Helping Maple to find an antiderivative by substitution

Sometimes a mathematician can see patterns not recognized by Maple. This can be used to help Maple. For instance:

```
> integrand := 1/(x^(1/2) + x^(1/3));
```

$$integrand := \frac{1}{\sqrt{x} + x^{(1/3)}}$$

```
> int(integrand,x);
```

$$-3\,x^{(1/3)} - 2\,\ln\left(x^{(1/3)} - 1\right) + \ln\left(x^{(2/3)} + x^{(1/3)} + 1\right) - \ln(x - 1) + 2\sqrt{x} - 2\,\mathrm{arctanh}\left(\sqrt{x}\right) + 6\,x^{(1/6)} + 2\,\ln\left(x^{(1/6)} - 1\right) - \ln\left(x^{(1/3)} + x^{(1/6)} + 1\right) - 2\,\ln\left(x^{(1/6)} + 1\right) + \ln\left(x^{(1/3)} - x^{(1/6)} + 1\right)$$

The result is not very attractive; if you try numerical evaluation you even find imaginary components ($-\pi i$). You can do better by guiding Maple as follows. First use the inert version of the integral, which does not try to calculate something:

```
> Int(integrand,x);
```

$$\int \frac{1}{\sqrt{x} + x^{(1/3)}}\,dx$$

Now change the variable as you would do it by hand. For this purpose, use **PDEtools[dchange]**.
(In releases before Release 5, use `student[changevar]`.)

```
> PDEtools[dchange]({x=t^6},%);
```

$$\int 6\,\frac{t^5}{\sqrt{t^6} + (t^6)^{(1/3)}}\,dt$$

In order to let Maple calculate this integral, use **value**:

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

$$\frac{1}{2}\left(t^6\right)^{(1/6)}\left(4\left(t^6\right)^{(1/3)}-6\left(t^6\right)^{(1/6)}+12\right)-6\ln\left(\left(t^6\right)^{(1/6)}+1\right)$$

Supposing that you want x to be positive, substitute $\sqrt[6]{x}$ for t:

```
> subs(t=x^(1/6),%);
```

$$\frac{1}{2}x^{(1/6)}\left(4x^{(1/3)}-6x^{(1/6)}+12\right)-6\ln\left(x^{(1/6)}+1\right)$$

```
> expand(%);
```

$$2\sqrt{x}-3x^{(1/3)}+6x^{(1/6)}-6\ln\left(x^{(1/6)}+1\right)$$

4.10 More tools for integration

In the package **inttrans** you can find procedures for calculation of several integral transforms: Laplace, Fourier (complex, real), Hilbert, Mellin, and Hankel, and their inverses. For numerical calculations, FFT is available.

Generally, integrals over lines, areas, surfaces, etc. must be converted to normal integrals, possibly using Maple for the manipulations, before Maple's integration can be used. The `student` package contains a procedure `Lineint` for calculating line integrals along parameterized curves in R^2, which must be used in combination with the procedure `value`. A quite different concept of line integral can be realized with `DEtools[line_int]`, which computes the solution of a total derivative.

4.11 Reliability of the calculation of antiderivatives

In practice, the antiderivatives produced with the procedure `int` turn out to be correct, apart from typical cases such as $\int x^n dx$, where Maple forgets the possibility that n might be 1.

However, there is a subtle question about continuity and constants. The procedure `int`, applied to a continuous function f, defined on an open, connected part U of the real or complex numbers, yields an antiderivative function of f that might contain *unnecessary singularities*: removable singularities, which sometimes can be removed by manipulations, and jumps, where the domain of the antiderivative is split into connected parts of U and the antiderivative in two adjacent parts can be made to join by adding different constant functions in the connected parts of the domain. Therefore, *evaluating an antiderivative at the boundaries and subtracting the results is not a reliable method for definite integration* and a good reason for the fact that *Maple does not supply an undetermined constant to antiderivatives.*

Getting unnecessary discontinuities in antiderivatives is a fundamental problem in today's state-of-the-art symbolic computing.

Although, generally, Maple is reliable in antiderivation, to be absolutely sure you can check the results of `int` by differentiating. Often, a check can be performed as in the following example, which you might have encountered already in section 11.9 on page 145:

```
> integrand:=1/(x^(1/2) + x^(1/3));
```

$$integrand := \frac{1}{\sqrt{x} + x^{(1/3)}}$$

```
> intresult:=int(%,x);
```

$$intresult := -3\,x^{(1/3)} - 2\,\ln\left(x^{(1/3)} - 1\right) + \ln\left(x^{(2/3)} + x^{(1/3)} + 1\right) - \ln(x-1) + 2\sqrt{x} - 2\,\operatorname{arctanh}\left(\sqrt{x}\right) + 6\,x^{(1/6)} + 2\,\ln\left(x^{(1/6)} - 1\right) - \ln\left(x^{(1/3)} + x^{(1/6)} + 1\right) - 2\,\ln\left(x^{(1/6)} + 1\right) + \ln\left(x^{(1/3)} - x^{(1/6)} + 1\right)$$

Maple does not see the pattern a mathematician would, but does find an antiderivative, which can be checked in the usual way:

```
> normal(diff(%,x)-integrand);
```

$$0$$

This proves that the result of `int` is a correct antiderivative. However, the result has discontinuities, although the integrand is continuous. If you look at the formula, you can see immediately that a discontinuity arises at $x = 1$. A calculation shows that the imaginary component is $-\pi$ for $x \in < 0, 1 >$ and π for $x > 1$. You can easily see the jump in 1 with Maple:

```
> assume(x,RealRange(Open(0),Open(1)));
> evalc(Im(intresult));
```

$$-\pi$$

```
> assume(x,RealRange(Open(1),2));
> evalc(Im(intresult));
```

$$\pi$$

You can use numerical and graphical tools as well, but these are not reliable when you are operating near a branch cut: if you are taking the logarithm of a negative number, you ought to find π as the imaginary component. But if a slight numer-

ical deviation causes a logarithm to be taken of a number with a small negative imaginary component, you get approximatily $-\pi$.

The discontinuity at 1 of the antiderivative can be removed easily; see section 15.8 on page 201.

The standard method of testing the correctness of an antiderivative is:

- differentiate the found antiderivative
- subtract this from the original expression and
- apply **normal** to this difference and check if this is 0.

Eventually, some extra manipulation may be necessary. See Chapter 15, *Manipulating algebraic expressions.*

4.12 Definite integrals of discontinuous functions

In calculations of definite integrals, Maple looks for the presence of discontinuities and asymptotes in the antiderivative and uses the results for the calculation. For instance:

```
> integrate( 1/x^2 , x=-1..1 );
```

$$\infty$$

The test of continuity of an antiderivative between the boundaries of a definite integral can be discarded by adding an option `continuous` to the procedure `int`.

However, if the boundary values of the interval are abstract objects, Maple cannot test if there are discontinuities between them. In these cases, a result is given without any restrictions:

```
> integrate( 1/x^2 , x=p..q );
```

$$-\frac{-q+p}{qp}$$

If this result is evaluated at p=-1 and q=1, the result does not equal $\int_{-1}^{1} \frac{1}{x^2}$.

```
> eval(%,[p=-1,q=1]);
```

$$-2$$

4.13 Definite integrals and branch cuts of functions

Here is another rare case, where Maple gets wrong results in calculating a definite integral:

```
> (-2*sin(x)+I*cos(x))/(2*cos(x)+I*sin(x));
```

$$\frac{-2\sin(x) + I\cos(x)}{2\cos(x) + I\sin(x)}$$

```
> int( % , x=0..2*Pi );
```

$$0$$

This result is not correct. It can be calculated easily by hand, yielding 2π. In fact, it is a contour integral for the complex function $z \mapsto \frac{1}{z}$, yielding $2\pi i$ times its residue 1 in 0. (A procedure `residue` is available.)

Let's calculate the corresponding antiderivative:

```
> int( %% , x );
```

$$\ln(2\cos(x) + I\sin(x))$$

This antiderivative is correct, which can be checked easily. However, if we ask for the integral from 0 to 2π, the antiderivative crosses the branch cut of `ln` at the negative part of the real axis, and this causes the wrong result.

It is not difficult for Maple to approximate this integral by numerical integration:

```
> evalf( Int(%%%,x=0..2*Pi) );
```

$$-.6395448337\,10^{-13} + 6.283185307\,I$$

```
> fnormal(%);
```

$$6.283185307\,I$$

That is a good approximation to $2\pi i$. Using numerical integration, possibly after having chosen some values for parameters, is a recommended testing method, although you should be aware of the risks of deviations, for instance with branch cuts.

4.14 Reliability of calculations of definite integrals

In finding definite integrals, the present release 5 of Maple V seems to be rather reliable, apart from rare problems concerning discontinuities and branch cuts, shown in the previous two sections. But it may be wise to test results, especially in complicated cases with discontinuities in the integrand or with nonreal ranges. In search of a definite integral, Maple generally works as follows:

- Try to find an antiderivative.

- Then test this for continuity in the given range.
- If no discontinuity is found in the given range, try to calculate the definite integral by subtracting the right limit of the right boundary from the left limit of the left boundary of this antiderivative. In case of discontinuities, try to use these by calculating limits.

In some cases, it can be complicated to test results in an exact way, but often it is easy to test results in a numerical way by numerical integration and/or plotting.

What can be done to obtain correct results, generally?

- First have a close look at patterns and properties of expressions in input and output that can predict something about the result.
- Look for discontinuities.
- Try to check the antiderivative by differentiating and comparing the result with the original expression.
- If possible, test numerically with `evalf(Int( , ))`, possibly with options: `_CCquad` and/or `_Dexp`. The extra options serve to prevent Maple from using symbolic methods near singularities.

4.15 Numerical integration

The following definite integral cannot be calculated exactly.

```
> int( exp(x-x^3) , x=0..1 );
```

$$\int_0^1 \mathrm{e}^{(x-x^3)}\,dx$$

A numerical approximation can be found by applying **evalf** to this result, with the number of digits as the second argument.

```
> evalf( % , 30 );
```

$$1.29264345165894609581636207792$$

In the following case, Maple could yield an exact result for $\int_0^1 \sqrt{x^3 - x}\,dx$, expressed in a lengthy and complicated expression containing Legendre functions. This could then be approximated with `evalf`. There is a faster and simpler way to achieve an approximation:

```
> Int( sqrt(x-x^3) , x = 0 .. 1 );
```

$$\int_0^1 \sqrt{x - x^3}\, dx$$

The procedure **Int** is the *inert* version of `int`: it does not execute any calculations, but if we ask for an approximation with `evalf`, numerical integration methods are used:

```
> evalf( % , 20 );
```

.47925609389423688298

In cases where the boundaries are definite real numbers and the integrand does not contain undetermined variables, you can ask Maple to calculate a numerical approximation to an integral, using `Int` and `evalf` as in the previous example. If the boundary numbers are complex, you must choose an integration path and convert the integral to an integral over a part of the real numbers, possibly using substitution. See section 11.9 on page 145.

In numerical integration, Maple tries to find a result where *all digits apart from the last one are significant.* It is possible to choose between several numerical integration techniques by adding an option `_CCquad`, `_Dexp` or `_NCrule`. Specific information on these options can be obtained with the on-line help for `int,numeric`.

You can speed up numeric integration by using `optimize`. See section 9.2 on page 119.

If Maple detects a discontinuity in between the boundaries of the integration, numerical integration generates an error: `(in evalf/int) unable to handle singularity`.

There is no direct facility to generate a table or interpolating function for a **numerical approximation to an antiderivative** on an interval. In order to obtain a graph of an antiderivative function, you can think of integration as solving a differential equation and use graphical facilities for this field. See section 17.5 on page 238.

4.16 Numerical approximations to multiple integrals

Here is an example of a double integral. Maple does not find a closed form but an intermediate result:

```
> 'int( int(exp(t^3),t=0..x) , x=0..1 )';
```

$$\int_0^1 \int_0^x e^{(t^3)}\, dt\, dx$$

```
> %;
```

$$\int_0^1 -\frac{1}{9}\,\frac{x\left(2\pi\sqrt{3}-3\,\Gamma\left(\frac{1}{3},-x^3\right)\Gamma\left(\frac{2}{3}\right)\right)}{\Gamma\left(\frac{2}{3}\right)\left(x^3\right)^{(1/3)}}\,dx$$

The procedure `evalf` can handle a multiple integral; however, be careful, **do not apply `evalf` as follows**:

```
> evalf( int( int(exp(t^3),t=0..x) , x=0..1 ) );
```

$$-.2803536934 - .4855868411\,I$$

Obviously the result is not correct: the imaginary component makes no sense. In fact, the previous result of the symbolic calculation of the double integral is incorrect. By applying `evalf` on `int(int(...))` we have caused Maple first to evaluate this by symbolical calculations and using that incorrect result. So it is wise to avoid such a calculation by the aid of the inert version of the integral:

```
> Int( Int(exp(t^3),t=0..x) , x=0..1 );
```

$$\int_0^1 \int_0^x \mathrm{e}^{(t^3)}\,dt\,dx$$

```
> evalf(%);
```

$$.5607073869$$

Such calculations are rather time-consuming. Taking a series development of the inner integral (to sufficient high order) and integrating this from 0 to 1 is much more efficient. See Chapter 8, *Taylor or Laurent expansion and limits*. Already for the present, rather undemanding case, the method below is about 20 times faster.

```
> series( exp(t^3) , t=0 , 30 );
```

$$1+t^3+\frac{1}{2}t^6+\frac{1}{6}t^9+\frac{1}{24}t^{12}+\frac{1}{120}t^{15}+\frac{1}{720}t^{18}+\frac{1}{5040}t^{21}+\frac{1}{40320}t^{24}+\frac{1}{362880}t^{27}+O(t^{30})$$

```
> convert( % , polynom );
```

$$1+t^3+\frac{1}{2}t^6+\frac{1}{6}t^9+\frac{1}{24}t^{12}+\frac{1}{120}t^{15}+\frac{1}{720}t^{18}+\frac{1}{5040}t^{21}+\frac{1}{40320}t^{24}+\frac{1}{362880}t^{27}$$

```
> int( int( % , t=0..x ) , x=0..1 );
```

$$\frac{4387950745386281}{7825740931008000}$$

```
> evalf( % );
```

$$.5607073866$$

From a rough estimate of the error with the Taylor remainder theorem, this result should be accurate to at least three digits. According to this method of estimating

the error, a 10-digit accurate result would require series expansion to order 60. This last task can be executed by Maple in no time, but a little experimenting with the order shows that order 30 is sufficient for 10-digit accuracy in this case.

4.17 Definite and indefinite sums and products

The procedure **sum** is the discrete variant of `int`:

```
> 'sum( 3^(-k) , k=1..10 )'; %;
```

$$\sum_{k=1}^{10} 3^{(-k)}$$

$$\frac{29524}{59049}$$

```
> 'sum( 3^(-k) , k=1..N )'; %;
```

$$\sum_{k=1}^{N} 3^{(-k)}$$

$$-\frac{3}{2}\left(\frac{1}{3}\right)^{(N+1)}+\frac{1}{2}$$

```
> 'sum( 3^(-k) , k )'; %;
```

$$\sum_{k} 3^{(-k)}$$

$$-\frac{3}{2}\left(\frac{1}{3}\right)^{k}$$

In the first and second example, the **range** is indicated with .. (two dots).
In the last case, we see the discrete sister of the antiderivative; discrete differentiation yields the original expression:

```
> eval(%,k=k+1) - % ;
```

$$-\frac{3}{2}\left(\frac{1}{3}\right)^{(k+1)}+\frac{3}{2}\left(\frac{1}{3}\right)^{k}$$

```
> expand( % );
```

$$\left(\frac{1}{3}\right)^{k}$$

Summing to infinity:

```
> 'sum( 3^(-k) , k=0..infinity )'; %;
```

$$\sum_{k=0}^{\infty} 3^{(-k)}$$

$$\frac{3}{2}$$

Keep in mind that the arguments of sum are evaluated before the procedure comes into action. So the "walking parameter" should not refer to something else:

```
> n := 1000;
```

$$n := 1000$$

```
> sum( 2^n, n=1..10 );
Error, summation variable previously assigned
               second argument evaluates to
1000 = 1 .. 10
```

You can use forward quotes to prevent early evaluation of the arguments in such cases:

```
> sum( '2^n' , 'n'=1..10 );
```

$$2046$$

Sums of less than 1,000 terms are calculated simply by calculating each term and adding them, but if there are more terms, Maple tries to find a closed form.

```
> sum(1/k,k=1..1000);
```

$$\Psi(1001) + \gamma$$

$\Psi(x)$ is defined as $\frac{d}{dx}\ln(\Gamma(x))$ and γ is the Euler constant.

This rule depends on the release, but it cannot be changed by the user unless the procedure sum is changed ("... `elif dab < 1000 then` ..."). However, there is a trick; for instance,

```
> sum(cos(x),x=0..7);
```

$$1 + \cos(1) + \cos(2) + \cos(3) + \cos(4) + \cos(5) + \cos(6) + \cos(7)$$

You can urge Maple to search for a formula by presenting the right boundary of the index as an indefinite, and then you can evaluate at $N = 7$:

```
> sum( cos(x) , x = 0..N );
```

$$-\frac{1}{2}\cos(N+1) - \frac{1}{2}\frac{\sin(1)\sin(N+1)}{\cos(1)-1} + \frac{1}{2}$$

```
> eval( % , N=7 );
```

$$-\frac{1}{2}\cos(8) - \frac{1}{2}\frac{\sin(1)\sin(8)}{\cos(1)-1} + \frac{1}{2}$$

The section 4.5 on page 47 shows a construction that yields sums over the set of roots of a polynomial, using the `RootOf` denotation. The sum of the elements of a set or list (see section 10.8 on page 133) can be found as follows:

```
> a_list := [ 2 , 3 , 5 ];
```

$$a\_list := [2, 3, 5]$$

```
> convert( a_list , `+` );
```

$$10$$

An inert version of `sum` is available: **Sum**. This does no calculations, but can be handled with manipulation procedures. It can be activated with `value`.

```
> Sum( x^2 , x=1..N );
```

$$\sum_{x=1}^{N} x^2$$

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

$$\frac{1}{3}(N+1)^3 - \frac{1}{2}(N+1)^2 + \frac{1}{6}N + \frac{1}{6}$$

In connection to summation, Maple offers the **Z-transform** with `ztrans`; several other tools can be found in the package `sumtools`. Moreover, you can use `PDEtools[dchange]` in the same way as in applying the substitution rule for integrals. (In releases before Release 5, use `student[changevar]`.)

Calculation of products is analogous to calculation of sums; use the procedure **product**:

```
> product( x^k , k=1..100 );
```

$$x^{5050}$$

4.18 Other tools and pedagogical facilities

Many more Maple tools are available for calculus. If you are looking for something special, don't forget to look into the share library; see section 5.6 on page 69.

Mathematical education can profit considerably from using Maple. For teaching calculus, a special package, `student`, is available. Here is an example: integration by parts. Maple can correctly compute the following antiderivative without the manual use of this technique. In fact, we must keep Maple from calculating an antiderivative by using the **inert form** of `int`: **Int**.

```
> Int( x*cos(x) , x );
```

$$\int x \cos(x)\, dx$$

```
> student[intparts]( % , x );
```

$$x \sin(x) - \int \sin(x)\, dx$$

The application of the procedure **value** makes Maple calculate the integral by converting the inert `Int` into the active `int`.

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

$$x \sin(x) + \cos(x)$$

It is advisable not to use the computer early. First, a student should become familiar with the concepts involved, and execute sufficiently calculations for a specific type of mathematical action by hand and mind, possibly only with the most basic methods, until she/he really understands what is happening and the goal of it. Then, she/he can learn to execute the same calculations with one or more Maple commands and learn to read the output generated by Maple. If a student has not done so, he/she might use the facilities by trial and error, not knowing what to do. In subsequently solving problems where this action is one of the steps, the student can concentrate on choosing which step to take, not being distracted from this task by the burden of the detailed calculation within each step. In this way, using Maple can enhance abstraction processes in learning and heuristics in problem solving.

chapter 5

Names and evaluation 2: applying procedures

This chapter explains the use of Maple procedures, arguments and output. Ways of using Maple's library of procedures are also explained. This chapter is a sequel to Chapter 3, Names and evaluation 1: mathematical variables.

5.1 Evaluation of names in arguments of procedures

The following rule is very important for using Maple:

> The arguments given to a procedure are evaluated before the procedure comes into action, excepting a few special procedures.

The exceptional procedures can be found at the end of section B.6 on page 289.

For example:

```
> x := 484^(1/2);
```

$$x := \sqrt{484}$$

```
> simplify( x );
```

$$22$$

After the second command, first `x` is evaluated, yielding $\sqrt{484}$. This is given to the procedure `simplify`, which yields 22 as the result. This action does not change the value of `x`:

```
> x ;
```

$$\sqrt{484}$$

Obviously, there is no possibility for a procedure such as `simplify` to change the reference of its argument, as the evaluation of it takes place before `simplify` comes into action.

Here is an example where problems arise because arguments are evaluated first:

```
> diff( sin(x) , x );
```

```
Error, wrong number (or type) of parameters in function diff
```

The last command seems to ask Maple to differentiate $\sin(x)$ with respect to x, but `x` has been assigned the value $\sqrt{484}$. Before `diff` came into action, Maple evaluated `x` to $\sqrt{484}$. Therefore, we asked Maple to differentiate $\sin(\sqrt{484})$ with respect to $\sqrt{484}$ and Maple cannot make sense of this strange command. In order to find out what is happening in such a case, evaluate the arguments:

```
> sin(x),x;
```

$$\sin\left(\sqrt{484}\right), \sqrt{484}$$

General advice:

> If a procedure is behaving in an odd manner,
> first test the values of the names contained in its arguments.

These troubles would have been prevented if we had unassigned the name `x` immediately, when its value became obsolete.

```
> x := 'x';
```

$$x := x$$

But people often forget to unassign names.

5.2 Options of procedures

For each procedure there is a minimum number of arguments, but many procedures can use more arguments. Some of these arguments can be options, deciding which type of action is asked for. For instance, the procedure `fsolve` tries to find numerical solutions to an equation or set of equations:

```
> fsolve(sin(x)=1/3,x);
```

$$.3398369095$$

Generally, `fsolve` tries to find real solutions and is content with just one solution; see section 16.7 on page 225.

```
> fsolve(sin(x)=2,x);
```

$$\text{fsolve}(\sin(x) = 2, x)$$

There are no real solutions here, but you can ask for a complex solution by adding the option `complex`:

```
> fsolve(sin(x)=2,x,complex);
```

$$1.570796327 - 1.316957897\,I$$

When a procedure fails to do what you want, always read the on-line help for this procedure. In many cases, adding or changing an option can help.

5.3 Output and results of procedures

Generally, execution of a procedure yields a Maple object as a result. There are three cases where no result is generated:

a. If the execution generates an error, the error message is printed to the screen and no result is produced.

b. A few procedures never yield a result.

   For instance, the procedures **print** and **lprint** print their arguments to the screen without pushing a result onto the ditto stack:

```
> ( (5*8*x^2-1) / x^3 );
```

$$\frac{40\,x^2 - 1}{x^3}$$

```
> lprint(expand(%^2));
1600/x^2-80/x^4+1/x^6
> % ;
```

$$\frac{40\,x^2 - 1}{x^3}$$

   Because `lprint` yields no result, it cannot leave a result on the stack of previous results, so the last ditto yields $\frac{40\,x^2-1}{x^3}$ as the last result.

c. Some procedures occasionally yield no result, for instance, `solve` in the case where no solution for the equation is found.

The result of a procedure can be used as input for another procedure by nesting them:

```
> solve( diff(x^3-x,x) , x );
```

$$\frac{1}{3}\sqrt{3},\ -\frac{1}{3}\sqrt{3}$$

The procedure `diff` has calculated an expression, and this result has been used by `solve`. The result of `solve` has been pushed onto the stack for quoting by the **ditto** (`%`).

However, you are well advised to *avoid nesting*; better look at the result of each step and each time use the ditto for reference to the previous result. It is easy, and by tracking results often you can prevent mistakes and unwanted results.

5.4 Assigning side results to arguments of procedures

In section 2.10 on page 29, the procedure **iquo** was introduced with a special construction:

```
> iquo( 25 , 7 , 't' );
```

$$3$$

```
> t;
```

$$4$$

The integer division of 25 by 7 yields 3 with remainder 4; this remainder is assigned to `t` by `iquo`. If `iquo` gets a third argument, Maple tries to assign the remainder of the division to that argument. This is only possible if the argument is a name.

The **forward quotes** around `t` in the third argument to `iquo` make sure that `t` cannot be evaluated. Let's see what happens without these forward quotes:

```
> iquo(45,13,t);
Error, wrong number (or type) of parameters in function iquo
```

Maple has tried to assign the remainder 6 to the third argument. But the third argument has been evaluated to 4 before execution of the procedure started, so Maple would have to assign 6 to 4 and refuses to do so, reporting that there is something wrong with the parameters.
This premature evaluation of `t` can be prevented with **forward quotes**:

```
> iquo(45,13,'t');
```

$$3$$

```
> t;
```

$$6$$

Obviously these forward quotes are not necessary if the third argument of `iquo` is an unassigned name, as was the case in the first example of `iquo`. However, in order to avoid problems, it is good practice to use forward quotes for the third argument of `iquo` and generally for arguments of this kind.

5.5 Names referring to procedures

Procedures are common objects in Maple. For instance, "`normal`", "`abs`", etc. are names referring to Maple procedures. Simply typing "`abs`" does not reveal the fact that `abs` is a procedure:

```
> abs;
```

$$abs$$

Names of procedures must be dealt with in a special way; they are evaluated only on special request:

```
> eval(abs);
```

$$\textbf{proc}()\ \textbf{option}\ \textit{builtin};64\ \textbf{end}$$

The result states that the name "abs" is referring to a procedure contained in the Maple kernel and supplies the internal number.

Names of Maple V procedures are protected against assignments from Release 3:

```
> abs := sqrt(7) ;
Error, attempting to assign to `abs` which is protected
```

However, this protection can be made idle by the procedure unprotect:

```
> unprotect(abs):
```

(In earlier releases this is done by unassigning the name.) Now the name abs can take any value:

```
> abs:=sqrt(7);
```

$$abs := \sqrt{7}$$

The name "abs" refers to $\sqrt{7}$ and the procedure abs has been lost, although only for the current Maple session.

5.6 The Maple library of procedures

The Maple system contains an enormous number of procedures available for the user. In Maple V release 5 more than 600 procedures can be used directly as we have done up to now. A list of these procedures is shown by the command ?index,function. Only a few of these (the "interior" procedures) are present in the computer's working memory at start-up. Most other procedures are automatically read from file into the memory when invoked the first time. Some other procedures must be loaded explicitly by the user before they can be used with the aid of the procedure **readlib**. For instance, the procedure psqrt can be used for finding the square root of a polynomial, if it exists:

```
> expand( (3*x^3*y^4 - 2*x*y^6 + 12*x^5*y^2)^2 );
```

$$-39\,x^6\,y^8 - 12\,x^4\,y^{10} + 72\,x^8\,y^6 + 4\,x^2\,y^{12} + 144\,x^{10}\,y^4$$

```
> psqrt( % );
```

$$\mathrm{psqrt}\left(-39\,x^6\,y^8 - 12\,x^4\,y^{10} + 72\,x^8\,y^6 + 4\,x^2\,y^{12} + 144\,x^{10}\,y^4\right)$$

The procedure has done nothing because it is not known to the present Maple session and is not loaded automatically. We have to read it from the library:

```
> readlib(psqrt);
```

$$\textbf{proc}(p) \ \ldots \ \textbf{end}$$

From now on, the procedure `psqrt` can be used:

```
> psqrt("""");
```

$$-12\,x^5\,y^2 - 3\,x^3\,y^4 + 2\,x\,y^6$$

For each procedure that is not loaded automatically, the on-line help for that procedure says so.

Many other procedures are contained in packages for special fields. For a list of these packages; see `index,packages`. For instance, there is a package `orthopoly` for orthogonal polynomials. For a list of all the procedures in this package; see `?orthopoly`. One of them is the procedure `P`, which can calculate Legendre polynomials. You can call it as `orthopoly[P]`:

```
> orthopoly[P](4,x);
```

$$\frac{35}{8}\,x^4 - \frac{15}{4}\,x^2 + \frac{3}{8}$$

However, such an indexed call to a procedure in a package can only be used for some packages. For *all* packages it is possible to load a procedure using **with**:

```
> with(orthopoly,P);
```

$$[P]$$

From now on, we can call this procedure directly:

```
> P(5,X);
```

$$\frac{63}{8}\,X^5 - \frac{35}{4}\,X^3 + \frac{15}{8}\,X$$

If you want all the procedures of a package to be loaded at once, you can also use `with`:

```
> with(orthopoly);
```

$$[G, H, L, P, T, U]$$

Now we can use the other procedures from the package as well.

A package may contain subpackages. This is the case with the `stats` package for statistics. How such a package can be handled is shown in the on-line help for these packages.

Moreover, contributions of Maple users are gathered in the **share library**. The way this can be used is explained if one types `?share`:

a. enter `with(share);`

b. search your tool by choosing your subject from `?share,contents` and reading the contents of ?share,<subject>;

c. pick up the wanted tools according to their description, generally using `readshare`.

5.7 Asking procedures for additional information with `infolevel`

Several procedures print additional information to the screen about their activities if the `infolevel` of this procedure is set higher than 0 (maximal 5).

```
> infolevel[simplify]:=1:
> exp(sin(a)^2+cos(a)^2);
```

$$e^{(\sin(a)^2+\cos(a)^2)}$$

```
> simplify( % );
simplify:   applying   trig   function to expression
simplify:   applying   power   function to expression
simplify:   applying   exp   function to expression
simplify:   applying  commonpow  function to expression
simplify:   applying  power  function to expression
```

$$e$$

If you have already executed the command before raising the corresponding infolevel, and then execute the command again, you may not get this additional information:

```
> simplify( %% );
```

$$e$$

The remedy is applying `forget` to the procedure. An explanation of this is given in Appendix D, *Procedures remembering previous results.*

```
> readlib(forget)(simplify);
> simplify( %%% );
simplify:   applying   trig   function to expression
simplify:   applying   power   function to expression
simplify:   applying   exp   function to expression
simplify:   applying  commonpow  function to expression
simplify:   applying  power  function to expression
```

$$e$$

5.8 Printing standard procedures from Maple's library

If you want to see how a procedure is programmed in the Maple language, you must enable printing of these procedures by the command

```
> interface(verboseproc=2);
```

Now you can enter commands such as `print(exp)` and you will see how procedures are programmed. This book does not discuss programming, but a short introduction can be found in Appendix E, *Control structures*.

chapter 6

Creating and using mathematical functions

The first part of this chapter discusses some aspects of the many mathematical functions contained in Maple.

Any expression containing "indeterminate variables" can be seen as a function in these variables; function values can be found by substituting values for the variables. In practice, this can be rather clumsy. It is often more efficient to create and use your own functions. In the second part of the present chapter this subject is discussed.

Moreover, calculation of derivative functions, creating functions from existing functions, especially by composition, and piecewise-defined functions are discussed.

Creating your own functions is a start in programming. A full guide to programming is beyond the scope of this book. A basic step in this direction can be found in Appendix E, Control structures.

6.1 Standard mathematical functions

Maple knows a considerable number of mathematical functions, such as `sin`, `exp`, `abs`, `GAMMA`, `Heaviside`, etc. Some of the mathematical functions in Maple are less common, like those used specifically for expressing antiderivatives. A concise survey of the functions available at start-up of Maple is accessible with `?inifcns`.

Several Maple packages for special mathematical fields yield additional mathematical functions, especially `orthopoly`, which can generate orthogonal polynomials of several types: **Chebyshev**, **Gegenbauer**, **Hermite**, **Jacobi**, **Laguerre**, and **Legendre** polynomials.

These mathematical functions are implemented in Maple as procedures in the same way as other procedures such as `expand`, `diff`, etc. Applying a mathematical function to an argument brings the *simplification rules* programmed in this procedure into action:

```
> sin(100/3*Pi);
```

$$-\frac{1}{2}\sqrt{3}$$

(Remember that the mathematical constant π is to be entered as `Pi`, not as `pi`.)

If a function cannot calculate an exact function value for some argument, it returns an **unevaluated function call**:

```
> sin(1);
```

$$\sin(1)$$

Generally, an approximation can be asked for by `evalf`:

```
> evalf(%,30);
```

$$.841470984807896506652502321630$$

If the argument is a floating-point number, Maple approximates automatically, with accuracy determined by the value of `Digits`. See section 2.8 on page 26 and section 2.9 on page 28.

```
> sin( 1. );
```

$$.8414709848$$

Logarithms with a base different from e are available, but these are automatically converted into quotients of natural logarithms:

```
> log[a](b);
```

$$\frac{\ln(b)}{\ln(a)}$$

6.2 Definitions of inverse functions, branch cuts

For most mathematical functions that are defined as inverses of other functions, branches must be chosen. For instance, for the square root, being an inverse of the square function, $\sqrt{9}$ is defined as 3 and not as -3, and $\sqrt{i}$ is defined as $1/2\sqrt{2}+1/2\sqrt{2}i$. Likewise, $\arccos(-1/2)$ yields $\frac{2}{3}\pi$, although there are infinite many solutions for the equation $\cos(x) = -\frac{1}{2}$.

The function **ln** has been defined on the complex numbers (except for 0) as a right inverse of `exp`, where `ln(exp(r+I*`ϕ`))` yields `r+I*`ϕ if $-\pi < \phi \leq \pi$.

In many cases, the choice of branches is less evident than in the previous cases, especially if such a function is to be applied to complex numbers. These choices are fixed in Maple and cannot be changed without reprogramming the function. In order to find the definitions of the **inverse trigonometric functions** and the **inverse hyperbolic functions**, use `convert(...,ln)`:

```
> convert( arcsin(x) , ln );
```

$$-I\ln\left(\sqrt{1-x^2}+I\,x\right)$$

6.3 Denotation of the functions exp, Gamma, and Zeta

The function exp is printed in windowing versions of Maple as follows:

```
> exp(x);
```

$$\mathbf{e}^x$$

Do *not* enter this as

```
> e^x ;
```

$$e^x$$

In the last case, you see only the undefined variable e to the power x. In releases from V.4 you can see a difference in the e characters: the bold **e** denotes the base of the natural logarithm, while the slanted e is a variable.
If you like to use e as the base of the natural logarithm, you can enter:

```
> alias(e=exp(1));
```

$$I,\ e$$

(See section B.1 on page 285.)

Other possible causes of confusion are the functions GAMMA and Zeta. The name of the function GAMMA is printed in the same way as the name Gamma in windowing versions of Maple, and the name of the function Zeta is printed as a lower-case Greek letter:

```
> Gamma, GAMMA, zeta, Zeta;
```

$$\Gamma,\ \Gamma,\ \zeta,\ \zeta$$

```
> Gamma(5), GAMMA(5), zeta(0), Zeta(0);
```

$$\Gamma(5),\ 24,\ \zeta(0),\ \frac{-1}{2}$$

The differences can be made visible with **lprint**:

```
> lprint(%%):
Gamma   GAMMA   zeta   Zeta
```

The name Chi is not interpreted as a Greek letter, as it is the name of the hyperbolic cosine integral.

In text-only versions of Maple (such as for MS-DOS), this confusion does not exist, because exp(x) is printed as exp(x), and Greek letters are not printed on the screen.

6.4 Expressions versus functions, creating functions

In Maple, functional relations can be described in two ways:

A. By an expression. For instance, if temperature is descending with time exponentially, this can be embodied in:

```
> Texpr := T0 * exp(-a*t);
```

$$Texpr := T0\,\mathbf{e}^{(-a\,t)}$$

The temperature at t=5 can be calculated by evaluation at a value (or substitution):

```
> eval( Texpr , t=5 );
```

$$T0\,\mathbf{e}^{(-5\,a)}$$

B. By a function. For instance, in the previous example the temperature can be described as a function of the time:

```
> t -> T0 * exp(-a*t);
```

$$t \rightarrow T0\,\mathbf{e}^{(-a\,t)}$$

The previous command is entered with an **arrow** by using a combination of the minus and the greater than sign. In this way we obtain a Maple function, which we can assign to a name:

```
> Tfunc := %:
```

Now the temperature at t=5 can be calculated by *applying the function*:

```
> Tfunc(5);
```

$$T0\,\mathbf{e}^{(-5\,a)}$$

It is important that a user should be well aware of the distinction between an expression and a function. For example, in succession to the previous example, a user could try to find the moment when temperature descends to 100, in the following erroneous ways:

```
> solve( Tfunc=100 );
```

$$100$$

Here we have not entered the variable with respect to which the equation should be solved, so Maple supposes that this variable is `Tfunc` and easily finds the solution `100`. If the variable `t` is given to `solve`, we get no result:

```
> solve( Tfunc=100 , t );
```

The last command asks Maple to find values of `t` for which the algebraic expression `Tfunc` equals `100`, and no solutions are found, so no result is printed on the screen. A correct command is:

```
> solve( Tfunc(t)=100 , t);
```

$$-\frac{\ln\left(100\,\frac{1}{T0}\right)}{a}$$

(You might have thought that Maple has interpreted the command `solve (Tfunc=100,t);` as the odd question: "For which values of `t` does the function `Tfunc` equal the constant function $t \mapsto 100$?" This is not the case, as Maple does not evaluate the name `Tfunc`.)

6.5 Creating functions in several arguments

Functions in more than one argument can be created in the same way as functions in one argument. For instance:

```
> saddle:= (x,y) -> 5*x^2 - 3*y^2;
```

$$saddle := x, y \rightarrow 5\,x^2 - 3\,y^2$$

```
> saddle(4,1);
```

$$77$$

The *parentheses* in the definition are essential: if those parentheses had been omitted in the previous example by writing

```
x,y -> 5*x^2 - 3*y^2;
```

$$x,\ y \rightarrow 5\,x^2 - 3\,y^2$$

Maple would have interpreted this as the sequence of the name `x` and the function `y -> 5*x^2 - 3*y^2`.

Functions to vector spaces are discussed in section 18.19 on page 271.

6.6 A pitfall in creating mathematical functions

You might think that the following command could create a mathematical function:

```
> w(x) := 5*p^x;
```

$$\mathrm{w}(x) := 5\,p^x$$

Maple accepts this command and it seems that it does the job:

```
> w(x);
```

$$5\,p^x$$

But it does not:

```
> w(3);
```

$$\mathrm{w}(3)$$

In mathematics, you would say: *for each* x *in* $\mathbf{R}$, $f(x) := 5p^x$. Often, people are less precise in formulating, but Maple takes the definition entered above literally: we have created a function that renders the value $5\,p^x$ if its argument is the name `x`, but to all other arguments it yields an unevaluated function call. The correct function can be created by:

```
> f := x -> 5*p^x ;
```

$$f := x \to 5\,p^x$$

or with the aid of the procedure `unapply`, discussed in the next section or with the aid of `codegen[makeproc]`.

6.7 Using existing expressions for creating mathematical functions

An important rule for the arrow function construction is:

> In the creation of a function with the arrow,
> the expression to the right of the arrow is not evaluated.

It is often convenient to use an existing expression for the creation of a function. Here is such an expression, created by calculating an antiderivative expression (integral) of an expression:

```
> s5:= int( sin(x)^5 , x );
```

$$s5 := -\frac{1}{5}\sin(x)^4\cos(x) - \frac{4}{15}\sin(x)^2\cos(x) - \frac{8}{15}\cos(x)$$

If we want to find the **antiderivative function** or **primitive function** from this expression, we can do so by using the arrow and typing this expression to the right of the arrow, but it can be done in a much easier way with the procedure **unapply**:

```
> prim := unapply( s5 , x );
```

$$prim := x \to -\frac{1}{5}\sin(x)^4\cos(x) - \frac{4}{15}\sin(x)^2\cos(x) - \frac{8}{15}\cos(x)$$

First, the arguments of `unapply` are evaluated; then a function is created. Let's test this function:

```
> prim(Pi/2), prim(0);
```

$$0, \frac{-8}{15}$$

You might think this function could be defined by:

```
> missprim := x -> s5;
```

$$missprim := x \rightarrow s5$$

You see the effects of the special evaluation rule for the arrow construction: `s5` is not evaluated, so the parameter x of the function has no relation to the x in the expression that `s5` refers to:

```
> missprim(0);
```

$$-\frac{1}{5}\sin(x)^4\cos(x) - \frac{4}{15}\sin(x)^2\cos(x) - \frac{8}{15}\cos(x)$$

The function `missprim` yields `s5`, no matter what arguments are given to it.

Functions in more than one argument can be created with `unapply` as well:

```
> s := int( sin(x)^n , x );
```

$$s := \int \sin(x)^n\,dx$$

```
> prim2 := unapply( s , n , x );
```

$$prim2 := n, x \rightarrow \int \sin(x)^n\,dx$$

```
> prim2( 5 , t );
```

$$-\frac{1}{5}\sin(t)^4\cos(t) - \frac{4}{15}\sin(t)^2\cos(t) - \frac{8}{15}\cos(t)$$

A more powerful procedure to create functions is **codegen[makeproc]** (available in release V.5). Let's use it for the same example:

```
> prim3 := codegen[makeproc]( s , [n,x] );
```

$$\textbf{proc}(n, x)\ int(sin(x)\hat{}n, x)\ \textbf{end}$$

Observe the square brackets around the parameter, which are necessary here. The result looks a little bit different, but essentially it is the same as `prim2`. However, `codegen[makeproc]` can handle more, for instance, a calculation in two or more steps. Here is a simple example:

```
> twostep:=codegen[makeproc]([pol=x^2-1, exp(pol)], x);
```

$$\textbf{proc}(x)\ \textbf{local}\ pol;\ pol := x\hat{}2 - 1;\ \exp(pol)\ \textbf{end}$$

```
> twostep(a,b);
```

$$\mathbf{e}^{(a^2-1)}$$

The first argument is a list of calculation steps (brackets necessary). The intermediate result is assigned to `pol` in the first step (but using = instead of :=), then `pol` is used in the second step. The variable `pol` is "local": it is not accessible outside the procedure:

```
> pol;
```

$$pol$$

In section 9.5 on page 123 and section 18.19 on page 271 you can see a more advanced use of this procedure.

6.8 Evaluation of names of procedures

A mathematical function in Maple has the data type **procedure**. A name referring directly to a procedure is not evaluated to that procedure, unless `eval` acts on that name. See section 5.5 on page 67.

There is a difference between procedures available from the Maple system and procedures that are defined by the user: these are printed to the screen with all details:

```
> eval(saddle);
```

$$saddle := x, y \rightarrow 5\,x^2 - 3\,y^2$$

```
> print(s5);
```

$$s5 := -\frac{1}{5}\sin(x)^4\cos(x) - \frac{4}{15}\sin(x)^2\cos(x) - \frac{8}{15}\cos(x)$$

Procedures defined in Maple's library are not printed fully:

```
> eval(sin);
```

$$\mathbf{proc}(x :: algebraic)\ \ldots\ \mathbf{end}$$

This behavior can be changed by the command `interface(verboseproc=2)`. See section 5.8 on page 71.

6.9 Derivative functions

Maple can calculate derivative functions with the procedure **D**:

```
> D(ln);
```

$$a \rightarrow \frac{1}{a}$$

```
> D(cos);
```

$$-sin$$

```
> D( x -> exp(a*x) );
```

$$x \rightarrow a\,\mathrm{e}^{(a\,x)}$$

Even the derivatives of some exotic functions are available, for instance the derivative of the **Dirac** function, defined as the 'function' on the real numbers that yields

zero for any input apart from zero, at which point it has a singularity such that $\int_{-\infty}^{\infty} \mathrm{Dirac}(t)dt = 1$.

```
> D(Dirac*sin);
```

$$(a \rightarrow \mathrm{Dirac}(1, a))\, sin + Dirac\, cos$$

The derivative of `Dirac` is known to Maple as `a -> Dirac(1,a)`.
The procedure D can also handle a function created by **codegen[makeproc]** if there are no elements in it that are not amenable to differentiation with `D`. Let's differentiate the procedure `twostep` from the previous section:

```
> D(twostep);
```

proc(x) **local** $polx, pol$; $polx := 2 * x$; $pol := x\hat{}2 - 1$; $polx * exp(pol)$ **end**

```
> %(a);
```

$$2\, a\, \mathbf{e}^{(a^2-1)}$$

In some cases, where such a procedure contains difficult elements such as sums, it needs some polishing by applying **codegen[prep2trans]** before `D` can do its job.

The procedure `D` can also handle unknown functions:

```
> D( f1*f2 );
```

$$\mathrm{D}(f1)\, f2 + f1\, \mathrm{D}(f2)$$

An advantage of using functions over using expressions is that calculating function values by *applying* functions is more efficient than *substituting* in expressions. The same is true for calculating *derivatives at a point* by using derivative functions. The easiest way of getting the derivative of tan at $ax + b$ is:

```
> (D(tan))(a*x+b);
```

$$1 + \tan(a\, x + b)^2$$

Do *not* omit the parentheses around `D(tan)`.

The alternative with `diff` is more lengthy:

```
> diff( tan(t) , t );
```

$$1 + \tan(t)^2$$

```
> eval( % , t=a*x+b );
```

$$1 + \tan(a\, x + b)^2$$

Do *not* confuse `diff` and `D`:

```
> diff( cos^3 , x );
```

$$0$$

In the last case, Maple differentiates an expression, perceiving `cos` as a variable, independent of `x`.

In the next example, Maple interprets `a` and `x` as mathematical functions:

```
> D( a*x^2 );
```

$$\mathrm{D}(a)\,x^2 + 2\,a\,\mathrm{D}(x)\,x$$

6.10 Derivatives of functions of more than one variable

The operator `D` can also be used for functions of more than one variable. In this case you must indicate in respect to which variable the function should be differentiated. That can be done with indices. For instance, in order to find the derivative in the second argument of a function, apply D[2]:

```
> g := (x,y) -> cos(x*y) + exp(2*y);
```

$$g := x, y \rightarrow \cos(x\,y) + \mathbf{e}^{(2\,y)}$$

```
> D[2](g);
```

$$x, y \rightarrow -\sin(x\,y)\,x + 2\,\mathbf{e}^{(2\,y)}$$

Now let's differentiate g in respect to the second and then to the first variable:

```
> D[1,2](g);
```

$$x, y \rightarrow -\cos(x\,y)\,y\,x - \sin(x\,y)$$

Maple supposes that partial differentiation operators commute. In most practical cases, the encountered functions have all their partial derivatives (of any order) continuous; then this is nothing to worry about.

```
> D[3,1,2](anyfunc);
```

$$D_{1,2,3}\,(anyfunc)$$

6.11 Conversion between `diff` and `D`

Conversion between `D` and `diff` notation is available:

```
> diff( u(t)*v(t) , t);
```

$$\left(\frac{\partial}{\partial t}\,\mathrm{u}(t)\right)\mathrm{v}(t)+\mathrm{u}(t)\left(\frac{\partial}{\partial t}\,\mathrm{v}(t)\right)$$

```
> convert( % , D );
```

$$\mathrm{D}(u)\,(t)\;\mathrm{v}(t)+\mathrm{u}(t)\;\mathrm{D}(v)\,(t)$$

```
> convert( % , diff );
```

$$\left(\frac{\partial}{\partial t}\,\mathrm{u}(t)\right)\mathrm{v}(t)+\mathrm{u}(t)\left(\frac{\partial}{\partial t}\,\mathrm{v}(t)\right)$$

However, `D(f)` cannot be converted to `diff` notation in a direct way because the differentiation parameter is lacking:

```
> convert( D(g) , diff );
```

$$\mathrm{D}(g)$$

6.12 Piecewise-defined functions and expressions

Functions can be defined piecewise as in the following example:

```
> f:=x->piecewise(x<0,-x^2, x<1,x^2, x^3);
```

$$f := x \to \mathrm{piecewise}(x < 0, -x^2, x < 1, x^2, x^3)$$

What this means can be seen better if we apply this function to `x`:

```
> f(x);
```

$$\begin{cases} -x^2 & x < 0 \\ x^2 & x < 1 \\ x^3 & otherwise \end{cases}$$

As you can see, the five arguments are to be read as follows:

- if $x < 0$ then $-x^2$
- if that is not the case and if $x < 1$ then x^2
- otherwise x^3

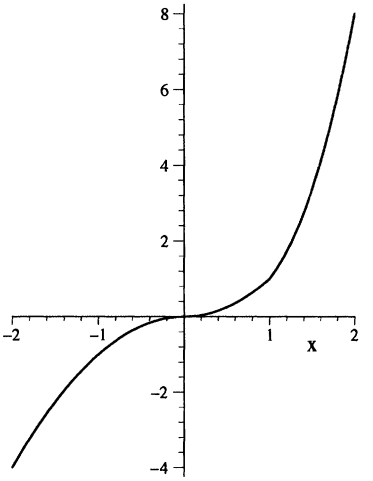
This function can be used as any other function:

```
> f(-3),f(1/2),f(3);
```

$$-9, \frac{1}{4}, 27$$

From release V.4 many procedures can handle such piecewise-defined functions or expressions, for instance:

```
> plot( f(x) , x=-2..2 );
```




```
> solve( f(x)=5*x-2 , x );
```

$$-\frac{5}{2} - \frac{1}{2}\sqrt{33}, \frac{5}{2} - \frac{1}{2}\sqrt{17}, 2$$

```
> diff( f(x) , x );
```

$$\begin{cases} -2x & x \le 0 \\ 2x & x < 1 \\ undefined & x = 1 \\ 3x^2 & 1 < x \end{cases}$$

```
> int( f(x) , x=-2..2 );
```

$$\frac{17}{12}$$

It is even possible to create other piecewise expressions by transformations such as:

```
> f(x^2);
```

$$\begin{cases} -x^4 & x^2 < 0 \\ x^4 & x^2 < 1 \\ 3x^2 & otherwise \end{cases}$$

```
> simplify(%);
```

$$\begin{cases} x^6 & x \le -1 \\ x^4 & x \le 1 \\ x^6 & 1 < x \end{cases}$$

```
> piecewise(x<0,f(x),x^4);
```

$$\begin{cases} \begin{cases} -x^2 & x < 0 \\ x^2 & x < 1 \\ x^3 & otherwise \end{cases} & x < 0 \\ x^4 & otherwise \end{cases}$$

```
> simplify(%);
```

$$\begin{cases} -x^2 & x \le 0 \\ x^4 & 0 < x \end{cases}$$

This can be converted into a function with `unapply`:

```
> g:=unapply(%,x);
```

$$g := x \mapsto \begin{cases} -x^2 & x \le 0 \\ x^4 & 0 < x \end{cases}$$

```
> g(4);
```

$$256$$

Because **piecewise** expressions can be handled so well, it can be useful to convert expressions and functions containing `abs`, `signum`, `Heaviside`, `max`, `min`, etc. to piecewise-defined functions with **convert(,piecewise)**:

```
> max(x+1,x^2);
```

$$\max(x+1, x^2)$$

```
> int(%,x=-2..2);
```

$$\int_{-2}^{2} \max(x+1, x^2)\, dx$$

```
> convert(%%,piecewise);
```

$$\begin{cases} x^2 & x \le \frac{1}{2} - \frac{1}{2}\sqrt{5} \\ x+1 & x \le \frac{1}{2} + \frac{1}{2}\sqrt{5} \\ x^2 & \frac{1}{2} + \frac{1}{2}\sqrt{5} < x \end{cases}$$

```
> int(%,x=-2..2);
```

$$\frac{16}{3} + \frac{5}{6}\sqrt{5}$$

6.13 Creating functions by elementary operations on functions

The **composition of two functions** can be made with the operator **@**, for instance,

```
> quint := x -> x^5;
```

$$quint := x \to x^5$$

```
> (quint @ sin) (x);
```

$$\sin(x)^5$$

```
> (sin @ quint) (x);
```

$$\sin\left(x^5\right)$$

The brackets are necessary here, otherwise you can get rubbish:

```
> quint @ sin(Pi/3);
```

$$quint@\frac{1}{2}\sqrt{3}$$

```
> (quint @ sin)(Pi/3);
```

$$\frac{9}{32}\sqrt{3}$$

For **repeated compositions**, you can use **@@**. For instance, you can get the function x $\mapsto$ `quint(quint(quint(x)))` as `quint@@3`. For example:

```
> (quint@@3) (x);
```

$$x^{125}$$

Please pay attention to the differences in printing:

```
> (sin@@2) (x); # This is in fact: sin(sin(x))
```

$$\left(sin^{(2)}\right)(x)$$

```
> sin(x)^2;      # This is in fact: sin(x) * sin(x)
```

$$\sin(x)^2$$

(After the sign **#** you can add **comment** in Maple.)
New functions can be created from existing functions with the elementary arithmetic operators **+**, **-**, *****, and **/**, and with the composition operators **@** and **@@**:

```
> f := (2*quint*sin - 3*cos@quint)/exp;
```

$$f := \frac{2\,quint\,sin - 3\,cos@quint}{exp}$$

```
> f(x);
```

$$\frac{2x^5\sin(x) - 3\cos\left(x^5\right)}{e^x}$$

In such a function expression, each name is interpreted as a function, if possible. For instance:

```
> (a*sin + b*cos) (x);
```

$$\mathrm{a}(x)\ \sin(x) + \mathrm{b}(x)\ \cos(x)$$

```
> D( a*sin + b*cos );
```

$$\mathrm{D}(a)\ sin + a\ cos + \mathrm{D}(b)\ cos - b\ sin$$

If you want D to consider a and b as constants, you can issue:

```
> D(a):=0: D(b):=0:
> D( a*sin + b*cos );
```

$$a\ cos - b\ sin$$

chapter 7

Graphics

This chapter gives an overview of the most useful graphical facilities in Maple. Not included are specific tools for differential equations, vector fields, etc.; these are discussed in Chapter 17, Solving differential equations.

7.1 Graphs of real functions in one real parameter

As demonstrated in previous chapters, the graph of a function in one parameter can be plotted by the procedure **plot**. For instance,

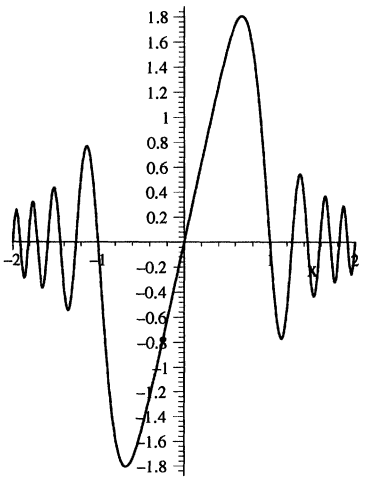
$$f: \quad x \mapsto \frac{\sin(\pi x^3)}{x^2}$$

can be plotted on the range $-2 \ldots 2$ by

```
> sin(Pi*x^3)/x^2 ;
```

$$\frac{\sin(\pi\, x^3)}{x^2}$$

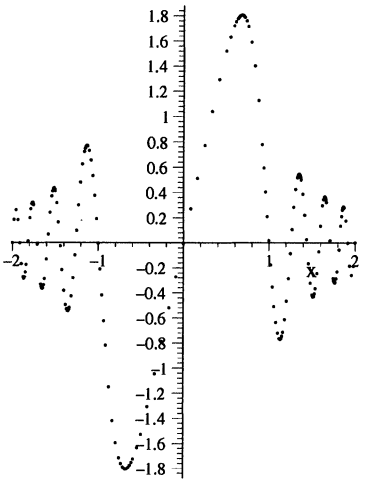
```
> plot( % , x=-2..2 );
```



It is possible to make some changes to this picture in an interactive way. You can bring the cursor into the region of the picture with the arrow keys and use the facilities of the toolbar at the top of the Maple window; or you can click the right

button of the mouse with the mouse arrow in that region and use the menu. Then you can change "Style", "Axes" and "Projection". For instance, to plot only the points of the graphic that are calculated by Maple, without connecting them, you can choose the option `Point` in the `Style` menu. After clicking on the `R` in the toolbar or on `Redraw` in the menu, you see the following picture, which can also be achieved with the command:

```
> plot( %% , x=-2..2 , style=POINT );
```



You are well advised to experiment a little with the choices offered by the menus. All changes made with graphic window menus can be achieved as well by adding options to the command, but with options you have many more facilities available, for instance `scaling=CONSTRAINED`, which causes the scalings of the axes to be equal.

In the next section, using such choices for a three-dimensional plot is demonstrated. In the following sections, effects of several choices and options are shown, but not of all options, for instance, not of options concerning color. Consult the online help with `?plot,options` or `?plot3d,options` for a particular option.

7.2 Graphs of real functions in two real parameters

The graph of a function in two parameters can be plotted by the procedure **plot3d**. For instance, the function

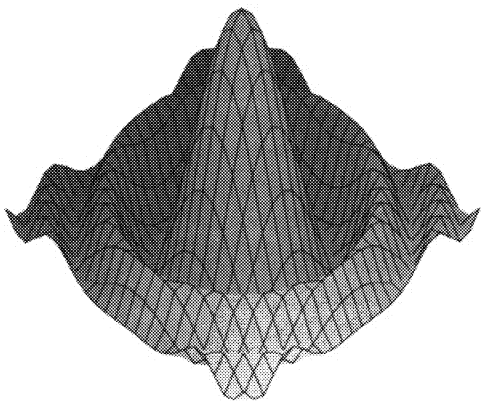
$$(x,y) \mapsto \frac{10}{x^2+y^2+1}\cos(x^2+y^2)$$

can be plotted in the range x from -3 to 3 and y from -3 to 3 by

```
> 10/(x^2+y^2+1)*cos(x^2+y^2);
```

$$10\,\frac{\cos(x^2+y^2)}{x^2+y^2+1}$$

```
> plot3d( % ,x=-3..3,y=-3..3);
```



Let's make some choices from the menu options, using the right mouse button or the toolbar.

- *Style:* Let's change the style to `Patch and contour`
- *Color:* As the pictures in this book are printed in black and white, the choices of this menu are not relevant here. Neither can the possibilities of `lighting` be used here; but this option *can* elucidate color pictures.
- *Axes:* We add a box with coordinates by choosing `Boxed` from this menu.
- *Projection:* Let's choose `Medium Perspective` instead of the default `No Perspective`.

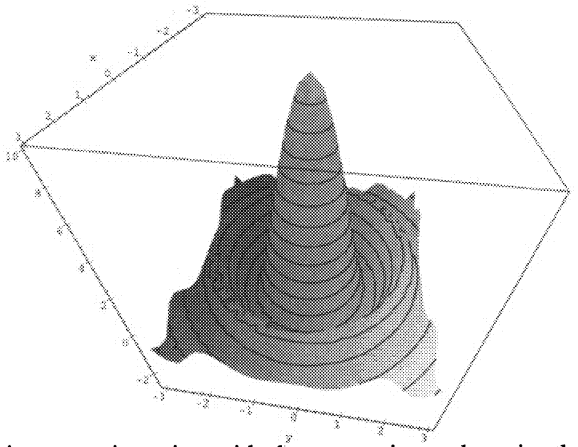
It is also possible to change the aspect (the direction of viewing the object) by pressing the (left) mouse button, keeping it pressed, and moving the mouse. In the MS-DOS version, use the cursor keys instead. Then you see a box moving, indicating the final direction of viewing. Moreover, the two spherical angles `Theta` and `Phi` are printed. The default is $\Theta = 45$ and $\Phi = 45$. In the next picture, this is changed to $\Theta = 8$ and $\Phi = 50$.

There is also an extra facility within the menu bar for changing the dimensions of the window.

As soon as a menu choice is made, the picture disappears from the window, leaving a box. The picture is redrawn if you click on the `R` in the toolbar or click on `Redraw` in the menu.

The result of the previous choices can also be created by a plot command with appropriate additional options.

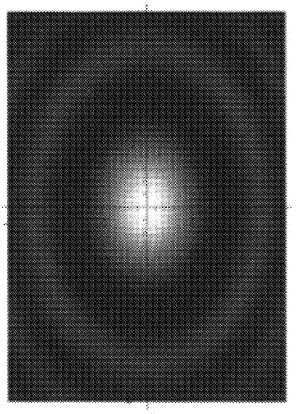
```
> plot3d(%%, x=-3..3,y=-3..3,style=PATCHCONTOUR,
>     axes=BOXED,projection=0.1,orientation=[8,50]);
```



In practice, experimenting with the menus is much easier than using options as additional arguments, but there are options that can do things that cannot be realized by the menus, such as changing the number of points to be calculated by the option `numpoints=...`, or rendering just a part of the picture by the option `view=` . It is possible to set certain options as defaults by using the procedures **plots[setoptions]** and **plots[setoptions3d]**.

Here is another method of rendering a graph of a function in two parameters: a **density plot**. Let's apply this to the expression of the previous plotting.

```
> plots[densityplot]( %%% , x=-3..3 , y=-3..3 ,
>    numpoints=2500 , style=PATCHNOGRID );
```



7.3 Assigning, manipulating, and printing graphical objects

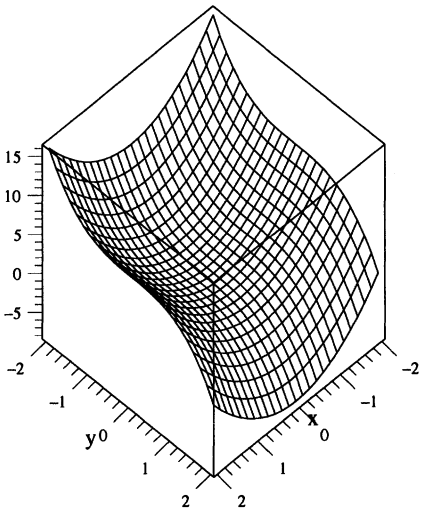
Graphical procedures such as `plot` and `plot3d` yield special Maple objects. Such an object can be called by the **ditto** (`%`) and it may be assigned to a name. For instance:

```
> saddle:=plot3d(2*x^2-y^3,x=-2..2,y=-2..2,
>     axes=BOXED, style=HIDDEN):
```

There is a special reason why we have used a colon instead of a semicolon in terminating the command: if we had not done so, the plot data structure would have been represented instead of the picture, making a lot of uninteresting data scroll over the screen.

If you want to see the picture, you can use **print** in this case. In some exceptional cases, using `print` yields the internal data structure; in such a case, you can use **plots[display]** instead.

```
> print( saddle );
```

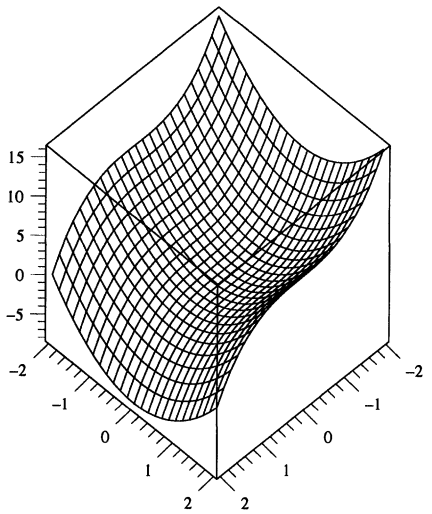


If you are using plotting in loops or procedures, see also section E.5 on page 302.

It is easy to send a worksheet to a printer or save it as a PostScript file, both with the menu option `Print` in the toolbar menu `File`, but if you want to print a separate picture or export it to a file, set `Plot Display` to `Window` in the `Options` menu. Now each picture is rendered in a separate worksheet, which can be printed or exported to a file with the `Print` option in the `File` menu. There are several formats available such as PostScript, jpeg, etc. Other facilities for this purpose are shown in Appendix C, *The user interface for text-only versions.*

There are several commands for **manipulation of graphical objects**, for instance `plottools[rotate]`; let's use this to rotate the saddle over $\frac{1}{2}\pi$ around the third axis:

```
> plottools[rotate](saddle,0,0,Pi/2);
```



Other commands for manipulation are:

- `plottools[translate]`
- `plottools[rotate]`
- `plottools[reflect]`
- `plottools[homothety]` (scaling the axes to the same ratio)
- `plottools[scale]`
- `plottools[project]`
- `plottools[transform]` (very general, applying your own function)
- `plots[display]`, which can change options and can do a lot more
- `plottools[stellate]` (for polygons).

With some of them (for instance `plottools[transform]`) it is also possible to change the dimension.

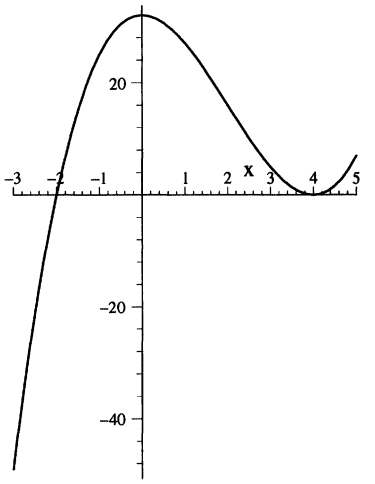
7.4 Vertical asymptotes and discontinuities

Here is an expression with two zeroes:

```
> Y := x^3 - 6*x^2 + 32;
```

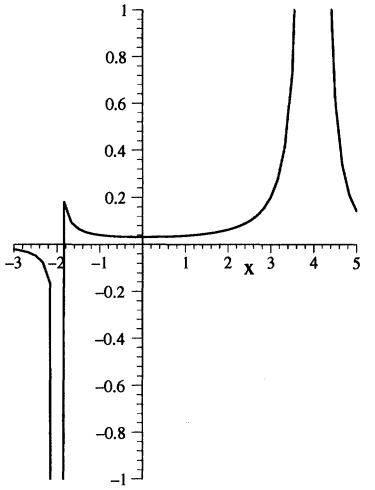
$$Y := x^3 - 6x^2 + 32$$

```
> plot(Y,x=-3..5);
```



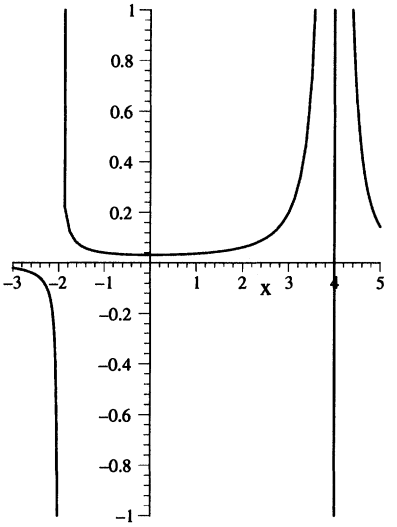
Let's ask for the graph of $1/Y$.

```
> plot( 1/Y , x=-3..5 , -1..1 );
```



This last graph is not correct, as can be seen from the previous graph. Maple has calculated some points and connected them without looking for discontinuities. To make Maple check on discontinuities or disconnected domains, supply the extra option `discont=true`.

```
> plot( 1/Y , x=-3..5, -1..1 , discont=true );
```

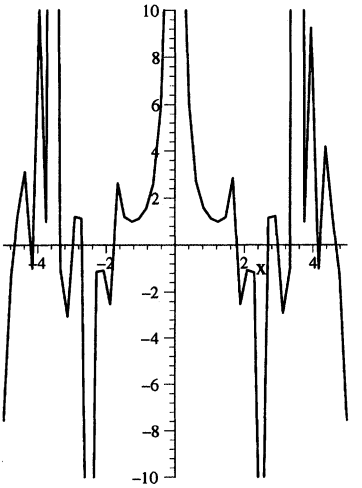


As an alternative, you can revert to plotting only the points that are calculated by Maple. Here is an example, first without special options:

```
> y1:= 1/sin(x^2);
```

$$y1 := \frac{1}{\sin(x^2)}$$

```
> plot( y1 , x=-5..5 , -10..10 );
```

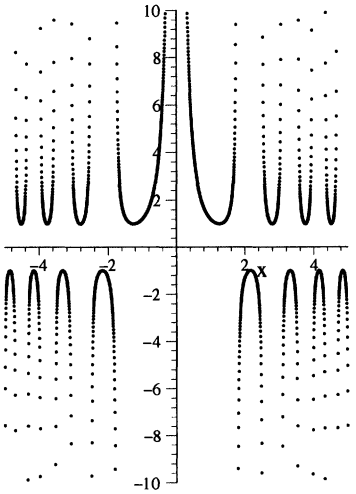


Here `plot` with the option `discont=true` yields an error due to a bug:

```
> plot( y1 , x=-5..5 , -10..10 , discont=true );
Error, Could not determine discontinuities
```

So, let's use the option `style=point` and at the same time ask Maple to calculate a sufficient number of points, say 3,000.

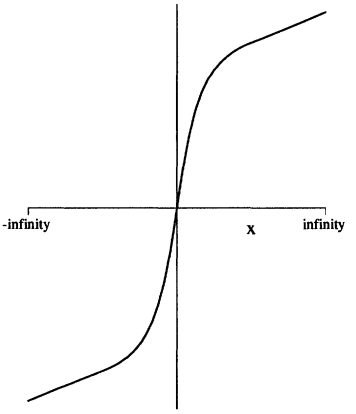
```
> plot( y1 , x=-5..5 , -10..10 ,
>     style=POINT , numpoints=3000 );
```



7.5 Graphs with ranges to infinity

Maple offers the facility for plotting on infinite ranges, for instance

```
> plot(arctan(x),x=-infinity..infinity);
```



This is realized by a transformation of the first coordinate with the following function:

$$x \mapsto \begin{cases} \frac{x}{8} & \text{if} \quad -4 \le x \le 4 \\ 1 - \frac{2}{x} & \text{if} \quad x \ge 4 \\ -1 - \frac{2}{x} & \text{if} \quad x \le -4 \end{cases}$$

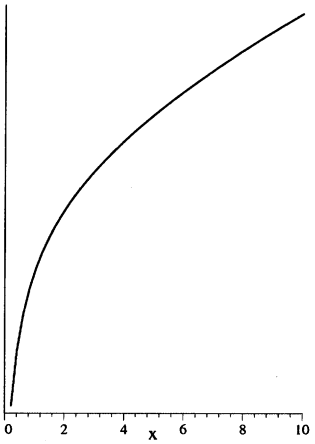
7.6 Logarithmic scalings

Log plots and log-log plots are available.

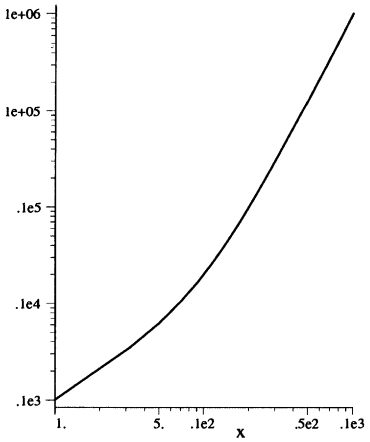
```
> 100*x+x^3;
```

$$100\,x + x^3$$

```
> plots[logplot]( % , x=0..10 );
```



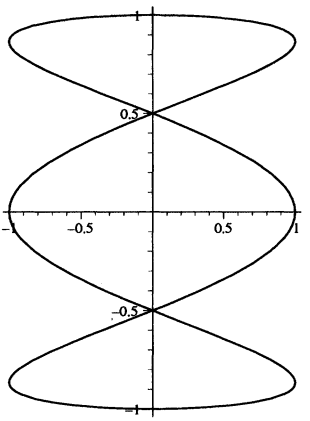

```
> plots[loglogplot]( %% , x=1..100 );
```



7.7 Parameterized curves and surfaces

Here is a graph of a **parameterized curve** in two dimensions:

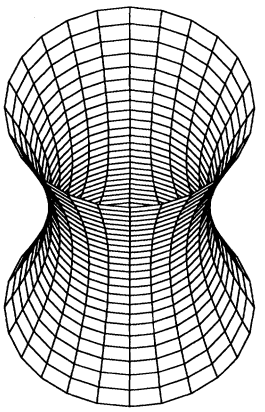
```
> plot( [sin(3*x),cos(x),x=0..2*Pi] );
```



Now an example in three dimensions: the hyperboloid described by the parameterizing function

$$(t, u) \mapsto [\cos(t)\cosh(u)\ ,\ \sin(t)\cosh(u), sinh(u)]$$

```
> plot3d( [cos(t)*cosh(u),sin(t)*cosh(u),sinh(u)] ,
>    t=0..2*Pi,u=-1..1);
```



Observe the differences in syntax used in parameterizing: in the two-dimensional case the range of the parameter is given as the third element of the list, so the basic syntax is:

```
plot( [ x(t),y(t),range_of_t ] )
```

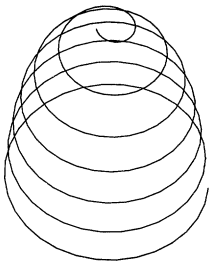
so it is given *within* the pair of brackets, but in the three-dimensional case the ranges are given *outside* the list:

```
plot3d( [ x(t,u),y(t,u),z(t,u) ] , range_of_t,
range_of_t )
```

Such a difference between parameterizing in two and three dimensions can also be found in analogous cases.

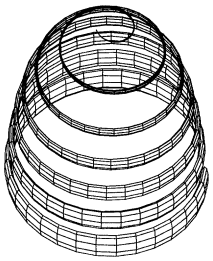
A curve in three-dimensional space can be drawn by **plots[spacecurve]**.

```
> plots[spacecurve]([(1+5*arctan(t))*cos(10*t),
>     (1+5*arctan(t))*sin(10*t),-t], t=0..4,
>     numpoints=200);
```



We can even blow up this curve to a tube by:

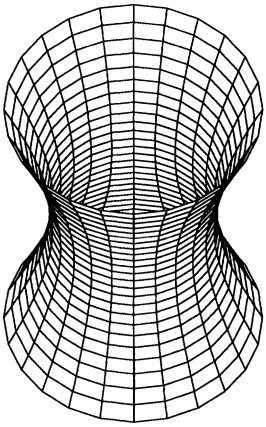
```
> plots[tubeplot]([(1+5*arctan(t))*cos(10*t),
>     (1+5*arctan(t))*sin(10*t),-t],t=0..4,
>     radius=0.07*t,numpoints=200);
```



7.8 Different types of coordinates

The second picture in the previous section could have been drawn more easily with cylindrical coordinates, like the following:

```
> plot3d( [cosh(t),phi,sinh(t)] , t=-1..1 , phi=0..2*Pi ,
>    coords=cylindrical );
```

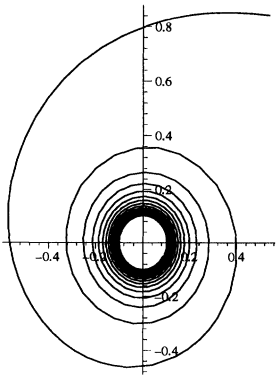


Spherical coordinates can be used in `plot3d` by using the option `coords=spherical`.

Maple offers a lot of coordinate systems: see the on-line help. Moreover, you can create your own coordinate system with `addcoords` (which must be read with `readlib`first).

Here is another example. In two-dimensional space, polar coordinates can be used, as in the following example, where the so-called lituus, described by the equation $r^2\phi = 1$, is drawn.

```
> plot([1/sqrt(phi),phi,phi=1..100],coords=polar);
```



7.9 Empty plots caused by complex values

Sometimes, plotting a graph may not be successful. This may be caused by approximations that should be real numbers but turn out to contain a small imaginary component.

```
> convert( tan(x) , exp );
```

$$-\frac{I\left(\left(\mathrm{e}^{(I\,x)}\right)^2-1\right)}{\left(\mathrm{e}^{(I\,x)}\right)^2+1}$$

Obviously, the values of this expression are real for real `x`, but `plot` gets into trouble.

```
> plot( % , x=-1..1 );
Plotting error, empty plot
```

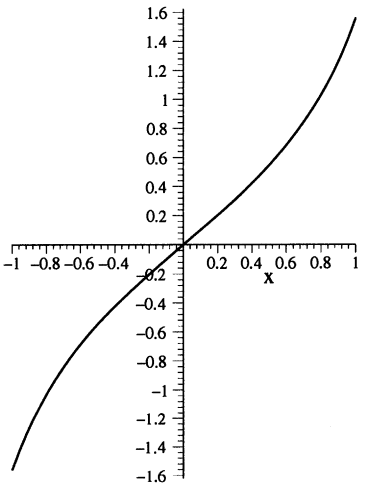
This is caused by the fact that `plot` uses numerical approximations. For instance,

```
> evalf( subs(x=0.5,%%) );
```

$$.5463024898 + .8060353952\,10^{-11}\,I$$

Such a complex number is not accepted by the `plot` procedure. However, it is easy to mend this problem: plot the real part of the expression.

```
> plot( evalc(Re(%%%)) , x=-1..1 );
```



7.10 Plotting data

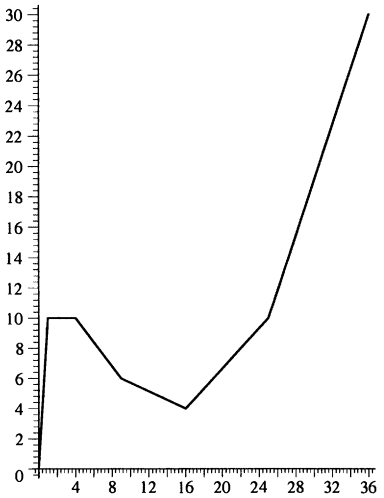
A set of two- or three-dimensional data can be represented in a graphic. Let's create a sequence of pairs of numbers.

```
> data:=seq([k^2,k^3-8*k^2+17*k],k=0..6);
```

$$data := [0, 0], [1, 10], [4, 10], [9, 6], [16, 4], [25, 10], [36, 30]$$

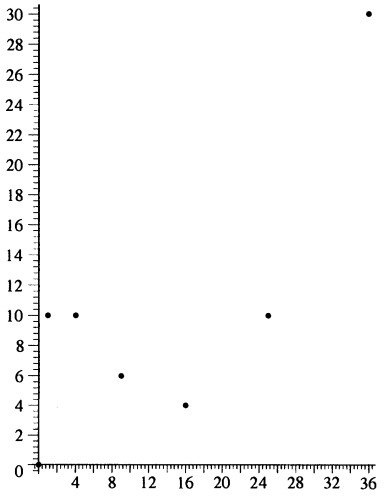
This sequence can be represented as a continuous graph, in fact, a polyline.

```
> plot( [ data ] );
```



or as a set of points, here indicated by small circles.

```
> plot( [ data ] , style=POINT );
```



In both cases, the data should be bundled together as one argument by enclosing them within an outer pair of square brackets.

It is also possible to create a smooth curve from these data with the aid of the procedure **spline**, which creates a piecewise polynomial function from data. This must be loaded first:

```
> readlib(spline);
```

$$\textbf{proc}(X, Y, z, d) \ \ldots \ \textbf{end}$$

For `spline` the data must be structured in a different way: a list of all first coordinates and a list of all second coordinates; see Chapter 10, *Manipulating several objects at once*:

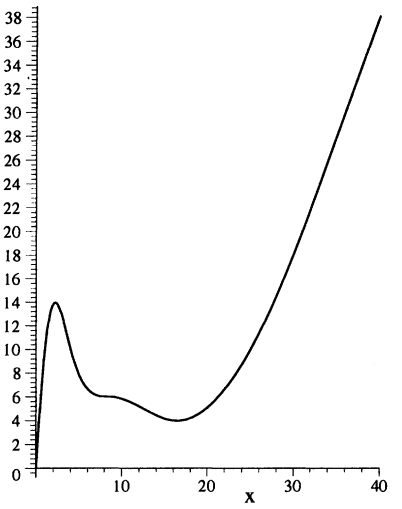
```
> xdata:=[seq(op(1,data[i]),i=1..7)];
```

$$xdata := [0, 1, 4, 9, 16, 25, 36]$$

```
> ydata:=[seq(op(2,data[i]),i=1..7)];
```

$$ydata := [0, 10, 10, 6, 4, 10, 30]$$

```
> plot( spline(xdata,ydata,x) , x=0..40 );
```



In a comparable way, a sequence of triads of numbers can be plotted. However, the procedure `plot3d` cannot be used for this purpose. Use `plots[surfdata]`.

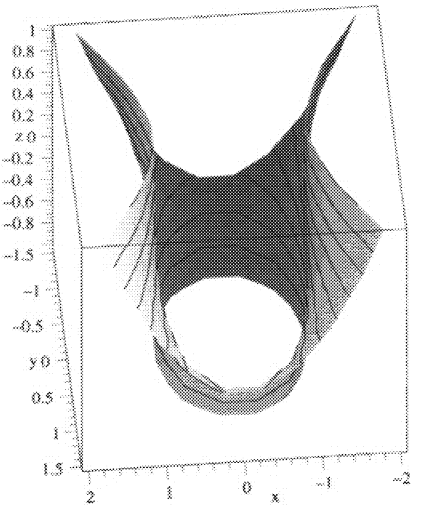
For statistical purposes you can use histograms, boxplots, etc.: see the on-line help for `stats,statplots`.

You can also use `plots[matrixplot]` and `plots[sparsematrixplot]`.

7.11 Graphs of relations or implicitly defined functions

A facility for plotting graphs of algebraic relations in two and three dimensions is available:

```
> plots[implicitplot3d]( x^2-z*y^2=1 ,
>    x=-2..2 , y=-1.5..1.5 , z=-1..1 , axes=BOXED ,
>    style=PATCHCONTOUR , orientation=[85,45] );
```



For the two-dimensional case, use `plots[implicitplot]`.

Generally, the results of plotting graphs of functions and parameterized curves and surfaces have better quality and can be produced faster than graphs of relations.

In some cases you also can use `algcurves[parameterization]`.

7.12 Combining graphs

If you want to combine graphical objects in one picture, first create each of them separately.

```
> parabolic := plot( x^2 , x=-1.5 .. 1.5 ):
```

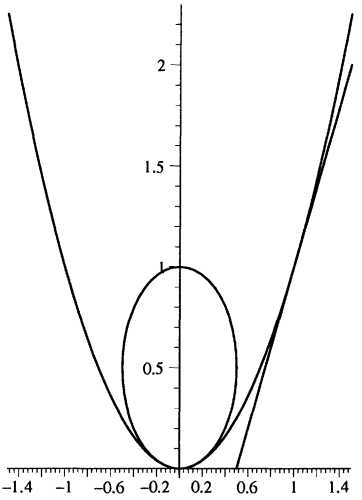
Remember that an assignment of a graphical object to a name must be terminated by a colon in order to keep Maple from printing the internal data structure to the screen.

In the same way we can assign two other plots to names.

```
> circle := plot( [1/2*cos(phi) , 1/2*sin(phi)+1/2 ,
>    phi=0..2*Pi] ):
> line := plot( [1+h,1+2*h,h=-1/2..1/2] ):
```

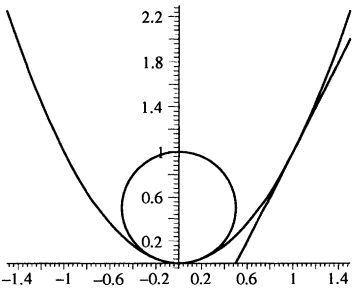
Now let's ask Maple to print all three together with the aid of the procedure **plots[display]**:

```
> plots[display]( {parabolic,circle,line} );
```



Note that Maple chooses the range of the second coordinate, if the user does not specify this. Therefore, a circle is rendered as an ellipse, generally. This can be rectified by adding the option **scaling=CONSTRAINED**:

```
> plots[display]( {parabolic,circle,line},
>    scaling=CONSTRAINED );
```

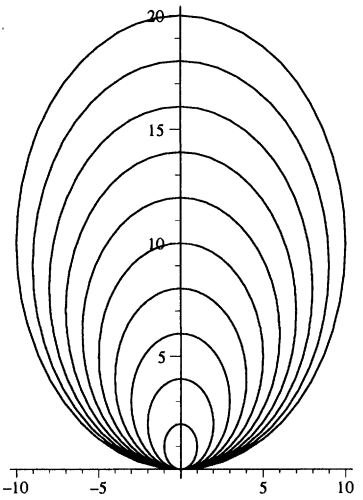


With `display` it is easy to plot a bundle of graphs in one picture. For instance, let's create a sequence of circle graphs without printing them.

```
> seq ( plot([n*cos(phi) , n*sin(phi)+n , phi=0..2*Pi]) ,
>    n=1..10 ):
```

These can be plotted together by

```
> plots[display]( [%] );
```



In the same way, `display` can be used for combining three-dimensional graphics. (In earlier releases, use `plots[display3d]`.)

7.13 Maple's movies

It is easy to create a small movie of parameterized graphics in Maple, with or without looping, with the procedures **plots[animate]** and `plots[animate3d]`. If you are interested, try some examples given by Maple in the on-line help for these two procedures. Moreover, you can use `plots[display]` to create a movie from a list of plot objects by adding the option `insequence=true`. It is also possible to use html format for creating movies to be played with software for these formats.

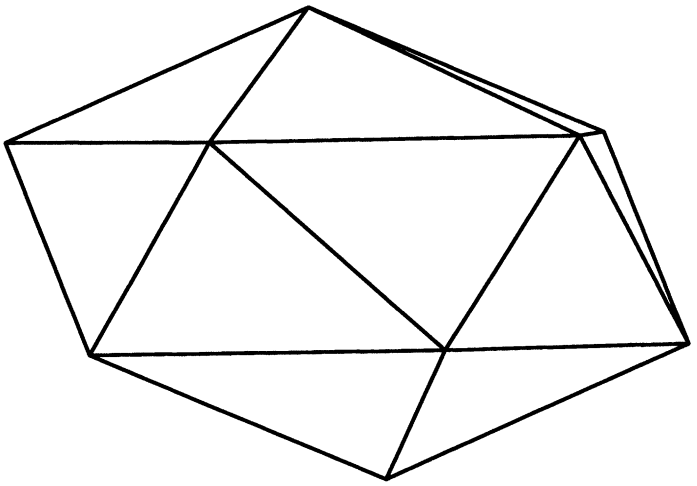
7.14 More tools in graphics

The `plots` and `plottools` packages contain several procedures not mentioned previously for special types of plots, for instance some ready-to-use pictures, such as:

```
> plottools[icosahedron]([0,0,0],1):
```

The result of this command is suppressed by the colon: it is not a picture but an internal structure for polygons. To get a picture, enter:

```
> plots[display](%);
```



Special tools aimed at differential equations can be found in the packages `plots` and `DEtools`. These are discussed in section 17.5 on page 238.

The packages `geometry` and `geom3d` for geometry and the pedagogical package `student` also contain some graphical tools.

chapter 8

Taylor or Laurent expansion and limits

When an exact algebraic computation fails, you can often resort to series approximations, using the theory of Taylor and Laurent expansions. Sometimes, a series approximation is even sufficient to get exact results, for instance, in calculations of limits. This idea is exploited by Maple; therefore, this chapter includes calculation of limits.

8.1 Taylor expansion

For the calculation of a Taylor expansion of a function in one variable, the procedure **series** can be used:

```
> (x+1)/(cos(x)+2);
```

$$\frac{x+1}{\cos(x)+2}$$

```
> series( % , x=0 , 5 );
```

$$\frac{1}{3}+\frac{1}{3}x+\frac{1}{18}x^2+\frac{1}{18}x^3+\frac{1}{216}x^4+O(x^5)$$

We have asked for a Taylor approximation centered at $x = 0$. The term $O(x^5)$ indicates a remainder term such that $\lim_{x\to 0} \frac{O(x^5)}{x^5}$ exists.

The second argument to `series` determines the variable in which the expression is to be expanded, and the point from which this is to be done:

```
> series( exp(x) , x=a , 5 );
```

$$\mathrm{e}^a+\mathrm{e}^a\,(x-a)+\frac{1}{2}\,\mathrm{e}^a\,(x-a)^2+\frac{1}{6}\,\mathrm{e}^a\,(x-a)^3+\frac{1}{24}\,\mathrm{e}^a\,(x-a)^4+O((x-a)^5)$$

The last argument to `series` may seem to ask for a fifth order expansion, but the next section explains why that is not true.

8.2 The order of a series expansion

If Maple calculates a series expansion of an expression, generally it calculates series expansions of "elementary" subexpressions first and then combines these expansions. The third argument to `series`, 5 in the previous examples, determines the order of the series expansions of these subexpressions. It is quite possible that the series expansion found at the end has lower order than these intermediate results. So sometimes you must experiment a little with the order argument to `series`. For instance,

```
> sin(x)^4/x^4;
```

$$\frac{\sin(x)^4}{x^4}$$

```
> series( % , x=0 , 5 );
```

$$1 + O(x^2)$$

The result is not of order 5 but only 2. In this case, order 9 in the calculations is necessary for getting a series approximation of order ≥ 5:

```
> series( %% , x=0 , 9 );
```

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

However, `series` remembers the previous results. So, if we again ask for a series expansion of order 5, Maple does not try to calculate it, but it remembers this sixth order result and uses it:

```
> series( %%% , x=0 , 5 );
```

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

If no third argument is supplied to `series`, the expansion order is taken to be equal to the value of the environment variable **Order**, which is initially 6.

8.3 Estimating the order term

A rough estimation of the order term can be calculated by using the Taylor Remainder Theorem. Let's estimate the remainder for order 6 on the domain $[-1 \ldots 1]$ in the following series expansion:

```
> Y := sin(1+x^2);
```

$$Y := \sin(1 + x^2)$$

```
> series( % , x=0 , 6 );
```

$$\sin(1) + \cos(1)x^2 - \frac{1}{2}\sin(1)x^4 + O(x^6)$$

First we must take the sixth order derivative:

```
> diff( Y , x$6 );
```

$$-64\sin\left(1+x^2\right)x^6+480\cos\left(1+x^2\right)x^4+720\sin\left(1+x^2\right)x^2-120\cos\left(1+x^2\right)$$

Now we can use the procedure **numapprox[infnorm]**, which estimates numerically the maximal absolute value of an expression:

```
> numapprox[infnorm]( % , x=-1..1 );
```

$$488.3141273$$

So we get a rough estimate of an upper limit of the difference between function value and series value on the domain $[-1..1]$ as $\frac{489}{6!}\,x^6$.

8.4 The subexpression structure of results from `series`

A result from `series` may look like a polynomial up to the order term, but generally it has a special data structure within the computer. As a consequence, such a result behaves differently from polynomials, generally. Here is an example. Consider the polynomial

```
> pn := expand( (1+x)^3 );
```

$$pn := x^3+3x^2+3x+1$$

If we ask for a series expansion to order 4 of this third-degree polynomial, the order term is rightly omitted:

```
> sr := series( (1+x)^3, x=0, 4);
```

$$sr := 1+3x+3x^2+O(x^3)$$

The result is printed as if it is a polynomial, but it is a Maple object of a quite different structure; therefore, subtracting `sr` and pn does not yield $O(x^3)-x^3$:

```
> sr - pn;
```

$$\left(1+3x+3x^2+O(x^3)\right)-x^3-3x^2-3x-1$$

The structural difference between pn and `sr` can be shown with **whattype** and **op**:

```
> whattype(pn); op(pn);
```

$$+$$

$$x^3,\ 3x^2,\ 3x,\ 1$$

```
> whattype(sr); op(sr);
```

$$series$$

$$1,\ 0,\ 3,\ 1,\ 3,\ 2,\ 1,\ 3$$

First we see, that pn is just a *sum*. Its summands (the operands of the + operator) are shown here with the aid of op. Then we see that sr has the **series structure**, with coefficient 1 to the zeroth-order term, coefficient 3 to the first-order term, coefficient 3 to the second-order term and coefficient 1 to the third-order term. The expansion variable x can be obtained with op(0,sr).

The series structure may be somewhat willful occasionally, but it can prevent some manipulations that could be hazardous. For instance,

```
> series(cos(x),x=0,5);
```

$$1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 + O(x^5)$$

```
> subs(x=0,%);
```

$$1$$

```
> subs(x=1,%%);
Error, invalid substitution in series
```

The first substitution is acceptable, but the second cannot generate an exact result because of the existence of the order term, so this command is rejected. If the user insists on substitution of 1 and wants to discard the order term, the series expansion must first be converted to a polynomial:

```
> convert( %% , polynom );
```

$$1 - \frac{1}{2}x^2 + \frac{1}{24}x^4$$

```
> subs(x=1 , % );
```

$$\frac{13}{24}$$

The calculation of a Taylor expansion is based on differentiation, which is **reliable** in Maple in principle.

8.5 The leading term

If only the leading term of a series (not vanishing term of minimal degree) is to be calculated, use the following construction:

```
> series( leadterm(cos(x^30+x^50)-1) , x=0 , 1000000 );
```

$$-\frac{1}{2}x^{60}$$

The third argument, 1000000, has been chosen absurdly high. There is no objection to so doing. It does not slow down calculations in practice and prevents Maple

from failing; if the order (the third argument) is not high enough, the result is an order term:

```
> series( leadterm(cos(x^30+x^50)-1) , x=0 , 10 );
```

$$+O(x^{60})$$

As this procedure has to test coefficients of a series expansion for being zero, this procedure is not fully **reliable** in principle, but generally no reliability problems of this type are expected.

8.6 Laurent, Puisseux, and generalized truncated power series

The procedure `series` can do more than Taylor expansion. For instance,

```
> s1 := series( 1/tan(x^2) , x=0 , 11 );
```

$$s1 := x^{-2} - \frac{1}{3}x^2 - \frac{1}{45}x^6 + O(x^7)$$

Here a Laurent expansion has been calculated. Notice that the third argument (the order 11) yields an expansion up to degree 6. As explained earlier in this chapter, this is caused by the fact that `series` calculates a result based on intermediate calculations of series expansions of order 11. In practice, you must experiment with the order argument to `series`.

This result can be converted to a generalized polynomial, containing terms of negative integer degree:

```
> convert( % , polynom );
```

$$\frac{1}{x^2} - \frac{1}{3}x^2 - \frac{1}{45}x^6$$

If the limit of a function f to ∞ exists, you can ask for a possible Laurent expansion in ∞:

```
> series( arctan(x) , x=infinity , 8 );
```

$$\frac{1}{2}\pi - \frac{1}{x} + \frac{1}{3}\frac{1}{x^3} - \frac{1}{5}\frac{1}{x^5} + \frac{1}{7}\frac{1}{x^7} + \mathrm{O}\left(\frac{1}{x^8}\right)$$

The calculation can be represented by substituting $y = 1/x$ in $\arctan(x)$, expanding the result in $y = 0+$, and substituting $y = 1/x$ in the result. You can use the procedure **asympt**, too.

Here is a case where no Laurent expansion exists, but where `series` yields a Puisseux expansion:

```
> series( sqrt(x^3 + x) , x=0 , 5 );
```

$$\sqrt{x} + \frac{1}{2}x^{(5/2)} - \frac{1}{8}x^{(9/2)} + \mathrm{O}\left(x^{(13/2)}\right)$$

This is a very special case, as the result of `series` is *not* of type `series` in this case, but simply a sum of a polynomial in `x` and a formal order term.

```
> whattype( % ); op( %% );
```

$$+$$

$$\sqrt{x},\ \frac{1}{2}x^{(5/2)},\ -\frac{1}{8}x^{(9/2)},\ O\left(x^{(13/2)}\right)$$

This order term can be omitted by substituting `0` for it:

```
> subs( O=0 , %%% );
```

$$\sqrt{x}+\frac{1}{2}x^{(5/2)}-\frac{1}{8}x^{(9/2)}+0\left(x^{(13/2)}\right)$$

```
> % ;
```

$$\sqrt{x}+\frac{1}{2}x^{(5/2)}-\frac{1}{8}x^{(9/2)}$$

The last step evaluates the result of the substitution. This is necessary because, after a substitution, the result is presented without automatic evaluation and simplification.

If the growth of a function is too fast, Maple may use a generalized series:

```
> series( x^x , x=0 , 2 );
```

$$1+\ln(x)x+O(x^2)$$

Here the coefficient of `x` is `-ln(1/x)`, not a constant, but its order is lower than the order of the monomial that it belongs to, in this case, lower than `x`.

8.7 Application of `series` to integration

Series can be used when no exact antiderivative (primitive) function can be found. For instance,

```
> x^4/cos(x);
```

$$\frac{x^4}{\cos(x)}$$

```
> series( % , x=0 , 12 );
```

$$x^4+\frac{1}{2}x^6+\frac{5}{24}x^8+\frac{61}{720}x^{10}+O(x^{12})$$

```
> int( % , x );
```

$$\frac{1}{5}x^5+\frac{1}{14}x^7+\frac{5}{216}x^9+\frac{61}{7920}x^{11}+O(x^{13})$$

The approximation of this antiderivative now has the `series` data type like the original expression yielded by `series`. In order to use it for further computations, you can apply `convert( ,polynom)` or, as explained in the next section, `convert( ,ratpoly)`.

If you want to use a series expansion for a definite integral, convert that series to a polynomial or a rational expression, and then apply `int`, but, please, remember, that you are responsible for neglecting the order term by removing it. This term is why Maple refuses to calculate a definite integral of an object of type series.

Sometimes, `series` can be applied to an unevaluated integral (or an inert integral), but only if expansion in 0 is asked for; for instance, in a variant of the previous example:

```
> int( x^4/cos(x) , x );
```

$$\int \frac{x^4}{\cos(x)}\,dx$$

```
> series( % , x=0 , 12 );
```

$$\frac{1}{5}x^5 + \frac{1}{14}x^7 + \frac{5}{216}x^9 + \frac{61}{7920}x^{11} + O(x^{13})$$

The use of series in solving equations or differential equations is discussed in section 16.10 on page 230 and section 17.6 on page 240. Series expansions of solutions of equations (series approximations to implicitly defined functions) are discussed in section 16.9 on page 227.

8.8 Numerical evaluation of a series

For numerical evaluation of a series, it can be efficient to convert it to a rational expression that has the present series as its series approximation (of the given order). This is possible with `convert( , ratpoly )`. Subsequently applying `convert( ,confrac)` can speed up things still more.

8.9 Multivariate Taylor expansion

For Taylor expansion of functions of more than one variable, the procedure **mtaylor** is available. This must be read from the miscellaneous library with `readlib` before it can be used:

```
> readlib(mtaylor):
> m :=  y - y*x - x^2 + sin(a*y+b*x);
```

$$m := y - y\,x - x^2 + \sin(a\,y + b\,x)$$

```
> mtaylor(m,{x=0,y=0},2);
```

$$b\,x + (1+a)\,y$$

```
> mtaylor(m,{x=0,y=0},3);
```

$$b\,x + (1+a)\,y - y\,x - x^2$$

```
> mtaylor(m,{x=0,y=0},4);
```

$$b\,x+(1+a)\,y-y\,x-x^2-\frac{1}{6}\,b^3\,x^3-\frac{1}{2}\,a\,y\,b^2\,x^2-\frac{1}{2}\,a^2\,y^2\,b\,x-\frac{1}{6}\,a^3\,y^3$$

The result is not of type `series` and does not contain an order term. In fact, `mtaylor` uses the simple trick of substituting `t*x` for `x`, `t*y` for `y`, etc., calculating the Taylor expansion to `t` with order equal to the third argument to the call to `mtaylor` (excluding the possibility of a Laurent expansion, etc.), and then converting the result to a polynomial and substituting `1` for `t`. This explains the following result:

```
> mtaylor( (x^3-y^3)/(x^2+y^2) , {x,y} , 10 );
```

$$\frac{x^3-y^3}{x^2+y^2}$$

From the same reason, you may guess the background for the following error message with its not very useful advice:

```
> mtaylor( x/(x^2+y^2) , {x=0,y=0} , 3 );
Error, does not have a taylor expansion, try series()
```

A variant with weighted orders of the variables is available with an option to `mtaylor`.

8.10 Calculating limits

The procedure **limit** can be used for the calculation of limits:

```
> limit( sin(x)/x , x=0 );
```

$$1$$

```
> limit( arctan(x) , x=infinity );
```

$$\frac{1}{2}\,\pi$$

```
> limit( 1/x , x=0 );
```

$$undefined$$

If it is desirable, you can indicate the *direction* with a third argument:

```
> limit( 1/x , x=0 , left );
```

$$-\infty$$

The possible directions are `left`, `right`, `real`, and `complex`. Where the limit point is not ∞ or $-\infty$ the default is `real` (bidirectional).

As limits are calculated with the aid of `series`, it can be important that series expansion is executed up to sufficient order. Sometimes, it can help to raise the level of **Order**:

```
> series(cos(x),x=0,12);
```

$$1-\frac{1}{2}x^2+\frac{1}{24}x^4-\frac{1}{720}x^6+\frac{1}{40320}x^8-\frac{1}{3628800}x^{10}+O(x^{12})$$

```
> convert(%,polynom);
```

$$1-\frac{1}{2}x^2+\frac{1}{24}x^4-\frac{1}{720}x^6+\frac{1}{40320}x^8-\frac{1}{3628800}x^{10}$$

```
> limit((cos(x)-%)/x^12,x=0);
```

$$\lim_{x\to 0}\frac{\cos(x)-1+\frac{1}{2}x^2-\frac{1}{24}x^4+\frac{1}{720}x^6-\frac{1}{40320}x^8+\frac{1}{3628800}x^{10}}{x^{12}}$$

The procedure `limit` has called `series`, expanding the first argument up to the standard order of series expansion given by the value of the name `Order`, and this expansion is not sufficient for finding the limit. So let's raise this order and start the calculation again:

```
> Order:=13;
```

$$Order := 13$$

```
> %%;
```

$$\frac{1}{479001600}$$

In the following case, Maple cannot give a result:

```
> limit(exp(-p1*x),x=infinity);
```

$$\lim_{x\to\infty} \mathbf{e}^{(-p1\,x)}$$

The user might intend `p1` to represent a positive real number, but Maple does not know so. You can tell this to Maple by using the procedure **assume**. See section 3.4 on page 36 or section A.4 on page 278.

```
> assume(p1,positive);
```

Now Maple can find the limit. As the previous command leaves no result on the stack, we can use the double quote for repeating the limit command:

```
> % ;
```

$$0$$

The procedure `PDEtools[dchange]` can be used for substitutions in limits, alternatively using the inert variant `Limit` of `limit`. (In releases before Release 5, use `student[changevar]` instead of `PDEtools[dchange]`.)

Generally, calculating limits is **reliable**.

8.11 Multiple limits

Calculation of multiple limits is available in principle:

```
> limit( (x^2+y^2)/(x+y+1) , {x=0,y=0} );
```

$$0$$

```
> limit( (x^2-y^2)/(x^2+y^2) , {x=0,y=0} );
```

$$undefined$$

8.12 Continuity, singularities, and residues

The procedure **iscont** tries to check if a function is continuous on a given interval. It uses some knowledge about standard functions and it depends on solving equations. So nobody can have a blind faith in the results of this procedure. The same holds for the procedure `singular` that tries to calculate singularities of expressions. If it returns an expression containing `_N` or `_NN`, these symbols represent an integer or a natural number, respectively.

The procedure **residue** tries to compute the residue of a function of one variable at a given point.

8.13 Other facilities for series calculations

Facilities for working with infinite formal power series are offered by the shareware packages `PS` and `FPS` and by an old, rather unfriendly package `powseries`.

Expansion into Chebyshev polynomials is available as `numapprox[chebyshev]`, where the Chebyshev polynomials can be calculated by `orthopoly[T]`.

chapter 9

Numerical calculations with Maple

Symbolical manipulations are an essential part of using Maple, but powerful numerical facilities are available, too. That makes it easy to handle parts of a calculation with symbolical methods, so that the usual rounding problems when applying numerical procedures can be kept restricted to the remaining parts of the calculation.

This combination of symbolical and numerical facilities makes Maple a very flexible and efficient tool. However, it may be necessary, for specialized purposes, to use other numerical software. Maple offers facilities for using the programming languages Fortran and C and the numerical software system Matlab.

This chapter links up with section 2.8 on page 26 and section 2.9 on page 28 on floating-point numbers. Some special subjects on numerical approximations are discussed in connection with matching subjects in other chapters of this book.

9.1 Accuracy

The general approximation procedure in Maple is **evalf**:

```
> evalf( sqrt(3) , 50 );
```

$$1.7320508075688772935274463415058723669428052538104$$

```
> Int( sin(x^3) , x=0..Pi^(1/4) ); evalf( % , 20 );
```

$$\int_0^{\pi^{(1/4)}} \sin\left(x^3\right)\, dx$$

$$.53996849568350107601$$

Observe the use of `Int` instead of `int` in order to prevent Maple from starting symbolical calculation of the integral. The second argument to `evalf` is the number of digits to be used in the calculations, generally, it is *not* the expected **accuracy** of the result. For instance, in the following case, the last two digits are not correct:

```
> evalf((1+sqrt(5))^100,5);
```

$$.10051\, 10^{52}$$

In most cases it is easy to find more accurate results simply by increasing the number of digits:

```
> evalf((1+sqrt(5))^100,10);
```

$$.1004069091\,10^{52}$$

If no second argument is given to `evalf`, the value of the name **Digits** is used instead. The value of `Digits` is 10 at the start-up of Maple.

If `evalf` has to handle an expression that contains floating-point numbers itself, be cautious. For instance,

```
> evalf( 0.3 ^ 100 , 50 );
```

$$.5153775207\,10^{-52}$$

The reason for rounding to 10 digits, instead of 50, is that 0.3^{100} is calculated immediately by automatic simplification before the procedure `evalf` comes into action, according to the rule of evaluation of arguments. Since `Digits` is referring to 10, this automatic simplification yields a 10-digit result. To get the desired number of digits, use `3/10` instead of `0.3`:

```
> evalf( (3/10)^100 , 50 );
```

$$.51537752073201133103646112976562127270210752200100\,10^{-52}$$

In more complicated cases, you can convert floating-point elements occurring in expressions into rationals with **convert(,rational)**. See section 12.7 on page 156.

The procedure `evalf` can handle many built-in procedures. More specific information is found in this book where the relevant procedures are discussed and in section 2.8 on page 26.

9.2 Speeding up by optimizing

Suppose that numerical values of an expression are to be calculated for many values of `a`, `x`, and `y`, for instance, for the purpose of numerical integration. Then calculations may be speeded up by manipulating that expression before. Consider the expression

```
> p0 := 9*a^3*x^5*exp(y)^2-6*a^3*x^4*exp(y)+
>   a^3*x^3+27*exp(y)^3*a^2*x^4-18*exp(y)^2*a^2*x^3+
>   3*exp(y)*a^2*x^2+27*exp(y)^4*a*x^3-
>   18*exp(y)^3*a*x^2+3*exp(y)^2*a*x+9*exp(y)^5*x^2-
>   6*exp(y)^4*x+exp(y)^3-x-1;
```

$$p0 := 9\,a^3\,x^5\,(\mathrm{e}^y)^2 - 6\,a^3\,x^4\,\mathrm{e}^y + a^3\,x^3 + 27\,(\mathrm{e}^y)^3\,a^2\,x^4 - 18\,(\mathrm{e}^y)^2\,a^2\,x^3 + 3\,\mathrm{e}^y\,a^2\,x^2 + 27\,(\mathrm{e}^y)^4\,a\,x^3 - 18\,(\mathrm{e}^y)^3\,a\,x^2 + 3\,(\mathrm{e}^y)^2\,a\,x + 9\,(\mathrm{e}^y)^5\,x^2 - 6\,(\mathrm{e}^y)^4\,x + (\mathrm{e}^y)^3 - x - 1$$

The number of multiplications can be reduced with **convert(,horner)**:

```
> p1 := convert( % , horner );
```

$$p1 := -1 + (\mathrm{e}^y)^3 + \left(-6\,(\mathrm{e}^y)^4 - 1 + 9\,(\mathrm{e}^y)^5\,x\right)x + \left(\left(3\,(\mathrm{e}^y)^2 + \left(-18\,(\mathrm{e}^y)^3 + 27\,(\mathrm{e}^y)^4\,x\right)x\right)x + \left(\left(3\,\mathrm{e}^y + \left(-18\,(\mathrm{e}^y)^2 + 27\,(\mathrm{e}^y)^3\,x\right)x\right)x^2 + \left(1 + \left(-6\,\mathrm{e}^y + 9\,(\mathrm{e}^y)^2\,x\right)x\right)x^3\,a\right)a\right)a$$

This has reduced the number of multiplications from 76 to 45. However, in the last expression e^y is to be calculated 11 times and several powers of this are to be calculated. It is possible to make a further improvement with the procedure **codegen[optimize]**, which yields an efficient sequence of calculations. (In releases before Maple V.5, the package `codegen` is not available, use `optimize` after the command `readlib(optimize)`.)

```
> p2 := codegen[optimize]( %% );
```

$$p2 := t1 = a^2,\ t2 = t1\,a,\ t3 = x^2,\ t4 = t3^2,\ t7 = \mathrm{e}^y,\ t8 = t7^2,\ t14 = t3\,x,\ t16 = t8\,t7,\ t26 = t8^2,\ t41 = 9\,t2\,t4\,x\,t8 - 6\,t2\,t4\,t7 + t2\,t14 + 27\,t16\,t1\,t4 - 18\,t8\,t1\,t14 + 3\,t7\,t1\,t3 + 27\,t26\,a\,t14 - 18\,t16\,a\,t3 + 3\,t8\,a\,x + 9\,t26\,t7\,t3 - 6\,t26\,x + t16 - x - 1$$

The result is a sequence of equations, indicating the calculation steps, the last yielding the result. The procedure `optimize` uses the names `t1`, `t2`, `t3` ..., supposing that these names have not been assigned a value.
You can make `optimize` do still more than standard by adding the option `tryhard`.

The easiest way of using this sequence of calculations is by incorporating it into a procedure. This can be done with the procedure **codegen[makeproc]**. (In releases before Maple V.5, use `` `optimize/makeproc` ``; don't forget the back quotes.)

```
> f2 := codegen[makeproc]( [p2] , [a,x,y] );
```

```
f2 := proc(a, x, y)
local t1, t2, t41, t8, t4, t3, t7, t14, t16, t26;
    t1 := a^2;
    t2 := t1 * a;
    t3 := x^2;
    t4 := t3^2;
    t7 := exp(y);
    t8 := t7^2;
    t14 := t3 * x;
    t16 := t8 * t7;
    t26 := t8^2;
    t41 := 9 * t2 * t4 * x * t8 - 6 * t2 * t4 * t7 + t2 *
    t14 + 27 * t16 * t1 * t4 - 18 * t8 * t1 * t14 + 3 * t7 *
    t1 * t3 + 27 * t26 * a * t14 - 18 * t16 * a * t3 + 3 *
    t8 * a * x + 9 * t26 * t7 * t3 - 6 * t26 * x + t16 - x -
    1
end
```

This procedure can now be used as a function that calculates values of the expression p0 quickly:

```
> f2( 0.33333 , -1.2345 , 3.1417 );
```

$$.8831624520\,10^{8}$$

Such a procedure can be differentiated with the procedure **D**. See section 6.9 on page 79. Moreover, when the original expression contains only one variable, and consequently the corresponding procedure has just one argument, this procedure can be **numerically integrated**:

```
> evalf(Int(f2(1.35,t,t^3/(t^2+1)),t=0..1));
```

$$47.00305394$$

(In releases before Maple V.5, use '`evalf/int`' for the calculation above.) Even for the relatively small expression in the present example, this last calculation is considerably (about twice) faster than numerical integration of the original expression p0.

The facilities discussed here can be quite efficient for more complicated calculations. The reduction in work can be shown by the procedure **cost**:

```
> codegen[cost](p1);
```

$$13\,additions + 46\,multiplications + 11\,functions$$

```
> codegen[cost](p2);
```

$$39\,multiplications + 10\,assignments + functions + 13\,additions$$

In the last result "*functions*" is to be read as "1 function call".

For rational expressions and function calls, efficiency can be improved by approximating the expression with a continued fraction, using the procedure **convert(,confrac,<var>)**. This procedure calculates a series approximation to the given expression with respect to the variable given as a third argument, and then converts this into a truncated continued fraction (**Pade approximation**):

```
> convert( exp(t) , confrac , t );
```

$$1+\cfrac{t}{1+\cfrac{t}{-2+\cfrac{t}{-3+\cfrac{t}{2+\cfrac{t}{5+\cfrac{t}{-2-\frac{1}{7}t}}}}}}$$

The value of `Order`, or an additional fourth argument, determines the accuracy of this approximation. See section 8.2 on page 109.

After conversion of an expression to a continued fraction, you can apply `codegen[optimize]` as well.

9.3 Speeding up with floating-point facilities of the system

Maple can use the standard facilities of the system for floating-point calculations by the procedure **evalhf**:

```
> evalhf( exp(3) );
```

$$20.08553692318767$$

Generally, this procedure is considerably faster than `evalf`, but the number of digits is determined by the properties of the computer system, using "double precision". This procedure is also used in the context of `hfarrays`.

9.4 Some special procedures

The `numapprox` package contains several approximation procedures, for instance `infnorm`, which estimates the **maximum absolute value of a function** on an interval, and `minimax`, which tries to construct an optimal **approximation to an expression by a rational expression**. The share library offers more approximation procedures; see section 5.6 on page 69. Moreover, Maple contains procedures for **Fast Fourier Transform** (`FFT` and its inverse `iFFT`).

9.5 Using Fortran and C in combination with Maple

In Maple V.5 you can translate Maple procedures into Fortran or C routines, if the original routine does not contain untranslatable parts. Let's translate the procedure `f2` from the above into C:

```
> codegen[C](f2);
#include <math.h>
double f2(a,x,y)
double a;
double x;
double y;
{
  double t1;
  double t2;
  double t41;
  double t8;
  double t4;
  double t3;
  double t7;
  double t14;
  double t16;
  double t26;
  {
    t1 = a*a;
    t2 = t1*a;
    t3 = x*x;
    t4 = t3*t3;
    t7 = exp(y);
    t8 = t7*t7;
    t14 = t3*x;
    t16 = t8*t7;
    t26 = t8*t8;
    t41 = 9.0*t2*t4*x*t8-6.0*t2*t4*t7+t2*t14+27.0*t16*t1*t4-
18.0*t8*t1*t14+3.0*t7*t1*t3+27.0*t26*a*t14-18.0*t16*a*t3+
3.0*t8*a*x+9.0*t26*t7*t3-6.0*t26*x+t16-x-1.0;
    return(t41);
  }
}
```

There are several options possible for these commands; for instance the option `filename =` causes the output to be appended to the named file.
In previous releases, only a list of equations could be translated by fortran and C.

9.6 Data files

For reading in a data file of numbers, you can use the procedure `readdata`. This is discussed in section 18.21 on page 274. For more complicated cases, `readline` and `sscanf` are available, preferably in combination with an `hfarray`. For simple output to files, you can use `writeto` and `appendto` or `write` and `writeln`; for `hfarrays`, use `printf`. More information about these features can be found in the on-line help.

chapter 10

Manipulating several objects at once

In Maple, several mathematical objects can be joined together in a sequence, a set, or a list. This chapter shows you how to handle these, in which ways they can be used, and how you can process several objects at once.
Besides these three constructions, Maple has tables and arrays. The last section of this chapter introduces tables. Arrays are more important in the common interactive use of Maple, especially when using matrices and vectors. This subject is discussed in Chapter 18, Vectors and matrices.

10.1 Creation of sequences, sets, and lists

Maple procedures often yield **sequences**, for instance,

```
> 4*x^6+(16-4*b)*x^5+(12*b-75)*x^4+(63-9*b)*x^3;
```

$$4x^6 + (16 - 4b)x^5 + (12b - 75)x^4 + (63 - 9b)x^3$$

```
> sol := solve(%,x);
```

$$sol := 0, 0, 0, \frac{3}{2}, \frac{3}{2}, -7 + b$$

Now `sol` is a sequence of six objects.

We can also create a sequence in a direct way. For instance,

```
> sqA := 111, 2^x, a^3, cos(4), 0;
```

$$sqA := 111, 2^x, a^3, \cos(4), 0$$

Now sqA is a sequence of five objects.

If we create a sequence of five objects by replacing the third element of `sqA` by the sequence `sol`

```
> sqB := 111, 2^x, sol, cos(4), 0;
```

$$sqB := 111, 2^x, 0, 0, 0, \frac{3}{2}, \frac{3}{2}, -7 + b, \cos(4), 0$$

we do not obtain a sequence of five objects, but the elements of `sol` are *merged* into the sequence and so we find a sequence of 10 objects. If we want to tie the elements of `sol` together as a whole, we can bundle them into a **set** or a `list`.

```
> setA := {sol};
```

$$setA := \{0,\ -7+b,\ \frac{3}{2}\}$$

In this example you can see that Maple conforms to the mathematical idea of a set: it is useless to denote an element more than once. Moreover, Maple renders the elements of a set in an order according to Maple's internal preferences. Here is an example:

```
> setA :={111, 2^x, a^3, cos(4), 0};
```

$$setA := \left\{0,\ 111,\ 2^x,\ \cos(4),\ a^3\right\}$$

Often, these characteristics of a set are not appropriate. Therefore, Maple has another structure: **list**. A list can be created by writing down a sequence between brackets:

```
> listA := [111, 2^x, a^3, cos(4), 0];
```

$$listA := \left[111,\ 2^x,\ a^3,\ \cos(4),\ 0\right]$$

```
> listB :=[ sol ];
```

$$listB := [0,\ 0,\ 0,\ \frac{3}{2},\ \frac{3}{2},\ -7+b]$$

These constructions can be combined as in the following example:

```
> 111, 2^x, [sol], cos(4), 0;
```

$$111,\ 2^x,\ [0,\ 0,\ 0,\ \frac{3}{2},\ \frac{3}{2},\ -7+b],\ \cos(4),\ 0$$

Here the elements of `sol` are tied together in a list, which is used as one element of the sequence containing five elements in all.

The **void sequence** (the sequence with no elements) is denoted by **NULL**, and the **void list** and **void set** are denoted by [] and { }.

You can convert a set or a list to a sequence with **op**:

```
> op(listA);
```

$$111,\ 2^x,\ a^3,\ \cos(4),\ 0$$

The procedure op cannot be applied to a sequence:

```
> op(%);
Error, wrong number (or type) of parameters in function op
```

The **number of elements** of a list or set can be calculated by **nops**:

```
> nops( [ p, q, r] );
```

The number of elements of a sequence can be calculated by counting the elements of the corresponding list:

```
> nops( [%%] );
```

$$5$$

The number of elements in a sequence, list, or set can be up to $2^{17} - 2 = 131070$. If there are more, you get a message:

```
> a $ 131071:
Error, object too large
```

10.2 Selecting elements of sequences, sets, and lists

Selecting from a set or a list is possible by using op:

```
> op( 2, setA );
```

$$111$$

```
> op( 3, listA );
```

$$a^3$$

You can use a range to obtain a subsequence:

```
> op( 2..4, listA);
```

$$2^x,\ a^3,\ \cos(4)$$

Indexing can be used for lists and sets as well:

```
> listB[6];
```

$$-7 + b$$

It is not possible to change a sequence or set by assigning to an indexed element, but it is possible for a list (from release V.4):

```
> listA[1]:=10;
```

$$listA_1 := 10$$

```
> listA;
```

$$[10,\ 2^x,\ a^3,\ \cos(4),\ 0]$$

Selecting from a sequence is possible by indexing, using square brackets:

```
> sol[4];
```

$$\frac{3}{2}$$

You can obtain a subsequence by indexing with a range:

```
> sol[3..5];
```

$$0, \frac{3}{2}, \frac{3}{2}$$

Remember:

> The procedure op cannot be applied to sequences.

That is because op tries to convert its argument into a sequence:

```
> op( 1 , sol );
```

```
Error, wrong number (or type) of parameters in function op
```

Searching and selecting special elements from a list or a sequence is discussed in sections at the end of this chapter.

10.3 Applying a procedure to several objects at once

It is often efficient to process several objects at once. For instance, suppose that a parameterized equation has been solved like this:

```
> equ := x^4+5*x^3*a-4*x^3*b-6*a^2*x^2-24*a*x^2*b+
>        36*a*x*b+14*a^2*x+16*x*b^2-16*b^2-16*b*a-4*a^2;
```

$$equ := x^4+5x^3a-4x^3b-6a^2x^2-24ax^2b+36axb+14a^2x+16xb^2-16b^2-16ba-4a^2$$

```
> solutions := solve(%,x);
```

$$solutions := -3a+\sqrt{9a^2+4b+2a},\ -3a-\sqrt{9a^2+4b+2a},\ 2b+\frac{1}{2}a+\frac{1}{2}\sqrt{16b^2+8ba+a^2-16b-8a},\ 2b+\frac{1}{2}a-\frac{1}{2}\sqrt{16b^2+8ba+a^2-16b-8a}$$

Now you may want to substitute values for a and b in this sequence of four expressions and try to do so as follows:

```
> subs( a=6 , b = 3 , solutions );
```

```
Error, wrong number (or type) of parameters in function subs
```

The cause of this error is that the name `solutions` refers to a sequence of four expressions. So the arguments `(a=6 , b=3 , solutions)` are evaluated to a sequence of *six* objects: two equations and four algebraic expressions. But `subs` expects as its arguments one or more equations and at the end just one object to which the substitutions have to be applied.

To substitute in all the solutions at once, we have to tie them together into a list (or a set), in order to give it as one last argument to `subs`:

```
> subs( a=6 , b = 3 , [solutions] );
```

$$\left[-18+\sqrt{348},\ -18-\sqrt{348},\ 9+\frac{1}{2}\sqrt{228},\ 9-\frac{1}{2}\sqrt{228}\right]$$

Many procedures can handle lists and sets, for instance `diff`, `evalf`, `expand`, `normal`, and `simplify`:

```
> simplify( % );
```

$$\left[-18+2\sqrt{87},\ -18-2\sqrt{87},\ 9+\sqrt{57},\ 9-\sqrt{57}\right]$$

Generally, standard mathematical functions (see Chapter 6, *Creating and using mathematical functions*) cannot be applied to a list or a set:

```
> listC := [ 5*a^2*ln(3) , 0 , ln(b) ];
```

$$listC := \left[5\,a^2\ln(3),\ 0,\ \ln(b)\right]$$

```
> exp( % );
```

$$\exp\left(\left[5\,a^2\ln(3),\ 0,\ \ln(b)\right]\right)$$

The procedure `exp` cannot use a list as an exponent to e, but it does not protest against this input and yields the exp of this list unevaluated, whatever that may mean. For such a case, use the procedure **map**:

```
> map( exp , listC );
```

$$\left[\mathbf{e}^{\left(5\,a^2\ln(3)\right)},\ 1,\ b\right]$$

The procedure `map` has applied `ln` to all the elements of `listC`, preserving the list structure. If the elements of this list should be simplified by `simplify`, you don't need `map`; `simplify` can be applied to a list directly:

```
> simplify( % );
```

$$\left[243^{(a^2)},\ 1,\ b\right]$$

The procedure map applies the first argument, which should be a procedure, to the operands of the second argument:

```
> map( x->x^2 , [a,b,c] );
```

$$[a^2, b^2, c^2]$$

Note that Maple chooses the order of elements in a set. So if you apply map to a set, the order in the result may not correspond to the order in the original set:

```
> map( x->x^2 , {b,a,c} );
```

$$\{a^2, b^2, c^2\}$$

It is even possible to use map in combination with a sum. Here the procedure is applied to the operands of the sum, the terms. The result is the sum of the procedure values:

```
> map( x->x^2 , a+b+c );
```

$$a^2 + b^2 + c^2$$

Sometimes a procedure expects more than one argument, for instance iquo, which calculates the integer quotient of two integers:

```
> iquo(6020,6);
```

$$1003$$

The procedure map can be used for such procedures, too. As an object for demonstrating this, we generate the first 18 prime numbers, using the procedure seq, explained in the next section.

```
> listD := [ seq( ithprime(i) , i=1..18 ) ];
```

$$listD := [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61]$$

The integer quotients of the elements of listD divided by 6 can be calculated in two ways:

```
> map( n->iquo(n,6) , listD );
```

$$[0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 5, 6, 6, 7, 7, 8, 9, 10]$$

```
> map( iquo , listD , 6 );
```

$$[0, 0, 0, 1, 1, 2, 2, 3, 3, 4, 5, 6, 6, 7, 7, 8, 9, 10]$$

In the last command the procedure map applies iquo to pairs consisting of an element of listD and the number 6. You can enter the second argument of the procedure as the third argument to map. And you may go on with more arguments.

Now suppose we want to calculate the integer quotients of 6020, dividing this subsequently by the elements of listD. Then we cannot use the previously explained

facility of map with additional arguments. But **map2** can do the job:

```
> map2( iquo, 6020 , listD );
```

$$[3010, 2006, 1204, 860, 547, 463, 354, 316, 261, 207, 194, 162, 146, 140, 128, 113, 102, 98]$$

The procedure map2 applies iquo to pairs consisting of the number 6020 and an element of listD.
It is also possible to multiply each element of a list with a number (not a symbolic number) in a direct way:

```
> 2*[a,b,c];
```

$$[2a, 2b, 2c]$$

10.4 Finding a special element in a set or a list

You can ask if a special object occurs in a set or a list by **member**:

```
> member( 3/2 , listB );
```

$$true$$

You can even ask for the first position where this element occurs, by using a name as a third argument. This construction is explained in section 5.4 on page 67 for the procedure iquo:

```
> member( 3/2 , listB , 'pos' );
```

$$true$$

```
> pos;
```

$$4$$

10.5 Finding the minimal or the maximal element

The **minimal** and **maximal elements** of a sequence of real numbers, possibly containing symbolic elements, are calculated by **min** and **max**:

```
> min( sqrt(10), Pi, 22/7 );
```

$$\pi$$

In the following case, Maple assumes a to be a real number:

```
> max( a + 5, a+6 , 2*a+6);
```

$$\max(2\,a+6, a+6)$$

Here Maple cannot decide fully, but if we indicate that `a` is a positive number, Maple can do the full job:

```
> assume(a,positive);
> max( a + 5, a+6 , 2*a+6);
```

$$2\,a\sim +6$$

10.6 Selecting the elements that satisfy a special condition

You can select the elements of a list or a set that satisfy a test with the procedure **select**. For instance, you can look for all prime numbers between 100,000 and 100,100 as follows:

```
> [ seq( i , i=100000..100100 ) ]:
> select( isprime , % );
```

$$[100003, 100019, 100043, 100049, 100057, 100069]$$

The first argument to `select` must be a test procedure (yielding `true` or `false`). In the previous case, we have used the procedure `isprime`, which tests if its argument is a prime number and yields `true` or `false` as the result. The second argument of `select` should be a list or a set (or a sum or a product).

You can use a test procedure that uses options, too. For instance, let's select all the elements that are of type numeric from a given set:

```
> { 3/2 , sqrt(2) , x + 7 , 0 };
```

$$\left\{0, x+7, \frac{3}{2}, \sqrt{2}\right\}$$

```
> select( type , % , numeric );
```

$$\{0, \frac{3}{2}\}$$

Here the second argument to the procedure `type` is given as the third argument to `select`. For information about the procedure `type`, consult the on-line help on this subject and section A.1 on page 275.

The test procedure for `select` may be constructed with the aid of the arrow. For instance, here is a numerical test function that yields `true` if and only if the approximation by `evalf` of its argument x is a positive number smaller than 1:

```
> test := x->type(x,numeric) and x>0 and x<1;
```

$$test := x \rightarrow false$$

The procedure `test` checks first if its argument is a numeric object; if not, Maple recognizes that the result is false and so will not try to check if the argument lies in the range -1 ... 1.

Now we can select the solutions of an equation that lie in this range:

```
> fsolve( 16*x^4-60*x^3+36*x^2+4*x-4 , x );
```

$$-.3058273664, .4089006016, .6720007242, 2.974926041$$

```
> select( test , [%] );
```

$$[.4089006016, .6720007242]$$

If you would have a list of numbers that contains complex numbers, you would have to select the real ones first:

```
> [.5+3*I,.7,-Pi];
```

$$[.5 + 3.I, .7, -\pi]$$

```
> select(x->(x=evalc(Re(x))),%);
```

$$[.7, -\pi]$$

In this simple case, we can trust the condition `x=evalc`. In more complicated cases, `evalc` might yield an imaginary component that equals zero mathematically (see for instance section 15.11 on page 211). In such a case, you might gamble with numerical approximations using **fnormal** (see section 2.8 on page 26) to avoid problems with small imaginary components:

```
> select(x->(fnormal(evalf(x))=evalc(Re(evalf(x)))),%):
```

The counterpart of `select` is **remove**, which is used in the same way, but removes elements from a set or a list.

10.7 Generating sequences as values of a function or an expression

A useful tool for generating sequences from an expression is the procedure **seq**:

```
> seq( n! , n=1..10 );
```

$$1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800$$

The name `n` gets the integer values in the **range** from 1 to 10, denoted as `1 .. 10`, and `n!` is calculated each time, yielding a sequence of values.

Here is an application: let's calculate the function values at the (approximated) zeros of the derivative of a function:

```
> f:=x->x^5-50*x^3-25*x^2+15*x-1;
```

$$f := x \to x^5 - 50\,x^3 - 25\,x^2 + 15\,x - 1$$

```
> fsolve( D(f)(x) , x );
```

$$-5.291800567,\ -.5277249269,\ .1908553686,\ 5.628670126$$

Now we want to see the function values at each of these points:

```
> seq( f(%[n]) , n = 1..nops([%]) );
```

$$2479.193139,\ -8.570742823,\ .604836765,\ -3975.227232$$

From release V.4, the counting parameter of `seq` is a formal parameter, so it is not assigned to by `seq` and it is not important if it is assigned to previously.

10.8 Manipulating sequences, sets, and lists

The sum or the product of the elements of a set or a list can be calculated by `convert( ,'+')` or `convert( ,'*')`:

```
> convert( listA , '+' );
```

$$10 + 2^x + a\sim^3 + \cos(4)$$

For sets, the operators **union**, **intersect**, and **minus** are available:

```
> {1,3,5} union {2,3,4};
```

$$\{1, 2, 3, 4, 5\}$$

```
> % intersect {3,4,5,6,7};
```

$$\{3, 4, 5\}$$

```
> % minus {4,6};
```

$$\{3, 5\}$$

These operators can also be used as procedures, but then the name should be enclosed in a pair of back quotes; see section 3.8 on page 40. Using such an operator as a procedure can be convenient for handling a sequence of sets. An example can be found in section E.6 on page 305.

Here are two lists:

```
> listA; listB;
```

$$[10,\ 2^x,\ a\sim^3,\ \cos(4),\ 0]$$

$$[0,\ 0,\ 0,\ \frac{3}{2},\ \frac{3}{2},\ -7 + b]$$

Let's join them together into one list. This can be done as follows:

```
> [ op( %% ) , op( % ) ];
```

$$\left[10,\ 2^x,\ a\sim^3,\ \cos(4),\ 0,\ 0,\ 0,\ 0,\ \frac{3}{2},\ \frac{3}{2},\ -7+b\right]$$

Exchanging an element of a list for something else is easy with the aid of `subsop`. For instance, the second element of `listA` can be exchanged for `X` as follows:

```
> listA;
```

$$[10,\ 2^x,\ a\sim^3,\ \cos(4),\ 0]$$

```
> subsop( 2=X , % );
```

$$[10,\ X,\ a\sim^3,\ \cos(4),\ 0]$$

Note that *this action does not change the value of* `listA`:

```
> listA;
```

$$[10,\ 2^x,\ a\sim^3,\ \cos(4),\ 0]$$

To change the value of `listA`, you must assign the new value:

```
> listA[2] := X;
```

$$listA_2 := X$$

The procedure `subsop` can also be used for omitting an element:

```
> subsop( 3=NULL , listA );
```

$$[10,\ X,\ \cos(4),\ 0]$$

Here `subsop` has created a new list by exchanging the third element of the previous list by the void sequence, which omits the third element effectively.

10.9 Conversions between sequences, sets, and lists

In the previous section, you saw how a sequence could be converted into a list or a set by enclosing it between brackets (`[ ]`) for a list or braces (`{ }`) for a set. For the opposite direction, use the procedure **op**:

```
> listA;
```

$$[10,\ X,\ a\sim^3,\ \cos(4),\ 0]$$

```
> op( % );
```

$$10,\ X,\ a\sim^3,\ \cos(4),\ 0$$

```
> op( setA );
```

$$0,\ 111,\ 2^x,\ \cos(4),\ a\sim^3$$

The procedure op is a general tool that can be applied to any Maple object that is not itself a sequence; it yields a sequence of the operands of its argument.

Conversion between lists and sets is achieved by going through a sequence:

```
> listB;
```

$$[0,\ 0,\ 0,\ \frac{3}{2},\ \frac{3}{2},\ -7+b]$$

```
> { op( % ) };
```

$$\{0,\ -7+b,\ \frac{3}{2}\}$$

```
> [ op( % ) ];
```

$$[0,\ -7+b,\ \frac{3}{2}]$$

Observe that the last result is not equal to `listB`.

10.10 Tables

Names with an index are introduced in section B.3 on page 287.

```
> P[n];
```

$$P_5$$

If you assign something to such a name, you create a table.

```
> P[1]:=sqrt(3);
```

$$P_1 := \sqrt{3}$$

```
> P;
```

$$P$$

```
> eval(P);
table([
   1 = √3
   ])
```

In the second command the procedure `eval` is used in order to see the value of `P`. For tables, the same evaluation rule is used as for procedures. Therefore, the second command renders `P`; full evaluation is demanded with `eval`.
Such a table is very compliant; for instance:

```
> P[cos]:=sin;
```

$$P_{cos} := sin$$

```
> P[1,2,3,4,5]:=much;
```

$$P_{1,2,3,4,5} := much$$

```
> eval(P);
```

table([

$\cos = \sin$

$1 = \sqrt{3}$

$(1,2,3,4,5) = much$

])

You can use tables easily if you like. Maple itself uses many tables, such as the remember tables for procedures; see section D.1 on page 293.

chapter 11

Substitution and subexpressions

Substitution is an important tool in manipulating Maple expressions. It is not restricted to substituting something for a name, but can be used for many more purposes. There are several general tools for substitution. This chapter shows the possibilities and restrictions of the substitution tools in Maple.

11.1 Some examples of substitution

The obvious usage of substitution is as in the following example:

```
> expr := sin( a );
```

$$expr := \sin(a)$$

```
> subs( a = 7, expr );
```

$$\sin(7)$$

Remember that *the last action has not changed the value of* `expr`.

```
> expr;
```

$$\sin(a)$$

To change the value of `expr` to the last result, assign it.

The procedure **subs** can do much more. For instance, it is possible to replace a number with another expression.

```
> subs( 7=a+b , %% );
```

$$\sin(a+b)$$

Moreover, you can replace the name of a procedure with the name of another procedure, as in the following two examples:

```
> subs( sin=cos , % );
```

$$\cos(a+b)$$

```
> subs( cos=(x->x+3*x^2) , % );
```

$$\left(x \rightarrow x + 3\,x^2\right)(a+b)$$

You may wonder why Maple does not apply this simple function. The root of it is that *Maple does not evaluate the result found by* `subs` *automatically.*

When we offer this result again to Maple, it is evaluated.

```
> %;
```

$$a + b + 3\ (a + b)^2$$

11.2 A substitution that fails

Let's go on with the last result and now substitute something for a+b.

```
> subs( a+b=7 , % );
```

$$a + b + 147$$

You may wonder why the first occurrence of a+b is not exchanged for 7, whereas the second occurrence is exchanged. The cause is that Maple has not seen the first occurrence as a **subexpression.**

We can break down an expression into its **components**, called "**operands**" in Maple, by the procedure **op**.

```
> %%;
```

$$a + b + 3\ (a + b)^2$$

```
> op( % );
```

$$a,\ b,\ 3\ (a + b)^2$$

Maple interprets this expression as a sum of three terms. This is why Maple does not see the sum of the first two terms, a+b, as a subexpression. However, the third term contains a component a+b.

```
> op( %[3] );
```

$$3,\ (a + b)^2$$

```
> op( %[2] );
```

$$a + b,\ 2$$

In the last substitution Maple encountered a+b as a component of a component of the third component of the original expression, and substituted 7 only for this subexpression.

In this case, there is an easy trick to exchange all occurrences of a+b for 7: substitute 7-b for a.

```
> subs( a=7-b , a + b + 3 * ( a + b)^2 );
```

Remember:

> The procedure `subs` replaces only subexpressions of the given expression, generally.

The **subexpressions** of an expression, say '`expr`', are `expr` itself, all the components of `expr` together with all the components of these components, all the components of the components of the components, and so on.

11.3 Subexpressions of polynomials, substitution

When an expression is to be manipulated, it is often important to know how it is built from subexpressions. Expression structure can be analyzed with the procedure **op**. For instance, here is a case where the substitution command fails.

```
> pol := x^3 - 5*x^2 + 4*x - 3;
```

$$pol := x^3 - 5x^2 + 4x - 3$$

```
> subs( 5=t , pol );
```

$$x^3 - 5x^2 + 4x - 3$$

In order to find out why substitution did not work as intended, we analyze the structure of `pol`.

```
> whattype(pol);
```

$$+$$

```
> op(pol);
```

$$x^3,\ -5x^2,\ 4x,\ -3$$

We can select from this sequence the second element by indexing.

```
> %[2];
```

$$-5x^2$$

but we can also pick up the second subexpression by using op in another way.

```
> op(2,pol);
```

$$-5x^2$$

The procedure op gives the nth component of an expression with the command `op(n, )`.

```
> whattype(%);
```

*

```
> op(%%);
```

$$-5,\ x^2$$

Obviously the number `-5` is a subexpression of the original expression, so we can substitute 5 for `t` in `pol` by

```
> subs( -5=-t , pol );
```

$$x^3 - x^2\,t + 4\,x - 3$$

Let's analyze the first subexpression of `pol`.

```
> op(1,pol);
```

$$x^3$$

```
> whattype(%);
```

^

```
> op(%%);
```

$$x,\ 3$$

Here is the full listing of subexpressions of x^3-5x^2+4x-3, achieved by applying op several times.

$$x,\ 3,\ x^3,\ -5,\ x,\ 2,\ x^2,\ -5x^2,\ 4,\ x,\ 4x,\ -3,\ x^3 - 5x^2 + 4x - 3$$

By this analysis, you can picture

$$x^3 - 5\,x^2 + 4\,x - 3$$

as

sum(power(x,3) , product(-5 , power(x,2)) , product(4,x) , 3)

In fact, Maple stores such a polynomial in a slightly different way internally. This is evident when something is substituted for `1`.

```
> subs( 1=ONE, pol );
```

$$x^3\,ONE - 5\,x^2 + 4\,x - 3\,ONE$$

Therefore, never substitute something for `1` in a Maple expression.

The procedure `subs` behaves in an exotic way in other special cases, for instance,

```
> subs( a*b=p , t*a*b - 5*a*b );
```

$$t\,a\,b - 5\,p$$

Here `a*b` is not a subexpression according to the results of `op`, but substitution did work in the last term.

Experiment with subs yourself. You might predict the results of the following commands, then check them:

- `subs(a=t,-a*x);`
- `subs(-a=-t, -a*x);`
- `subs(5=t, -5*x);`
- `subs(-5=-t,-5*x);`
- `subs(5/6=t, 5/6*x);`
- `subs(a/b=t,a/b*x);`

In all cases, it is wise to analyze the subexpression structure by repeatedly applying the procedure op.

11.4 Subexpressions of rational expressions, substitution

Let's try a substitution in a rational expression.

```
> (a^2 - b)/(a^2*c);
```

$$\frac{a^2 - b}{a^2 c}$$

```
> subs( a^2=t , % );
```

$$\frac{t - b}{a^2 c}$$

This may not be what you expected, so let's analyze the original expression.

```
> op( %% );
```

$$a^2 - b,\ \frac{1}{a^2},\ \frac{1}{c}$$

A quotient is taken as just a product, in the same way a difference is taken as a sum by Maple. Now let's see how the factors of the denominator of this quotient are stored.

```
> %[2];
```

$$\frac{1}{a^2}$$

```
> whattype( % );
```

```
^
```

```
> op( %% );
```

$$a,\ -2$$

It is useful to know that

> Maple takes quotients as products,
> where each factor of the denominator is described
> as a power with a negative exponent.

However, **rational numbers** are dealt with differently.

```
> op( 5/8 ), whattype( 5/8 );
```

$$5,\ 8,\ fraction$$

It is not necessary to remember details on subexpression structure, such as the last one. It is sufficient to keep in mind that generally `subs` can do nothing other than substitute something for a subexpression, and that these subexpressions can be found with the aid of `op`. Skip over the next section if you are not interested in more details.

11.5 Subexpressions of unevaluated function calls

In the example in the first section of this chapter, `sin` was replaced with another name in the expression `sin(a+b)`. Let's find the operands of this expression.

```
> sin(a+b);
```

$$\sin(a+b)$$

```
> op( % );
```

$$a+b$$

This looks strange: `sin` is not an operand of this expression, but it can be replaced.

```
> subs( sin=cos , %% );
```

$$\cos(a+b)$$

However, `sin` really is an artificial operand: the zero-numbered operand.

```
> op( 0 , %%% );
```

$$sin$$

As such, it is recognized by `subs` as a subexpression. Here is another example:

```
> int(f(x),x);
```

$$\int \mathrm{f}(x)\,dx$$

```
> op( % );
```

$$\mathrm{f}(x),\ x$$

```
> op( 0 , %% );
```

$$int$$

11.6 The procedure eval

The procedure **eval** can be used in a similar way as subs (from release V.5), but with essential differences. Here are some examples where both do the same:

```
> sin(a);
```

$$\sin(a)$$

```
> eval( % , a=7 );
```

$$\sin(7)$$

```
> eval( % , 7=ln(5) );
```

$$\sin(\ln(5))$$

Observe the difference in syntax:

```
subs( x=y , expr )
```

versus

```
eval( expr , x=y )
```

In the following command we see a difference:

```
> eval( % , sin=exp );
```

$$5$$

```
> subs(sin=exp, %% );
```

$$\mathbf{e}^{\ln(5)}$$

As you could see earlier, after substitution with subs no evaluation takes place, but it does after substitution with eval.

The most important difference can be seen in combination with procedures such as int, diff, sum, etc.:

```
> int(f(x),x);
```

$$\int \mathrm{f}(x)\,dx$$

```
> eval(%,x=a);
```

$$\left(\int \mathrm{f}(x)\,dx\right)\Big|_{\{x=a\}}$$

Although x is a subexpression of `int(f(x),x)`, it is not replaced by a, as `subs` would have done, but `eval` interprets it as an expression in x that might be calculated in the future.

11.7 The procedures subs and eval—a survey

The main facts on the procedure `subs` are the following:

- before `subs` and `eval` come into action, the arguments are evaluated, as usual in procedures
- usually, `subs` and `eval` can only replace subexpressions of an expression; a subexpression of an expression is the expression itself, its operands or components (to be determined with the aid of the procedure op) or an operand of an operand, and so on
- `subs` replaces *all* occurrences of the given subexpression are replaced in the given expression, but `eval` respects such things as the integration variable in an unevaluated call to `int`
- after this replacement `subs` does not evaluate the resulting expression, (only elementary automatic simplifications are executed), but `eval` does
- `subs` and `eval` do not change references (values) of names

11.8 More than one substitution at once

It is possible to execute more than one substitution at once. For instance,

```
> plin := 2*x - 10*y;
```

$$plin := 2x - 10y$$

```
> subs( x=a , y=b , plin );
```

$$2a - 10b$$

But if you try to switch x and y in the same way, it fails:

```
> subs(x=y , y=x , plin);
```

$$-8\,x$$

The procedure subs has executed the substitutions successively. It is possible to do these two **substitutions simultaneously** as follows:

```
> subs( [x=y,y=x] , plin );
```

$$2\,y - 10\,x$$

For eval it is also possible to execute more than one substitution at once, but only simultaneously, and square brackets (or braces) are compulsory.

```
> eval( plin , [x=y,y=x] );
```

$$2\,y - 10\,x$$

11.9 The procedure PDEtools[dchange] for changing variables

The procedure PDEtools[dchange] is meant to be a general tool for changing variables (from release V.5; in earlier releases DEtools[Dchangevar] and student[changevar] are predecessors of this more powerful procedure). Examples:

```
> PDEtools[dchange](x=t+5,sin(x)-x^2);
```

$$\sin(t+5) - (t+5)^2$$

The procedure DEtools[dchange] cannot substitute objects for general subexpressions, but only for variables; it is meant for such cases as:

```
> PDEtools[dchange](x=t+5,Int(sin(x),x=a..b));
```

$$\int_{a-5}^{b-5} \sin(t+5)\,dt$$

In the following, the command cannot be interpreted uniquely, so Maple issues an error:

```
> PDEtools[dchange](x=t+u,Int(sin(x),x));
Error, Missing a list with the new variables
```

We have to indicate, with a third argument, what will be the new variable instead of x:

```
> PDEtools[dchange](x=t+u,Int(sin(x),x),[t]);
```

$$\int \sin(t+u)\,dt$$

Square brackets are compulsory for this argument.

The following looks strange:

```
> PDEtools[dchange](x=t+u,Int(f(x),x),[t]);
```

$$\int \mathrm{f}(t,u)\,dt$$

What has happened is that Maple has changed the meaning of `f` at the same time. The reason for this is the main object of `PDEtools[dchange]`: it is to be used for differential equations, where `f` could be an unknown function to be found. However, this can be changed with another option:

```
> PDEtools[dchange](x=t+u,Int(f(x),x),[t],known=f);
```

$$\int \mathrm{f}(t+u)\,dt$$

More on this procedure can be found in Chapter 17, *Solving differential equations*.

11.10 Substitution of algebraic subexpressions

The following substitution fails:

```
> a^2*x*b + a*b^3 - 3*a^5*b^5*c;
```

$$a^2\,x\,b + a\,b^3 - 3\,a^5\,b^5\,c$$

```
> subs(a^2*b=NEW,%);
```

$$a^2\,x\,b + a\,b^3 - 3\,a^5\,b^5\,c$$

The procedure `subs` cannot do what the user seems to want in the last command because $a^2\,b$ is not a Maple subexpression of the given expression. But it is an algebraic subexpression. For substitution of algebraic subexpressions the procedure **algsubs** can be used. (In previous releases use `student[changevar]` and `asubs`.)

```
> algsubs(a^2*b=NEW,%%);
```

$$\left(-3\,c\,NEW^2 + 1\right) b^3\,a + x\,NEW$$

The same result can be achieved with a more advanced tool, simplification to side relations, explained in the next section.

You might use a variant of the previously mentioned trick of substituting `NEW/a^2` for `b`, but the result may not be desirable:

```
> subs( b=NEW/a^2 , %%% );
```

$$x\,NEW + \frac{NEW^3}{a^5} - 3\,\frac{NEW^5\,c}{a^5}$$

Here is another example:

```
> x+y-1;
```

$$x + y - 1$$

```
> algsubs( x-1 = t^2 , % );
```

$$t^2 + y$$

A drawback of this procedure is that it must choose what to do in less basic cases, sometimes making the use of it complicated (see the on-line help for it). A clearer approach can be found in the next section.

11.11 Applying side relations

Sometimes, when substitution falls short, you can use simplification to **side relations**. For example, let's take the expression

```
> a*x^2+b*x^3+c*x^4+d*x^5;
```

$$a\,x^2 + b\,x^3 + c\,x^4 + d\,x^5$$

Suppose that you want to substitute $p\,y$ for x^2. Substitution with the procedure `subs` only changes the first term. In order to change the others as well you can issue

```
> simplify( % , {x^2 = p*y}, [x] );
```

$$a\,p\,y + c\,p^2\,y^2 + \left(b\,p\,y + d\,p^2\,y^2\right) x$$

The last argument `[x]` asks for reduction of powers of x. In this case, it could have been omitted, in fact.

Here is another example. Suppose that we have two relations:

```
> rels := {x^2 = p*y , y^2 = q};
```

$$rels := \left\{ y^2 = q,\ x^2 = p\,y \right\}$$

and that we want to reduce the following expression to these two relations.

```
>  expand( (x^3+y*x-y)^2 );
```

$$x^6 + 2\,x^4\,y - 2\,x^3\,y + y^2\,x^2 - 2\,y^2\,x + y^2$$

That can be done with the same trick:

```
> simplify( % , rels, [x,y] );
```

$$q + \left(-2\,q - 2\,p\,q\right) x + \left(p\,q + 2\,p^2\,q + p^3\,q\right) y$$

In the result you don't see x^2: this could be eliminated with the relation $x^2 = p * y$ and it is eliminated because `[x,y]` indicates that in the first place `x` should be eliminated as far as possible, and then `y`.
If we enter `[y,x]` as the last argument, we get:

```
> simplify( %% , rels, [y,x] );
```

$$q + (-2q - 2pq)\,x + (q + 2pq + p^2 q)\,x^2$$

Here y is eliminated and x is reduced to maximal degree 2.
The procedure `simplify` with side relations is based on the Buchberger algorithm in Gröbner basis theory. Essentially, this aims at reduction of degrees of terms.

The present method is not restricted to polynomials. For instance,

```
> a*exp(y)+a^2*(2*x*y-3*y*z)+x*y*z;
```

$$a\,\mathrm{e}^y + a^2\,(2\,y\,x - 3\,y\,z) + x\,y\,z$$

We will convert this into a complicated expression, to be used as an example for simplifying to side relations.

```
> a:=exp(x)-y^2*z+1;
```

$$a := \mathrm{e}^x - y^2\,z + 1$$

```
> expand(%%):
> combine(%,exp);
```

$$\mathrm{e}^{(y+x)} - \mathrm{e}^y\,y^2\,z + \mathrm{e}^y + 2\,\mathrm{e}^{(2\,x)}\,y\,x - 3\,\mathrm{e}^{(2\,x)}\,y\,z - 4\,\mathrm{e}^x\,y^3\,z\,x + 6\,\mathrm{e}^x\,y^3\,z^2 + 4\,\mathrm{e}^x\,y\,x - 6\,\mathrm{e}^x\,y\,z + 2\,y^5\,z^2\,x - 3\,y^5\,z^3 - 4\,y^3\,z\,x + 6\,y^3\,z^2 + 2\,y\,x - 3\,y\,z + x\,y\,z$$

Now let's ask Maple to reduce the previous expression to the relation $a = 0$, that is to say, to $\mathrm{e}^x - y^2 z + 1 = 0$, eliminating e^x as far as possible.

```
> simplify( % , {a=0} , [exp(x)] );
```

$$x\,y\,z$$

More examples can be found in the Chapters 13, 14, and 15.

11.12 Finding the structure and subexpressions of large expressions

If an expression is too large to readily see what its structure is, use **whattype** in order to see if it is a sum, a product, a power, an unevaluated function call, or something else. See section A.1 on page 275. The **number of operands** of an expression can be found with **nops**. With the aid of the procedure **has** you can find out if one expression is a subexpression of another expression.

```
> has( factor(x^1000-1) , x^12 );
```

$$true$$

11.13 Selecting suboperands

Any level of components can be found with op. From release V.4 you can access suboperands with op directly. For instance

```
> cos(x) + (3 - 5*x*sin(x-t))^2;
```

$$\cos(x) + (3 - 5\,x\,\sin(x - t))^2$$

```
> op( [2,1,2,3] , % );
```

$$\sin(x - t)$$

The first argument of op may be a list of numbers. Here we select:

- the second operand of the given expression:

$$(3 - 5\,x\,\sin(x - t))^2$$

- the first operand of this:

$$3 - 5\,x\,\sin(x - t)$$

- the second operand of this:

$$-5\,x\,\sin(x - t)$$

- the third operand of this:

$$\sin(x - t)$$

11.14 Substituting something for one component of an expression

If an operand of an expression is to be exchanged for something else without changing any other component, you can use **subsop**. For instance,

```
> 3*x^2 + 5*x*y^3 - 4*x^2*y^2 + x^6*s*y;
```

$$3\,x^2 + 5\,x\,y^3 - 4\,x^2\,y^2 + e\,x^6\,s\,y$$

```
> subsop(3=WHOOPS, % );
```

$$3\,x^2 + 5\,x\,y^3 + WHOOPS + e\,x^6\,s\,y$$

The previous command asks Maple to exchange the third component for `WHOOPS`.

By an index list, you can replace suboperands as well. The selection is the same as with op. For instance, in order to replace in the last term x^6 by a you can issue

```
> subsop( [3,2]=a , % );
```

$$3\,x^2 + 5\,x\,y^3 + WHOOPS + e\,a\,s\,y$$

chapter 12

Manipulating and converting numbers

In Chapter 2, Numbers and algebraic operators, *some basic manipulations for numbers are discussed:* `expand`, `evalc`, *and* `simplify( ,radical)`. *The present chapter is a sequel to that chapter, discussing more specialized manipulations.*

12.1 Real and imaginary parts of a complex number

The procedure **evalc** converts a complex number to the form $a+bI$, where a and b are real numbers, supposing that all variables concerned can be assumed to be real numbers. The same procedure can be used to take the real and imaginary parts of a complex expression separately as follows:

```
> a + b*I;
```

$$a + I\,b$$

```
> evalc( Re(%) ); evalc( Im(%%) );
```

$$a$$

$$b$$

```
> (-3)^(1/4)*exp(a*I);
```

$$(-3)^{(1/4)}\,\mathbf{e}^{(I\,a)}$$

```
> evalc( Re(%) ); evalc( Im(%%) );
```

$$\frac{1}{2}\,3^{(1/4)}\,\sqrt{2}\,\cos(a) - \frac{1}{2}\,3^{(1/4)}\,\sqrt{2}\,\sin(a)$$

$$\frac{1}{2}\,3^{(1/4)}\,\sqrt{2}\,\cos(a) + \frac{1}{2}\,3^{(1/4)}\,\sqrt{2}\,\sin(a)$$

In the last calculation Maple assumes again that `a` is real. Remember:

> The procedure `evalc` assumes all unassigned names to be real.

12.2 Argument and absolute value of a complex number

The absolute value and the argument of a complex number ($\neq 0$) can be searched for separately by the procedures **abs** and **argument**:

```
> abs( I^(5/3) );
```

$$1$$

```
> argument( I^(5/3) );
```

$$\frac{5}{6}\pi$$

Conversion to **polar coordinates** is accomplished by **convert(,polar)**:

```
> convert( I^(5/3) , polar );
```

$$\mathrm{polar}(1, \frac{5}{6}\pi)$$

Conversion to standard complex notation is performed by `evalc`:

```
> evalc( % );
```

$$-\frac{1}{2}\sqrt{3} + \frac{1}{2}I$$

A number given in standard complex form `a+b*I` can be conjugated by **conjugate**. Do *not* apply this procedure to objects that are not in this standard form, but then apply `evalc` first.

```
> conjugate( evalc(I^(5/3)) );
```

$$-\frac{1}{2}\sqrt{3} - \frac{1}{2}I$$

If you don't apply `evalc` first, `conjugate` can go on strike:

```
> conjugate( I^(5/3) );
```

$$\mathrm{conjugate}\left((-1)^{(5/6)}\right)$$

(In older releases, `conjugate` without `evalc` could yield wrong results.)

12.3 The sign of a real or a complex number

The procedure **signum** is the standard Maple test if a *real* number is positive or negative: it yields 1 for a positive number and -1 for a negative number.

```
> signum(Pi - sqrt(10));
```

$$-1$$

```
> signum(sqrt(10) - Pi);
```

$$1$$

```
> signum(0);
```

$$0$$

Generally, if `signum` is applied to an expression that might be zero for some special values of the indeterminates, Maple ignores that possibility. For instance

```
> signum(abs(x)), signum(-abs(x));
```

$$1, -1$$

This looseness can be tightened by assigning a value to the special Maple variable `_Envsignum0` or with an extra argument. For details on `_Envsignum0`, see the on-line help to `signum`.

The procedure `signum` can also be applied to complex numbers: signum(x) is defined as

$$\text{signum}(x) := \frac{\text{abs}(x)}{\text{conjugate}(x)}$$

if $x \neq 0$, and it is 0 if $x = 0$.

Don't confuse `signum` with **sign**: the last is defined as the sign of the leading coefficient of a polynomial with real coefficients. In some cases the results are the same, but in many others not.

For complex numbers you can also use **csgn**, yielding `1`, if Maple can ascertain that it lies in the right half-plane, and `-1`, if Maple knows that it lies in the left half-plane.

12.4 Manipulating products and quotients of radicals

Products and quotients of radicals can be combined with the procedure **combine**:

```
> sqrt(10 - sqrt(7)) * sqrt(10 + sqrt(7));
```

$$\sqrt{10-\sqrt{7}}\sqrt{10+\sqrt{7}}$$

```
> combine( % );
```

$$\sqrt{93}$$

For quotients of radical expressions, you can use the procedure **rationalize**, which tries to remove radicals from the denominator. This procedure must be read from the library before it can be used.

```
> (1+sqrt(3))/(2+sqrt(3)-sqrt(-5));
```

$$\frac{1+\sqrt{3}}{2+\sqrt{3}-I\sqrt{5}}$$

```
> readlib(rationalize)(%);
```

$$-\frac{1}{24}\left(1+\sqrt{3}\right)\left(2+\sqrt{3}+I\sqrt{5}\right)\left(-3+\sqrt{3}\right)$$

```
> evalc(%);
```

$$\frac{1}{4}+\frac{1}{6}\sqrt{3}+\frac{1}{12}I\sqrt{3}\sqrt{5}$$

The last step can also be performed by `expand`.

12.5 Nested radicals and roots of complex numbers

A general tool for nested radicals (roots of expressions containing roots) is **radnormal**, which must be read from the library before it can be used:

```
> readlib(radnormal);
```

proc($expr, opts1$) ... **end**

```
> sqrt( sqrt(2)+I*(-sqrt(3)-sqrt(6)) );
```

$$\sqrt{\sqrt{2}+I\left(-\sqrt{3}-\sqrt{6}\right)}$$

```
> radnormal(%);
```

$$1+\frac{1}{2}\sqrt{2}-\frac{1}{2}I\sqrt{3}\sqrt{2}$$

In some special cases, a **root of a complex number** cannot be simplified in this way, but can be simplified by conversion to polar and back, where Maple can use trigonometric tricks. For example, let's simplify $\sqrt{1+\sqrt{-5}}$. First, we manipulate $1+\sqrt{-5}$:

```
> 1+sqrt(-5);
```

$$1+I\sqrt{5}$$

```
> convert( % , polar);
```

$$\mathrm{polar}\left(\sqrt{6}, \arctan\left(\sqrt{5}\right)\right)$$

Now take the square root of it and apply `evalc`:

```
> sqrt(%);
```

$$\sqrt{\mathrm{polar}\left(\sqrt{6}, \arctan\left(\sqrt{5}\right)\right)}$$

```
> evalc(%);
```

$$\frac{1}{2}\sqrt{2+2\sqrt{6}}+\frac{1}{2}I\sqrt{-2+2\sqrt{6}}$$

Sometimes, it is necessary to combine both methods, starting with the last one and applying `radnormal` and/or `rationalize` afterward.

The procedure `radnormal` can also be used with other roots than the square root. Here is an example:

```
> ( a*(sqrt(2)-1)^3 );
```

$$a\left(\sqrt{2}-1\right)^3$$

```
> expand(%);
```

$$5a\sqrt{2}-7a$$

```
> root[3](%);
```

$$\left(5a\sqrt{2}-7a\right)^{(1/3)}$$

Before `radnormal` can come into action, the factor a must be isolated. For this purpose, we use `simplify( ,power)` twice:

```
> simplify(%,power);
```

$$\left(a\left(5\sqrt{2}-7\right)\right)^{(1/3)}$$

```
> simplify(%,power);
```

$$\left(5\sqrt{2}-7\right)^{(1/3)}a^{(1/3)}$$

Now `radnormal` can do its job:

```
> radnormal(%);
```

$$-a^{(1/3)}+a^{(1/3)}\sqrt{2}$$

12.6 An example: substituting expressions with radicals in polynomials

Expressions containing radicals and complex numbers often originate from solving polynomial equations. For instance,

```
> equa := 2*x^3 - 3*x^2 - 12*x + 5 = 0;
```

$$equa := 2x^3 - 3x^2 - 12x + 5 = 0$$

```
> solutions := solve( equa , x );
```

solutions :=

$$\frac{1}{2}\left(3+12I\sqrt{5}\right)^{(1/3)} + \frac{9}{2}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} + \frac{1}{2},$$

$$-\frac{1}{4}\left(3+12I\sqrt{5}\right)^{(1/3)} - \frac{9}{4}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} + \frac{1}{2} +$$

$$\frac{1}{2}I\sqrt{3}\left(\frac{1}{2}\left(3+12I\sqrt{5}\right)^{(1/3)} - \frac{9}{2}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}}\right),$$

$$-\frac{1}{4}\left(3+12I\sqrt{5}\right)^{(1/3)} - \frac{9}{4}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} + \frac{1}{2} -$$

$$\frac{1}{2}I\sqrt{3}\left(\frac{1}{2}\left(3+12I\sqrt{5}\right)^{(1/3)} - \frac{9}{2}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}}\right)$$

Maple is fully reliable in solving such a polynomial equation, but we will check one of the solutions as a demonstration of more complicated manipulations with radicals. Let's substitute the first solution:

```
> subs( x=solutions[1] , equa );
```

$$2\left(\frac{1}{2}\left(3+12I\sqrt{5}\right)^{(1/3)} + \frac{9}{2}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} + \frac{1}{2}\right)^3 -$$

$$3\left(\frac{1}{2}\left(3+12I\sqrt{5}\right)^{(1/3)} + \frac{9}{2}\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} + \frac{1}{2}\right)^2 -$$

$$6\left(3+12I\sqrt{5}\right)^{(1/3)} - 54\frac{1}{\left(3+12I\sqrt{5}\right)^{(1/3)}} - 1 =$$

$$0$$

The first step in simplifying this is the same as you probably would do in a calculation by hand: the powers and multiplications must be elaborated; this can be achieved with the procedure `expand`. Do *not* start with `simplify`.

```
> expand( % );
```

$$-\frac{3}{4} + 3I\sqrt{5} + \frac{729}{4}\frac{1}{3+12I\sqrt{5}} = 0$$

The last expression can be handled well by `normal`, by `evalc`, or by `simplify`.

Of these three, the most powerful tool for simplifying natural powers of radical expressions is the procedure `simplify` without options. In many other cases, the other two are to be preferred. See Chapter 15, *Manipulating algebraic expressions.*

```
> simplify( % );
```

$$0 = 0$$

The other two solutions can be checked in the same way.

More about solving polynomial equations can be found in Chapter 14, *Polynomial equations and factoring polynomials.* In section E.6 on page 305, you can find how you can check all the candidate solutions at once.

12.7 Converting floating-point numbers to rational numbers

When an expression to be processed contains floating-point numbers, but numerical algorithms should not be used, these numbers must be converted to rational numbers first by **convert(,rational)** or `convert( ,rational, exact)`.

```
> 0.3333333333333333*x - 0.34567;
```

$$.3333333333333333\,x - .34567$$

```
> convert( % , rational , exact );
```

$$\frac{3333333333333333}{10000000000000000}\,x - \frac{34567}{100000}$$

If the second option `exact` is omitted, Maple yields rational numbers that approximate the given floating-point numbers, where the accuracy of these approximations is controlled by the value of `Digits`, which is 10 initially.

```
> convert( %% , rational );
```

$$\frac{1}{3}\,x - \frac{14206}{41097}$$

This procedure can be applied on polynomials as well in order to convert floating-point coefficients.

12.8 Rounding rational numbers to integers

Real numbers can be rounded to an integer by several procedures: `trunc`, `round`, `floor`, and `ceil`. The procedure **round** tries to yield the nearest integer:

```
> round(sqrt(5)), round(sqrt(8)), round(-sqrt(8));
```

$$2,\ 3,\ -3$$

The procedure **trunc** yields the first integer that is encountered in the direction of zero:

```
> trunc(sqrt(8)), trunc(-sqrt(8));
```

$$2, -2$$

The procedures **floor** and **ceil** do what their names suggest:

```
> floor(sqrt(8)), floor(-sqrt(8));
```

$$2, -3$$

```
> ceil(sqrt(8)), ceil(-sqrt(8));
```

$$3, -2$$

Moreover a fractional part can be calculated by **frac**:

```
> frac(37/8), frac(sqrt(8));
```

$$\frac{5}{8}, 2\sqrt{2} - 2$$

where `frac(x) + trunc(x) = x`.

chapter 13

Polynomials and rational expressions

This chapter deals with elementary aspects of handling polynomials: the use of arithmetic operations, extracting a coefficient, calculating greatest common divisors and resultants, etc., and also with handling rational expressions.

13.1 Polynomials and the standard arithmetic operators

For calculations with polynomials and quotients of polynomials, the arithmetic operators `+`, `-`, `*`, `/`, and `^` can be used. Generally, multiplications and exponentiations of polynomials, consisting of more than one term, are not worked out automatically. For instance,

```
> (z-1)/2*z + (y+a)^2;
```

$$\frac{1}{2}(z-1)\,z+(y+a)^2$$

Maple leaves the choice to the user: you can keep such an expression as it is or manipulate it into some other form. Here you can ask to expand the products and powers over the sums with the procedure **expand**:

```
> expand( % );
```

$$\frac{1}{2}z^2-\frac{1}{2}z+y^2+2\,y\,a+a^2$$

When using `expand` on expressions containing radicals and/or function calls, beware of possibly unwanted side effects. See section 15.6 on page 196 and section 15.7 on page 198 for restrictions in the use of `expand` for such cases and the methods to be used in such a case.

If one or more factors of a product should not be expanded over the other factors, the factors to be kept intact must be given as extra arguments:

```
> 10*(a-b)*(c+1)*(x+1)^2;
```

$$10\,(a-b)\,(c+1)\,(x+1)^2$$

```
> expand( % , a-b , c+1 );
```

$$10\,(a-b)\,(c+1)\,x^2+20\,(a-b)\,(c+1)\,x+10\,(a-b)\,(c+1)$$

The opposite procedure of `expand` for polynomials is `factor`, Maple's powerful procedure for factorization, which is discussed in section 14.6 on page 177, together with its variants.

Automatic simplification of products and quotients of polynomials is restricted to some simple cases; in all other cases the user is in command:

```
> x^3 * 2*x^4 / x^5 + 8*(x-1);
```

$$2x^2 + 8x - 8$$

Maple can handle **very large polynomials** with up to 65,535 terms, but if it finds too many terms in a polynomial, possibly as an intermediate result in a calculation, it gives a message that the object is too large, for instance here, where we try to calculate the product of two polynomials, each containing 256 terms:

```
> expand(sum('x.i',i=1..256)*sum('y.i',i=1..256)):
Error, object too large
```

13.2 Division of polynomials with a remainder

Division with a remainder is supplied by **quo**, to be compared with `iquo` for integers. Let's divide $3x^5$ by $x^2 - a$ as polynomials in x:

```
> quo( 3*x^5 , x^2-a , x );
```

$$3xa + 3x^3$$

The remainder can be found by appending a name as a fourth argument:

```
> quo( 3*x^5 , x^2-a , x , 'r1' );
```

$$3xa + 3x^3$$

```
> r1;
```

$$3xa^2$$

In the first command the forward quotes around `r1` are superfluous as `r1` is not assigned to previously, but generally it is wise to use these forward quotes for an argument that is assigned to by the procedure quo, in order to make sure that this argument cannot be evaluated before it is used by the procedure. This is explained for `iquo` in section 5.4 on page 67.

As a demonstration, we calculate the original polynomial from both results:

```
> expand(%%*(x^2-a)+r1);
```

$$3x^5$$

There is another procedure **rem** that calculates the remainder of the division and assigns the quotient to an optional fourth argument.

13.3 The greatest common divisor and the least common multiple

The greatest common divisor of two polynomials can be calculated with **gcd** or **gcdex**. We apply these to the polynomials p1 and p2 created here by expand, so that you can predict the result immediately:

```
> p1 := expand( (x-3) * (x-a) * (x^3+1) );
```

$$p1 := x^5 + x^2 - x^4\,a - x\,a - 3\,x^4 - 3\,x + 3\,x^3\,a + 3\,a$$

```
> p2 := expand( (x-3)^2 * (2*x-3*a) );
```

$$p2 := 2\,x^3 - 3\,x^2\,a - 12\,x^2 + 18\,x\,a + 18\,x - 27\,a$$

```
> gcd( p1 , p2 );
```

$$x - 3$$

Here no specification that p1 and p2 are to be perceived as polynomials in x is expected; gcd can be applied to polynomials in more than one variable. In fact, Maple perceives p2 as a polynomial in a and x, not being brainwashed about some special quality of the character x.

You might be tempted to tell Maple that the arguments to gcd are polynomials in x, but supplying x as a third argument generates an error:

```
> gcd(p1,p2,x);
Error, The optional 3rd argument given to 'gcd' was x
This argument to 'gcd' is for returning the cofactor.
It is not for specifying the variable in which to compute the GCD.
If assigned, it could create a recursive definition of a name.
```

In fact, Maple has no need for the knowledge that you see p1 and p2 as polynomials in x. If gcd gets a third and a fourth argument, it tries to assign the cofactors `p1/gcd(p1,p2)` and `p2/gcd(p1,p2)` to them.
If Maple would have assigned the cofactor `p1/gcd(p1,p2)` to x, this would generate a recursive referrence; see section 3.10 on page 41. However, Maple sees that x is a variable occurring in p1 and p2 and prevents the assignment (not before release V.4).

In many applications of the greatest common divisor, its main theorem can be used: if p_1 and p_2 are polynomials and if d is the gcd of these two polynomials, there exist polynomials q_1 and q_2 such that $q_1 p_1 + q_2 p_2 = d$. These polynomials q_1 and q_2 can be calculated with **gcdex**. Contrary to gcd, gcdex wants to know in which of the variables the gcd is to be calculated; in this case, we must give x as the third argument to gcdex. The fourth and fifth arguments must be names to which the calculated polynomials $q1$ and $q2$ are to be assigned:

```
> gcdex(p1, p2, x, 'q1', 'q2');
```

$$x - 3$$

```
> q1;
```

$$\frac{1}{14}\,\frac{(224+36\,a+18\,a^2+9\,a^3)\,x}{a\,(8\,a-24-81\,a^3+27\,a^4)}-$$
$$\frac{1}{28}\,\frac{1344+27\,a^4+54\,a^3+108\,a^2+224\,a}{a\,(8\,a-24-81\,a^3+27\,a^4)}$$

As a demonstration, we check that $\gcd(p1,p2) = q1\,p1 + q2\,p2$:

```
> normal( expand( q1 * p1 + q2 * p2 ) );
```

$$x-3$$

The use of `expand` and `normal` for the above type of simplification is discussed in section 13.8 on page 166.

The procedure **lcm** computes the least common multiple of its arguments, where these can be any number.

13.4 The resultant of two polynomials

The resultant of the two polynomials `p1` and `p2` from the previous section is zero, as their gcd is not 1. It can be calculated with **resultant**:

```
> resultant( p1 , p2 , x );
```

$$0$$

Here is one more example.

```
> resultant( p1+1 , p2 , x );
```

$$-81\,a^5+162\,a^4-24\,a^2+48\,a-32$$

The **bezout matrix** can be created with the procedure `linalg[bezout]`, the **discriminant** with `discrim`.

Another important concept in calculations with polynomials is the *Gröbner basis* together with the *Buchberger algorithm* for calculating such a basis. This can be very helpful, especially in solving systems of polynomial equations. This subject is discussed in section 14.9 on page 180.

13.5 The coefficients of a polynomial

You can ask Maple to describe a polynomial in several variables as a polynomial in one of these variables with the aid of
collect:

```
> pol := a^2*x^2*y+y^2*a-2*b*x*y-b^2*x^2*y+b*x*y^2+
>        c*y^2+3*x*y^2-6*x*y;
```

$$pol := a^2 x^2 y + y^2 a - 2 b x y - b^2 x^2 y + b x y^2 + c y^2 + 3 x y^2 - 6 x y$$

This is made into a polynomial in x with:

```
> collect( % , x );
```

$$\left(a^2 y - b^2 y\right) x^2 + \left(b y^2 - 2 b y - 6 y + 3 y^2\right) x + y^2 a + c y^2$$

It is also possible to have Maple perceive an expression as a polynomial in two or more given variables:

```
> collect( pol , [x,y] , distributed );
```

$$\left(a^2 - b^2\right) y x^2 + (3 + b) x y^2 + (-2 b - 6) x y + (a + c) y^2$$

Without the option `distributed`, the result is a polynomial in x where the coefficients are polynomials in y:

```
> collect( pol , [x,y] );
```

$$\left(a^2 - b^2\right) y x^2 + \left((3 + b) y^2 + (-2 b - 6) y\right) x + (a + c) y^2$$

A nice facility within `collect` is the ability to process the coefficients of the result with a procedure or function given as an extra argument, *without forward quotes*:

```
> collect( pol , x , factor );
```

$$y (a - b) (a + b) x^2 + y (y - 2) (3 + b) x + (a + c) y^2$$

Coefficients of a polynomial can be selected with the procedure **coeff**. For instance, the coefficient of x^2 in the previous polynomial can be achieved by

```
> coeff( pol , x , 2 );
```

$$a^2 y - b^2 y$$

The degree of a polynomial can be calculated with **degree**:

```
> degree( pol , x );
```

$$2$$

```
> degree( pol , [x,y] );
```

$$3$$

It is essential for this procedure that the polynomial be given in either collected or expanded form. Otherwise, Maple does not accept it, or it might yield incorrect results:

```
> (x-3)^2 - x^2;
```

$$(x-3)^2 - x^2$$

```
> degree( % , x );
```

$$2$$

The correct degree is found if the polynomial is expanded before `degree` comes into action:

```
> degree( expand(%%) , x );
```

$$1$$

The same is true for some other procedures, for example `tcoeff`, which tries to compute the coefficient of the lowest-degree monomial:

```
> (x-3)^2-9;
```

$$(x-3)^2 - 9$$

```
> tcoeff(%);
```

$$0$$

```
> tcoeff( expand(%%) );
```

$$-6$$

> The procedures `lcoeff`, `tcoeff`, `degree`, `ldegree`, `content`, and `norm` must only be applied to polynomials in expanded or collected form.

In releases before V.5 the same is true for the procedure `coeff`.

- **lcoeff** : yields the coefficient of the monomial corresponding to the degree of the polynomial
- **ldegree** : yields the lowest degree
- **tcoeff** : yields the trailing coefficient: the coefficient of the monomial corresponding to the lowest degree of the polynomial
- **content** : yields the greatest common divisor of the coefficients of a polynomial in one or more variables

- `norm` : the n-norm of the coefficients of an expanded polynomial p in x can be calculated with `norm(p,n,x)`

13.6 Truncating a polynomial above some degree

Here are three ways for truncating a polynomial above some degree. Consider the following polynomial

```
> pol := sum( 2^(-i)*t^i , i=0..18 );
```

$$pol := 1+\frac{1}{2}t+\frac{1}{4}t^2+\frac{1}{8}t^3+\frac{1}{16}t^4+\frac{1}{32}t^5+\frac{1}{64}t^6+\frac{1}{128}t^7+\frac{1}{256}t^8+\frac{1}{512}t^9+\frac{1}{1024}t^{10}+\frac{1}{2048}t^{11}+\frac{1}{4096}t^{12}+\frac{1}{8192}t^{13}+\frac{1}{16384}t^{14}+\frac{1}{32768}t^{15}+\frac{1}{65536}t^{16}+\frac{1}{131072}t^{17}+\frac{1}{262144}t^{18}$$

This can be truncated above degree 7 as follows. First, apply `series` to the order 8:

```
> series( pol , t , 8 );
```

$$1+\frac{1}{2}t+\frac{1}{4}t^2+\frac{1}{8}t^3+\frac{1}{16}t^4+\frac{1}{32}t^5+\frac{1}{64}t^6+\frac{1}{128}t^7+O(t^8)$$

Then convert the result into a polynomial:

```
> convert(%,polynom);
```

$$1+\frac{1}{2}t+\frac{1}{4}t^2+\frac{1}{8}t^3+\frac{1}{16}t^4+\frac{1}{32}t^5+\frac{1}{64}t^6+\frac{1}{128}t^7$$

It is also possible to realize the same result with the aid of simplifying according to **side relations** (see section 11.11 on page 147) as follows:

```
> simplify( pol , {t^8=0} );
```

$$1+\frac{1}{2}t+\frac{1}{4}t^2+\frac{1}{8}t^3+\frac{1}{16}t^4+\frac{1}{32}t^5+\frac{1}{64}t^6+\frac{1}{128}t^7$$

As an alternative, you can apply `select`. See section 10.6 on page 131. The second argument is the polynomial that should be truncated, the first argument is a procedure that tests each term for whether its degree is lower than 8. This yields the same result:

```
> select( polyterm->degree(polyterm,t)<8 , pol );
```

$$1+\frac{1}{2}t+\frac{1}{4}t^2+\frac{1}{8}t^3+\frac{1}{16}t^4+\frac{1}{32}t^5+\frac{1}{64}t^6+\frac{1}{128}t^7$$

13.7 Sorting a polynomial

When the order is not essential in mathematical sense, results are often printed in an order based on the order in which subexpressions are stored in the computer memory. This is also the case with polynomials. For instance,

```
> pol := a^5*x*y+y^2*a-b^2*x^2*y+b*x*y^2-y*x^2+
>      x^2*y^2-y^2*c+3*x*y^2;
```

$$pol := a^5\,x\,y + y^2\,a - b^2\,x^2\,y + b\,x\,y^2 - y\,x^2 + x^2\,y^2 - y^2\,c + 3\,x\,y^2$$

```
> a*y^2;
```

$$y^2\,a$$

The product is printed as $y^2\,a$, in a different order than it has been entered. Maple takes the printing order according to the order in which expressions are stored in the computer memory, and obviously y^2 precedes a in the present session. In the same way, we can see that the expression $y^2\,a$ is the second of the components of *pol* in the computer memory order.

Readability of a polynomial can be improved sometimes by changing the order of factors and terms. This is possible with the procedure **sort**. Let's sort `pol` as a polynomial in `x` and `y`:

```
> sort( pol , [x,y] );
```

$$x^2\,y^2 - x^2\,y - b^2\,x^2\,y + b\,x\,y^2 + 3\,x\,y^2 + a^5\,x\,y + a\,y^2 - c\,y^2$$

Or you can apply `collect` first:

```
> collect( pol , [x,y] , distributed );
```

$$a^5\,x\,y + (3+b)\,x\,y^2 + x^2\,y^2 + \left(-b^2-1\right)x^2\,y + (a-c)\,y^2$$

```
> sort( % , [x,y] );
```

$$x^2\,y^2 + \left(-b^2-1\right)x^2\,y + (3+b)\,x\,y^2 + a^5\,x\,y + (a-c)\,y^2$$

The procedure `sort` changes the order of subexpressions in the memory. Therefore, `pol` is printed differently from the way it was printed before `sort` was applied:

```
> pol;
```

$$x^2\,y^2 - x^2\,y - b^2\,x^2\,y + b\,x\,y^2 + 3\,x\,y^2 + a^5\,x\,y + a\,y^2 - c\,y^2$$

By applying `sort` on `pol`, we have ordered its subexpressions in the memory as well:

```
> y^2*a;
```

$$a\,y^2$$

But the execution of `sort(pol,[x,y])` does not enforce a standard order for all polynomials in x and y. For instance,

```
> y^10*a*x^20 + b*y*x^30;
```

$$y^{10}\,a\,x^{20} + b\,y\,x^{30}$$

13.8 Simplifying rational expressions

Rational numbers are simplified automatically by calculating the greatest common divisor of numerator and denominator and dividing this out. That is not the case with rational expressions: only factors that are "seen" by Maple directly are divided out automatically:

```
> expand(x*(x-a)^2); expand(x*(x-a)*(x+a));
```

$$x^3 - 2\,x^2\,a + x\,a^2$$

$$x^3 - x\,a^2$$

```
> %%/%;
```

$$\frac{x^3 - 2\,x^2\,a + x\,a^2}{x^3 - x\,a^2}$$

The procedure **normal** divides out the greatest common divisor. Clearly, this procedure does not bother about the possibility that such a factor might be zero for some set of values for the variables: that is the mathematical responsibility of the user.
It is not a good idea to use **simplify** for such a job as, generally, this procedure tries to do more, perhaps unwanted, simplifications, and takes more time:

```
> normal( % ) ;
```

$$-\frac{-x + a}{x + a}$$

The same result could have been found with the procedure `factor`, but factorization is much more complicated and time-consuming than calculating the greatest common divisor of two polynomials.

The procedure `normal` can also convert a sum of quotients into one quotient:

```
> 1/(x^2-3*x) - x + 1/(x^2-9);
```

$$\frac{1}{x^2 - 3\,x} - x + \frac{1}{x^2 - 9}$$

```
> normal( % ) ;
```

$$-\frac{x^4 - 9\,x^2 - 2\,x - 3}{(x^2 - 9)\,x}$$

Here the denominator is a product of polynomials. If you don't want this, call `normal` with a second argument `expanded`:

```
> normal( %% , expanded) ;
```

$$\frac{-x^4+9x^2+2x+3}{x^3-9x}$$

The counterpart of `normal` is **expand** in this case. It is easy to see what `expand` does with a quotient by looking at the Maple structure of such a quotient:

```
> (x^2-b*x*y+c*y^2)/(a-c);
```

$$\frac{x^2-bxy+cy^2}{a-c}$$

```
> op( % );
```

$$x^2-bxy+cy^2,\ \frac{1}{a-c}$$

Maple perceives the previous expression as the product of two factors: the numerator and $(a-c)^{-1}$. So `expand` yields the sum of the products of $(a-c)^{-1}$ with the terms of the first component:

```
> expand( %% );
```

$$\frac{x^2}{a-c}-\frac{bxy}{a-c}+\frac{cy^2}{a-c}$$

In the next example we expand the coefficient `-5/8` of the quotient over the terms of the numerator with **combine**:

```
> -5*(x+y)/(8*(x-2)*(y-1)); combine(%);
```

$$-\frac{5}{8}\frac{x+y}{(x-2)(y-1)}$$

$$\frac{-\frac{5}{8}x-\frac{5}{8}y}{(y-1)(x-2)}$$

13.9 Numerator and denominator

In the previous section, the procedure `op` has been used in finding the subexpressions of a quotient. This way of breaking down a quotient is not always efficient. Closer to the usual way of looking at quotients are the procedures **numer** and **denom**:

```
> 5*(x+y)/(8*(x-2)*(y-1));
```

$$\frac{5}{8}\frac{x+y}{(x-2)(y-1)}$$

```
> numer( % );
```

$$5x + 5y$$

```
> denom( %% );
```

$$8(x-2)(y-1)$$

You can use these two procedures when you want to manipulate the denominator or the numerator separately. For example:

```
> (x^3 - 4*x^2 + 7*x -12)/(x^2 + 7*x + 10);
```

$$\frac{x^3 - 4x^2 + 7x - 12}{x^2 + 7x + 10}$$

```
> numer(%) / factor(denom(%));
```

$$\frac{x^3 - 4x^2 + 7x - 12}{(x+5)(x+2)}$$

13.10 More tools

Maple supplies

- conversion of a rational expression to a truncated **continued fraction** by the procedure `convert( ,confrac,x)`, useful for numerical calculations;
- **partial fraction decomposition** by `convert( ,parfrac,x)` and `convert( ,fullparfrac,x)`; see also section 9.2 on page 119.
- calculation of an **interpolating polynomial function**, the graph of which contains a given sequence of points, with **interp**.

13.11 Reliability

Apart from the fact that sometimes the user must expand or collect a polynomial before it can be manipulated, as shown in section 13.5 on page 162, in principle there are no **reliability** problems in calculations on polynomials and quotients of rational functions in Maple.

chapter 14

Polynomial equations and factoring polynomials

Solving polynomial equations and systems of polynomial equations is a classical type of problem. Its study received new impetus from the fact that computers can now handle laborious algorithms, using factorization, resultants, and the more recent Buchberger algorithm for calculating Gröbner bases. Maple is in the front line of using these developments. This chapter discusses Maple's powerful tools in this field, as well as the related subject of symbolic roots of polynomials.

When solving equations with parameters, remember that these parameters are interpreted as abstract items; solutions are to be interpreted as "general solutions". If special values for these parameters are substituted, the solution might be no longer valid for this case; see for an example section 1.3 on page 6.

14.1 Solving polynomial equations symbolically

Maple uses several tricks for solving polynomial equations. Obviously, it knows the standard algorithms for first-, second-, third-, and fourth-degree polynomials. No general algorithms for degrees higher than four can be constructed, yet **solve** can often find a solution of a polynomial equation of higher degree. Here is an example:

```
> x^5+x^4+1=0;
```

$$x^5 + x^4 + 1 = 0$$

```
> solve( % , x );
```

$$-\frac{1}{2}+\frac{1}{2}I\sqrt{3}, -\frac{1}{2}-\frac{1}{2}I\sqrt{3}, -\frac{1}{6}\ \%1^{(1/3)} - 2\frac{1}{\%1^{(1/3)}},$$
$$\frac{1}{12}\ \%1^{(1/3)} + \frac{1}{\%1^{(1/3)}} +$$
$$\frac{1}{2}I\sqrt{3}\left(-\frac{1}{6}\ \%1^{(1/3)} + 2\frac{1}{\%1^{(1/3)}}\right),$$
$$\frac{1}{12}\ \%1^{(1/3)} + \frac{1}{\%1^{(1/3)}} -$$
$$\frac{1}{2}I\sqrt{3}\left(-\frac{1}{6}\ \%1^{(1/3)} + 2\frac{1}{\%1^{(1/3)}}\right)$$

$$\%1 := 108 + 12\sqrt{69}$$

E. Kamerich, *A Guide to Maple*

We obtain five solutions, expressed with the aid of the abbreviation %1. The procedure `solve` handles this equation with the aid of factorization of $x^5 + x^4 + 1$ over the rational numbers.

```
> factor( x^5+x^4+1 );
```

$$\left(x^2 + x + 1\right)\left(x^3 - x + 1\right)$$

Another trick is **decomposition of polynomials**, available by the procedure **compoly**. When Maple solves the equation,

$$x^6 + 2x^4 + 2x^3 + x^2 + 2x + 2 = 0$$

it does so by creating a decomposition of the left-hand side: it finds that the left-hand side can be described as $1 + t^2$, where $t = 1 + x + x^3$.

In spite of all the tricks of Maple, it is inevitable that solving fails for most high-degree polynomials. If Maple cannot find an explicit sequence of all exact solutions, a symbolic representation of the intermediate result emerges, using a special Maple construction, **RootOf** expressions.

```
> x^7+11*x^6+12*x^5+10*x^3-11*x^2-41*x^4-13*x+33;
```

$$x^7 + 11\,x^6 + 12\,x^5 + 10\,x^3 - 11\,x^2 - 41\,x^4 - 13\,x + 33$$

```
> solve( % , x );
```

$$-4 + 3\sqrt{3},\ -4 - 3\sqrt{3},\ \mathrm{RootOf}\left(\_Z^5 + 3\,\_Z^4 - \_Z^3 - \_Z - 3\right)$$

First, `solve` factors the given polynomial into a product of polynomials with rational coefficients. Then each factor is dealt with by `solve`. As there are no general algorithms for solving a fifth-degree polynomial equation and Maple's special tricks are not successful here, Maple yields

```
RootOf(_Z^5 + 3*_Z^4 - _Z^3 - _Z - 3)
```

as a part of the solution. This is a symbolic representation of the set of five roots of the equation

$$\_Z\hat{}5 + 3 * \_Z\hat{}4 - \_Z\hat{}3 - \_Z - 3 = 0$$

In order to avoid the complicated results of solving irreducible *fourth-degree equations*, these solutions are represented with RootOf expressions as well, unless the variable `_EnvExplicit` is set to the value `true`.

If `solve` handles a polynomial equation, it renders each solution as many times as its multiplicity.

The results of solving polynomial equations found with `solve` are **reliable**. (In releases before V.4, there is an exception: if an equation contains floating-point numbers, the set of solutions may be incomplete.)

Maple offers the procedures `sturmseq` and `sturm` based on the **Sturm theorem**, concerning real roots, and the procedure `realroots` for calculating isolating intervals of real roots.

14.2 Solving modest systems of polynomial equations

Here is an example: calculating the intersection points of two circles.

```
> circle1 := x^2 + y^2 = 2*x + 8*y + 25;
```

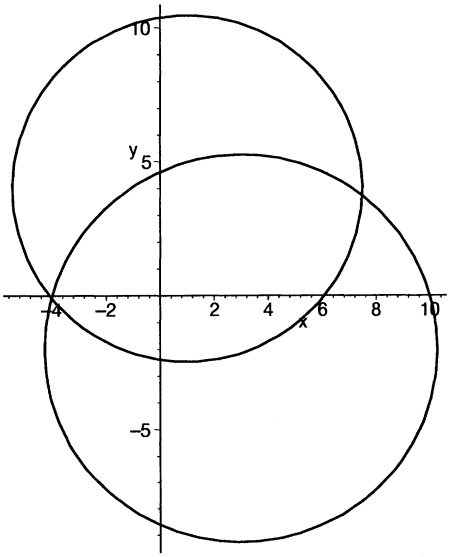
$$circle1 := x^2 + y^2 = 2\,x + 8\,y + 25$$

```
> circle2 := x^2 + y^2 = 6*x - 4*y + 40;
```

$$circle2 := x^2 + y^2 = 6\,x - 4\,y + 40$$

Let's draw graphs in order to "see" the solutions beforehand:

```
> plots[implicitplot](circle1,x=-7..12,y=-10..12):
> plots[implicitplot](circle2,x=-7..12,y=-10..12):
> plots[display]( {%,%%}, scaling=CONSTRAINED );
```



A system of equations to be solved should be entered tied together in a set; the variables must be entered as a set, too.

```
> solve( {circle1,circle2} , {x,y} );
```

$$\left\{ y = \frac{1}{4}\,\mathrm{RootOf}\left(-55 + 10\_Z^2 - 146\_Z\right),\right.$$
$$\left. x = \frac{3}{4}\,\mathrm{RootOf}\left(-55 + 10\_Z^2 - 146\_Z\right) - \frac{15}{4} \right\}$$

Here the solution is given as a set of two equations with the aid of a **RootOf** expression. The use of `RootOf` is a compact and well-ordered way of representing the solution; Maple has no difficulty in finding a more explicit solution, but leaves the choice to the user. We can substitute all possible values for this `RootOf` expression with **allvalues**.

```
> allvalues( % );
```

$$\left\{ y = \frac{73}{40} + \frac{1}{40}\sqrt{5879},\ x = \frac{69}{40} + \frac{3}{40}\sqrt{5879} \right\},$$
$$\left\{ y = \frac{73}{40} - \frac{1}{40}\sqrt{5879},\ x = \frac{69}{40} - \frac{3}{40}\sqrt{5879} \right\}$$

We see a sequence of two sets, each representing a solution. Generally, the procedure `allvalues` substitutes the same value in each occurrence of that `RootOf` expression, so it generates two values here. This behavior of `allvalues` (from release V.4) yields all solutions, if the equal `RootOf` expressions are dependent, which can be the case mostly. However, it is possible that two equal `RootOf` expressions must be handled independently from each other. Obviously, this is the case when a system can be split into two subsystems, for instance. Here is an obvious example:

```
> {x^2-3*x=20,y^2-3*y=20};
```

$$\left\{ x^2 - 3x = 20,\ y^2 - 3y = 20 \right\}$$

```
> eqs:=%: solve(eqs,{x,y});
```

$$\left\{ x = \mathrm{RootOf}\left(\_Z^2 - 3\_Z - 20\right),\ y = \mathrm{RootOf}\left(\_Z^2 - 3\_Z - 20\right) \right\}$$

```
> allvalues(%);
```

$$\left\{ x = \frac{3}{2} + \frac{1}{2}\sqrt{89},\ y = \frac{3}{2} + \frac{1}{2}\sqrt{89} \right\},$$
$$\left\{ y = \frac{3}{2} - \frac{1}{2}\sqrt{89},\ x = \frac{3}{2} - \frac{1}{2}\sqrt{89} \right\}$$

We obtain only two of the four solutions. In this case, we would have to add the option **`independent`**. In more complicated cases, for instance with nested `RootOf` expressions, `allvalues` may handle equal `RootOf` expressions independently. In such cases, checking solutions is a must.
(For earlier releases than V.4, independent treatment of `RootOf`s is the default and the other behavior can be ordered by the option `'d'`.)

```
> allvalues(%%,independent);
```

$$\{x = \%2, y = \%2\}, \{y = \%1, x = \%2\},$$
$$\{x = \%1, y = \%2\}, \{y = \%1, x = \%1\}$$

$$\%1 := \frac{3}{2} - \frac{1}{2}\sqrt{89}$$

$$\%2 := \frac{3}{2} + \frac{1}{2}\sqrt{89}$$

If you are not sure that `allvalues` generates all values, use the option `independent` and check each item. Checking is never a bad habit! In the present case you can check each value as follows:

```
> sols:=%:
> subs(sols[1],eqs);
```

$$\left\{ \left(\frac{3}{2} + \frac{1}{2}\sqrt{89}\right)^2 - \frac{9}{2} - \frac{3}{2}\sqrt{89} = 20 \right\}$$

```
> expand(%);
```

$$\{20 = 20\}$$

and so on. The section E.6 on page 305 shows an example where the work is done with more comfort by applying a loop.

It is wise to check solutions anyhow: solving systems of equations is a complicated process and `allvalues` might have neglected the equivalence of two `RootOf` expressions.

If `solve` handles a set of polynomial equations, it renders each solution as many times as its multiplicity (from release V.4).

Often the **mistake of entering in `solve` fewer variables than necessary** is made. Suppose you are interested only in the possible values of x. Then you might be tempted to enter:

```
> solve( {circle1,circle2} , x );
```

This does not yield a solution, because the system of equations is satisfied for no values of x unless y has a matching value.

For solving *systems of linear equations*, the specific facilities described in section 18.10 on page 257, are useful.

For stepwise **elimination**, use `eliminate`. For instance:

```
> eliminate( {circle1,circle2} , x );
```

A contrary procedure is `algcurves[parameterization]`, which can calculate a **parameterization** for a two-dimensional curve, implicitly defined by an irreducible polynomial in two variables, if the genus of this curve is 0 (check irreducibility with `AFactor` and genus with `algcurves[genus]`).

14.3 Finding or approximating the elements represented by a RootOf expression

In the previous sections you can see that Maple often uses a symbolic RootOf representation of the solutions of a polynomial equation instead of the actual sequence of solutions. Such a RootOf expression can be created explicitly by the procedure **RootOf**.

```
> 5*x^2 = 30*x + 10 ;
```

$$5x^2 = 30x + 10$$

```
> R1 := RootOf( % , x );
```

$$R1 := \mathrm{RootOf}(\_Z^2 - 6\_Z - 2)$$

The default variable for RootOf is _Z, so, in the last result, Maple omits the second argument of RootOf. Moreover, " = 0" is omitted and only the left-hand side of the equation is given.

Here Maple can find the roots in an explicit form. You can ask for them with the procedure **allvalues**, described in the previous section, possibly with the option independent for handling equal RootOf expressions within an expression or a list/set of expressions separately.

```
> allvalues(R1);
```

$$3 + \sqrt{11},\ 3 - \sqrt{11}$$

If Maple cannot find the sequence of explicit roots exactly, approximations are calculated.

```
> RootOf( x^6 + 6*x^5 + 2 );
```

$$\mathrm{RootOf}(\_Z^6 + 6\_Z^5 + 2)$$

```
> allvalues( % );
```

$$-5.999742743,\ -.8269055711,\ -.2289757797 - .7744287008\,I,$$
$$-.2289757797 + .7744287008\,I,\ .6422999369 - .4534117425\,I,$$
$$.6422999369 + .4534117425\,I$$

The procedure evalf can be applied directly to a RootOf expression, but this produces an approximation of only one of the implied roots.

```
> evalf( R1 );
```

$$-.3166247904$$

> Do not apply `evalf` to an expression containing a `RootOf` subexpression unless you are content with an approximation of only one of its values.

You can get a list of approximations in the following way: apply `allvalues`, then apply `evalf` to the *list* of the results. For instance,

```
> evalf( [ allvalues(R1) ] );
```

$$[6.316624790,\ -.316624790]$$

14.4 Calculating with `RootOf` expressions

Maple can handle `RootOf` expressions in algebraic calculations.

```
> eq :=  x^5-t*x^2+p=0;
```

$$eq := x^5 - t\,x^2 + p = 0$$

```
> RootOf( % , x );
```

$$\mathrm{RootOf}(\_Z^5 - t\,\_Z^2 + p)$$

```
> %^7;
```

$$\mathrm{RootOf}(\_Z^5 - t\,\_Z^2 + p)^7$$

```
> simplify(%,RootOf);
```

$$\mathrm{RootOf}(\_Z^5 - t\,\_Z^2 + p)^4\,t - \mathrm{RootOf}(\_Z^5 - t\,\_Z^2 + p)^2\,p$$

Instead of `simplify(  ,RootOf)` we could have used `evala`, discussed in section 14.10 on page 183. A different approach can be found in section 11.11 on page 147.

In order to obtain more compact results in such calculations, we can abbreviate this `RootOf` expression with an **alias**. See section B.1 on page 285.

```
> alias( RO = RootOf(eq,x) );
```

$$I,\ RO$$

The result of an `alias` command is a sequence of all aliases known to Maple at that moment.

From now on, Maple prints *RO* instead of RootOf(_$Z^5 - t_Z^2 + p$) and recognizes RO in the input. You should remember its definition, as it is not possible to find the internal interpretation of such an alias easily.

Now we can use RO in calculations such as:

```
> simplify( RO^7 );
```

$$RO^4\, t - RO^2\, p$$

14.5 RootOf expressions versus radicals

A radical can be converted to a RootOf expression by **convert(,RootOf)**:

```
> 5^(1/3); convert(%,RootOf);
```

$$5^{(1/3)}$$

$$\text{RootOf}\left(\_Z^3 - 5\right)$$

```
> a^2 - 2^(1/3)*b^(1/6); convert(%,RootOf);
```

$$a^2 - 2^{(1/3)}\, b^{(1/6)}$$

$$a^2 - \text{RootOf}\left(\_Z^3 - 2\right) \text{RootOf}\left(\_Z^6 - b\right)$$

Sometimes it helps Maple if the user applies this conversion. For instance, see section 4.8 on page 51, and section 14.10 on page 183.

> Maple prefers calculating with RootOf expressions above using radicals.

Conversion the other way is available with **convert(,radical)**:

```
> convert(%,radical);
```

$$a^2 - 2^{(1/3)}\, b^{(1/6)}$$

Observe the difference with **allvalues**: a RootOf expression is essentially a multivalued expression; all values can be achieved with allvalues, while convert (,radical) picks up only one, so you might lose solutions.

Often, RootOf expressions cannot be converted to radicals.

```
> RootOf(x^7+x+2,x); convert(%,radical);
```

$$\mathrm{RootOf}(\_Z^7 + \_Z + 2)$$

$$\mathrm{RootOf}(\_Z^7 + \_Z + 2)$$

14.6 Factoring with the procedure `factor`

Maple has a fast and powerful factorization facility, used frequently in various algorithms. It is called with the procedure **factor**.

```
> (8*x^5-6*x^4-78*x^3+72*x^2+190*x-210)/(x^2-49);
```

$$\frac{8x^5 - 6x^4 - 78x^3 + 72x^2 + 190x - 210}{x^2 - 49}$$

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

$$2\,\frac{(4x-7)\,(x^2-5)\,(x^2+x-3)}{(x-7)\,(x+7)}$$

The resulting factors are *irreducible* over Q, that is, each of them cannot be factored into a true product of polynomials of degrees > 1 with coefficients in Q. We can factor x^2-5 into $(x-\sqrt{5})(x+\sqrt{5})$ and x^2+x-3 into $(x+1/2+1/2\sqrt{13})(x+1/2-1/2\sqrt{13})$, but both cannot be factored over Q, so the procedure `factor` leaves these polynomials unfactored.

Here is another example, where `factor` is surprisingly successful:

```
> (a^2 - sqrt(5))^3;
```

$$\left(a^2 - \sqrt{5}\right)^3$$

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

$$a^6 - 3\,a^4\sqrt{5} + 15\,a^2 - 5\sqrt{5}$$

$$\left(a^2 - \sqrt{5}\right)^3$$

As you can see, `factor` can use $\sqrt{5}$ here. This is because it occurs in the given argument. But `factor` does not introduce $5^{1/4}$. However, in section 14.10 on page 183 you will see that we can tell `factor` to use roots of polynomial equations.

The procedure `factor` can handle polynomials in more than one variable equally well, and does so quickly.

```
> 192*c^6*u^4*t^4 + 64*c^4*u^2*t^2 + 168*c^5*u^6*t^2 +
> 56*c^3*u^4 - 120*c^5*u^5*t^3 - 40*c^3*u^3*t -
> 480*c^4*t^7*u^2 - 160*c^2*t^5 - 420*c^3*t^5*u^4 -
```

```
> 140*c*t^3*u^2 + 588*c^3*t^6*u^3 + 196*c*t^4*u -
> 768*u^6*t^4*c^3 - 256*u^4*t^2*c - 672*u^8*t^2*c^2 -
> 224*u^6 + 480*u^7*t^3*c^2 + 160*u^5*t +
> 252*t^4*u^5*c^2 + 84*t^2*u^3 - 180*t^5*u^4*c^2
> - 60*t^3*u^2:
> factor( % );
```

$$-4\,(8t^2c+7u^2-5ut)\,(3u^2t^2c^2+1)\,(5ct^3-3t^2u-2u^2c^3+8u^4)$$

Again the resulting factors are irreducible over Q.

There exists another procedure, **factors**, that is almost the same as `factor`, but yields the factors in an alternative form.

14.7 More tools for factoring

If necessary, Maple can try to factor a polynomial *in one variable* into a product of linear factors with the procedure **split**, which must be loaded first with `readlib`. Let's apply this to the denominator of the first example of the previous section:

```
> 8*x^5-6*x^4-78*x^3+72*x^2+190*x-210;
```

$$8x^5-6x^4-78x^3+72x^2+190x-210$$

```
> readlib(split) ( % , x );
```

$$8\,(x+\mathrm{RootOf}(\_Z^2-5))\,(x-\mathrm{RootOf}(\_Z^2+\_Z-3))\,(x+1+\mathrm{RootOf}(\_Z^2+\_Z-3))\,(x-\frac{7}{4})\,(x-\mathrm{RootOf}(\_Z^2-5))$$

As you can see, we obtain five linear factors instead of the three factors found with `factor`. However, this procedure `split` is very time-consuming and can only be applied to expressions perceived as polynomials in one variable. As in many other procedures, `split` can work much more efficiently with `RootOf` expressions than radicals, so it may be necessary to apply `convert( ,RootOf)` before this procedure is called.

You might wonder about the factor `x + 1 + RootOf(_Z^2 + _Z - 3)` in the result, but keep in mind that all the occurrences of `RootOf(_Z^2 + _Z - 3)` should be interpreted equally; in this case, if x_1 is one of the roots of x^2+x-3, then the other equals $-1-x_1$, so we get $x^2+x-3=(x+1+x_1)(x-x_1)$.

If you don't like reading expressions containing **RootOfs**, try converting to radicals with **convert(,radical)**:

```
> convert( % , radical );
```

$$8\left(x+\sqrt{5}\right)\left(x+\frac{1}{2}-\frac{1}{2}\sqrt{13}\right)\left(x+\frac{1}{2}+\frac{1}{2}\sqrt{13}\right)\left(x-\frac{7}{4}\right)\left(x-\sqrt{5}\right)$$

(In releases before V.4, `convert( ,radical)` could not do this job; then you could use `allvalues`, but you would see the desired result four times.)

The procedure `split` can handle only polynomials in one variable, but it can use indefinite coefficients:

```
> x^2 - 2*y^4 - 4*y^2 - 2;
```

$$x^2-2y^4-4y^2-2$$

```
> split( % , x );
```

$$\left(x+\text{RootOf}\left(\_Z^2-2y^4-4y^2-2\right)\right)\left(x-\text{RootOf}\left(\_Z^2-2y^4-4y^2-2\right)\right)$$

Moreover, **squarefree factorization** is available with the very fast procedure `convert( ,sqrfree)` and the procedure `sqrfree`. In fact, this is used by `factor` for its initial rough work.

14.8 Solving with numerical tools

If you apply **fsolve** to a polynomial equation, generally you will find all real roots in numerical approximation:

```
> expand( (x^2-2*x-5)*(x^2-2*x+5) );
```

$$x^4-4x^3+4x^2-25$$

```
> fsolve( %=0 , x );
```

$$-1.449489743,\ 3.449489743$$

If you want all complex roots, add the option `complex`:

```
> fsolve( %%=0 , x , complex );
```

$$-1.449489743,\ 1.-2.I,\ 1.+2.I,\ 3.449489743$$

The roots are rendered according to their multiplicity:

```
> fsolve( (x^2-2)^3 , x );
```

$$-1.414213562,\ -1.414213562,\ -1.414213562,\ 1.414213562,\ 1.414213562,\ 1.414213562$$

Generally, `fsolve` can find all roots, but that is not guaranteed. If not, try the option `fulldigits`.

If you are interested only in real solutions in a restricted range, you can ask for these, for instance:

```
> fsolve( x^4-x-1 , x=0..10 );
```

$$1.220744085$$

For systems of polynomial equations, `fsolve` tries to find one solution, not all solutions. For instance, let's solve the system of section 14.2 on page 171:

```
> fsolve( {circle1,circle2} , {x,y} );
```

$$\{y = 3.741865932,\ x = 7.475597795\}$$

In order to find the other solution, you can try to help by entering other starting values or by entering ranges for the variables; both are demonstrated here:

```
> fsolve( {circle1,circle2} , {x=-1,y=-1} );
```

$$\{x = -4.025597795,\ y = -.09186593167\}$$

```
> fsolve( {circle1,circle2} , {x=-10..0,y=-10..0} );
```

$$\{x = -4.025597795,\ y = -.09186593167\}$$

14.9 Solving complicated systems of polynomial equations with Gröbner basis

For the solution of systems of polynomial equations, Maple provides the procedure `Groebner[gsolve]`, which combines factorization with the Buchberger algorithm for calculation of a Gröbner base. (In releases before V.5, use `grobner` instead of `Groebner`.)

The Buchberger algorithm is a powerful tool, based on the idea of elimination of leading terms (terms of highest degree in some interpretation of "highest"). This is the same idea as used in finding the intersection points of two circles, where first the difference of the two equations is calculated in order to get rid of the quadratic terms. Additionally, the factorization in `Groebner[gsolve]` can bring about a splitting into several systems. Here is a simple example, which cannot reveal the structure and the power of the Buchberger algorithm, but can clarify the splitting action of factorization and the result.

If you want to solve the following system of equations in x and y:

$$\left\{\begin{array}{c} x^2y^2 + 12y = 3x^2 + 4y^3 \\ x^2 + x = y^2 \end{array}\right\}$$

you can try to manipulate this into a more accessible system as follows. First convert the first equation into the form " $= 0$" and factor the result:

$$\Leftrightarrow \left\{\begin{array}{c} (y^2 - 3)(x^2 - 4y) = 0 \\ x^2 + x - y^2 = 0 \end{array}\right\} \Leftrightarrow$$

Consequently, the systems splits into two systems:

$$\left\{\begin{array}{c} y^2=3 \\ x^2+x-y^2=0 \end{array}\right\} \quad \text{or} \quad \left\{\begin{array}{c} x^2=4y \\ x^2+x-y^2=0 \end{array}\right\} \Leftrightarrow$$

In the right system, y can be eliminated by setting $y=\frac{x^2}{4}$ in the left system, $y^2=3$ can be substituted in the second equation, yielding:

$$\Leftrightarrow \left\{\begin{array}{c} y^2=3 \\ x^2+x-3=0 \end{array}\right\} \quad \text{or} \quad \left\{\begin{array}{c} x^2=4y \\ x^2+x-(x^2/4)^2=0 \end{array}\right\} \Leftrightarrow$$

The right system splits into two systems because the second equation has a factor x:

$$\Leftrightarrow \left\{\begin{array}{c} y^2=3 \\ x^2+x-3=0 \end{array}\right\} \quad \text{or} \quad \left\{\begin{array}{c} x^2=4y \\ x=0 \end{array}\right\} \quad \text{or} \quad \left\{\begin{array}{c} x^2=4y \\ x+1-x^3/16=0 \end{array}\right\}$$

This is essentially what you find when you apply **Groebner[gsolve]** to the system at the beginning:

```
> x^2*y^2+12*y=3*x^2+4*y^3;
```

$$x^2y^2+12y=3x^2+4y^3$$

```
> x^2+x=y^2;
```

$$x^2+x=y^2$$

```
> eq1:=%%: eq2:= %%:
```

In releases before V.5 you can enter

```
> grobner[gsolve]( {eq1,eq2} , { x,y} );
```

but `Groebner[gsolve]` does not accept polynomial equations; they must be converted into polynomials first:

```
> map( eq->lhs(eq)-rhs(eq) , {eq1,eq2} );
```

$$\{x^2+x-y^2,\ x^2y^2+12y-3x^2-4y^3\}$$

```
> Groebner[gsolve]( % , {x,y} );
```

$$\Big\{[[y^2-3,\ x^2+x-3],\ \mathrm{plex}(x,y),\ \{\}],$$
$$[[y,\ x],\ \mathrm{plex}(x,y),\ \{\}],$$
$$[[-16x-16+x^3,\ 4y-x^2],\ \mathrm{plex}(y,x),\ \{x^2+x-3,\ x\}]\Big\}$$

The result is a set of three lists, each containing

- a system of equations (without $=0$)
- the order of the equation variables used in the calculation (useful information if you want to try other orders)
- polynomials that are assumed to be unequal to zero in the solution of that result system

Omitting the second and third elements of the lists we see that the result is a set of three systems of equations:

$$\left\{ \begin{array}{c} y^2-3=0 \\ x^2+x-3=0 \end{array} \right\} \quad \text{or} \quad \left\{ \begin{array}{c} y=0 \\ x=0 \end{array} \right\} \quad \text{or} \quad \left\{ \begin{array}{c} -16x-16+x^3=0 \\ 4y-x^2=0 \end{array} \right\}$$

which is equivalent to the result we found earlier by hand.

After this simplification of the original system, we can apply `solve` to each of the resulting systems of equations. Let's name the result `gsys` before continuing the manipulations.

```
> gsys := %:
```

The result of `Groebner[gsolve]` renders each system of equations as a *list* of polynomials, but since `solve` wants a *set* of equations as its first argument, we must convert first. Let's take the third system, and convert this into a set:

```
> gsys[3];
```

$$\left[\left[-16x-16+x^3,\ 4y-x^2 \right],\ \mathrm{plex}(y,x),\ \left\{ x^2+x-3,\ x \right\} \right]$$

```
> %[1];
```

$$\left[-16x-16+x^3,\ 4y-x^2 \right]$$

```
> { op(%) };
```

$$\left\{ 4y-x^2,\ -16x-16+x^3 \right\}$$

This set can be tackled with `solve`.

```
> solve( % , {x,y} );
```

$$\left\{ x = 2\ \mathrm{RootOf}\left(-4\_Z-2+\_Z^3\right),\ y = \mathrm{RootOf}\left(-4\_Z-2+\_Z^3\right)^2 \right\}$$

This is the third part of the solution, so let's save the result as

```
> sol3:=%:
```

The other systems can be handled the same way.

```
> sol1 := solve( {op(gsys[1][1])} , {x,y} ):
> sol2 := solve( {op(gsys[2][1])} , {x,y} ):
```

These solutions are joined together to find the solutions of the original system of equations.

```
> sol1, sol2, sol3;
```

$$\Big\{ y = \mathrm{RootOf}\big(\_Z^2 - 3\big)\,,\ x = \mathrm{RootOf}\big(\_Z^2 + \_Z - 3\big) \Big\},$$
$$\Big\{ x = 0,\ y = 0 \Big\},$$
$$\Big\{ x = 2\ \mathrm{RootOf}\big(-4\,\_Z - 2 + \_Z^3\big)\,,$$
$$y = \mathrm{RootOf}\big(-4\,\_Z - 2 + \_Z^3\big)^2 \Big\}$$

Now we can apply `allvalues`.

We could have solved this system with `solve` faster, but with this method more complicated systems of polynomial equations can be solved, where often the pure application of `solve` would take more time, possibly without finding any solution at all, or one less neat.

You can influence the solving process by passing a list of variables as the second parameter in `Groebner[gsolve]`, and choosing the order of these.

The `Groebner` package contains more tools for systems of equations, especially `Groebner[solvable]` and `Groebner[finite]`, which decide if a given system of polynomial equations is algebraically consistent, and if the system has a finite number of solutions.

Maple also uses the same Gröbner basis methods for simplifications with side relations. See section 11.11 on page 147 and section 14.9 on page 180.

14.10 Algebraic extensions of the rational number field

The elementary number field for Maple is the field of the rational numbers. This field can be extended with roots of polynomials, but doing so makes many calculations more complex, so Maple must be coaxed into it. For instance,

```
> (x^3 - 5)/(x - 5^(1/3));
```

$$\frac{x^3 - 5}{x - 5^{(1/3)}}$$

The denominator is a factor of the numerator, but **normal** does not see:

```
> normal( % );
```

$$\frac{x^3 - 5}{x - 5^{(1/3)}}$$

Even though this expression contains $\sqrt[3]{5}$, `normal` does not use this **radical** for simplification of the quotient. To make Maple do that, we must convert the radicals to `RootOf` expressions first.

```
> convert( % , RootOf );
```

$$\frac{x^3 - 5}{x - \mathrm{RootOf}(\_Z^3 - 5)}$$

We can order simplification in the following way:

```
> evala(Normal( % ));
```

$$x^2 + \text{RootOf}(\_Z^3 - 5)\, x + \text{RootOf}(\_Z^3 - 5)^2$$

The name of the procedure **evala** is an abbreviation of "evaluate in an algebraic extension field". The procedure `Normal` is the inert version of `normal`: it does nothing itself, but it tells `evala` what is to be done.

The `RootOf` expression can now be converted into a radical.

```
> convert( % , radical );
```

$$x^2 + 5^{(1/3)}\, x + 5^{(2/3)}$$

The procedure `evala` can make several types of calculations involving `RootOf` expressions in combination with inert procedures such as `Normal`, `Expand`, `Factor`, `Factors`, `Quo`, `Resultant`, etc.

```
> evala( Expand( (x - RootOf(t^3 - 5*t - 3,t))^3 ) );
```

$$x^3 - 3x^2\, \text{RootOf}(\_Z^3 - 5\_Z - 3) + 3x\, \text{RootOf}(\_Z^3 - 5\_Z - 3)^2 - 3 - 5\, \text{RootOf}(\_Z^3 - 5\_Z - 3)$$

In order to make the above result more readable, we can use **alias**:

```
> alias( RT = RootOf(t^3 - 5*t - 3,t) );
```

$$I,\ RO,\ RT$$

The procedure `alias` reports which names are aliased: the name `RO` has been used earlier in this chapter. The `alias` construction is discussed in section B.1 on page 285.

Now we achieve a nicer representation of the previous result.

```
> %% ;
```

$$x^3 - 3x^2\, RT + 3x\, RT^2 - 3 - 5\, RT$$

We can ask for factorization with `evala` and the inert factorization procedure `Factor`:

```
> evala(Factor(%));
```

$$(x - RT)^3$$

In the last command, Maple profits from the fact that the expression to be factored already contains the necessary `RootOf` expression. If that is not the case, it must be added as an extra argument to `Factor`.

```
> evala(Factor(x^3-5));
```

$$x^3 - 5$$

```
> evala(Factor( x^3-5 , RootOf(t^3-5,t) ));
```

$$\left(x - \text{RootOf}\left(\_Z^3 - 5\right)\right)$$
$$\left(x^2 + \text{RootOf}\left(\_Z^3 - 5\right) x + \text{RootOf}\left(\_Z^3 - 5\right)^2\right)$$

The algorithm used by `Factor` can be chosen by the user; consult the on-line help about `Factor`.

In the on-line help you can find a listing of the inert procedures that can be handled with `evala`.

In the last case, **factor** can do the job as well.

```
> factor( x^3-5 , RootOf(t^3-5,t));
```

$$\left(x - \text{RootOf}\left(\_Z^3 - 5\right)\right)$$
$$\left(x^2 + \text{RootOf}\left(\_Z^3 - 5\right) x + \text{RootOf}\left(\_Z^3 - 5\right)^2\right)$$

The combination of `Factor` with `evala` can do much more, such as managing powers of `RootOf` expressions, but `factor` offers another possibility, using a radical as its second argument. In this case, `factor` can use $\sqrt[3]{5}$ as the second argument.

```
> factor( x^3-5 , 5^(1/3) );
```

$$\left(x - 5^{(1/3)}\right)\left(x^2 + 5^{(1/3)}\, x + 5^{(2/3)}\right)$$

The procedure `factor` calls the procedure `evala` automatically in such a case, and converts resulting `RootOf`s back into radicals. The second argument must be a radical, for instance `sqrt(-2)` does not work: it is evaluated to $I\sqrt{2}$, which is not allowed, but `(-2)^(1/2)` can be used as a second argument.

If `evala` has to handle an expression that contains a `RootOf` expression, it tests if it is irreducible:

```
> evala( (RootOf(x^4+5*x,x) * RootOf(x^2+3))^4 );
Error, reducible RootOf detected.  Substitutions are
{RootOf(_Z^4+5*_Z) = 0, RootOf(_Z^4+5*_Z) = RootOf(_Z^3+5)}
```

If you want to solve polynomial equations in an algebraic number field, to be defined by yourself, you can use **roots**. For instance, let's extend the field of the rationals with the roots of $u^2 + u - 1 = 0$:

```
> alias(gs=RootOf(u^2+u-1,u));
```

$$I,\ RO,\ RT,\ gs$$

Now we can solve the equation $x^2 = 5$ in this field. The procedure `roots` does not accept equations; we must take left-hand side minus right-hand side as the first argument:

```
> roots(x^2-5,x,gs);
```

$$[[2\,gs + 1,\ 1],\ [-1 - 2\,gs,\ 1]]$$

14.11 Polynomial rings modulo ideals

For calculations with algebraic extensions in the setting of a polynomial ring modulo the ideal generated by one or more polynomials you can use `simplify` with **side relations**. This idea is discussed in section 11.11 on page 147. Here is a simple example that demonstrates the idea in the present context. The same example is handled in a different way in section 14.4 on page 175. We use the equality:

```
> eq;
```

$$x^5 - t\,x^2 + p = 0$$

We can calculate x^7 modulo the corresponding polynomial as follows:

```
> simplify( x^7 , {eq} , {x} );
```

$$x^4\,t - x^2\,p$$

We have reduced x^7 (first argument) as a polynomial in x (last argument) modulo the ideal generated by $x^5 - t\,x^2 + p$ (second argument). Much more complicated problems containing several variables and several equations can be handled efficiently thanks to the fact that Maple applies *Gröbner basis* methods. For details, please consult the on-line help about `simplify[siderel]`.

A package `Domains` is available for more general purposes. See section A.11 on page 283.

14.12 Polynomials over Z mod p

Maple can perform calculations on polynomials with coefficients in Z mod p as follows:

```
> poly := 20*x^7 - 7*x^6 + 28*x^5 + 5*x - 1;
```

$$poly := 20\,x^7 - 7\,x^6 + 28\,x^5 + 5\,x - 1$$

```
> poly mod 5;
```

$$3\,x^6 + 3\,x^5 + 4$$

The last result is not represented internally in Maple as a polynomial over Z mod 5 but as a polynomial over Q. So in the following command Maple does not handle this result as a polynomial over Z mod 5.

```
> expand( %^2 );
```

$$9x^{12} + 18x^{11} + 24x^6 + 9x^{10} + 24x^5 + 16$$

For each calculation mod 5 we must append `mod 5` to the command again. If you don't like that, you might prefer to use the `Domains` package with `Zmod`; see section A.11 on page 283.

```
> expand( %%^2 ) mod 5;
```

$$4x^{12} + 3x^{11} + 4x^6 + 4x^{10} + 4x^5 + 1$$

Modulo factorization is possible with **Factor**.

```
> Factor( % ) mod 5;
```

$$4\,(x+4)^2\,\left(x^4 + 2x^2 + 3x + 1\right)^2\,(x+2)^2$$

For solving an equation mod p, **msolve** can be used.

```
> msolve( poly = 0 , 5 );
```

$$\{x = 1\},\ \{x = 3\}$$

For polynomials, **Roots** can be used, too, in combination with `mod`.

The equivalents of several procedures for manipulating polynomials and rational expressions, such as `normal` and `gcd`, are available as well.

```
> Normal( (x^2 - 3)/(x^3+7) ) mod 11;
```

$$\frac{x+5}{x^2+5x+3}$$

```
> Gcd( x^2 - 3 , x^3+7 ) mod 11;
```

$$x + 6$$

More information can be found in the on-line help about `mod`.

Maple can handle algebraic extensions of Z mod p; information about this subject can be found in the on-line help to `evalgf`, the `GF` package for **Galois fields**, and the **Domains** package for creating domains, discussed in section A.11 on page 283.

chapter 15

Manipulating algebraic expressions

Manipulation of numbers, polynomials, rational expressions, and `RootOf` *expressions is discussed in Chapter 2,* Numbers and algebraic operators, *in Chapter 13,* Polynomials and rational expressions, *and in Chapter 14,* Polynomial equations and factoring polynomials.
This chapter continues discussing manipulation, but extends to more general types of algebraic expressions. The usual goal of manipulating algebraic expressions is conversion of an expression to a "simple" expression. But what is "simple" depends on what you want to do with the expression. Therefore, inverse actions are shown for most of the actions discussed.

The first section on using options for `simplify` *and* `combine` *is followed by sections on manipulating powers and radicals, expressions with* exp *and* ln, *and trigonometric expressions. Although there are facilities for using properties of several other mathematical functions, we do not try to cover them all; if you have seen these cases, you will be able to find your way yourself.*

Restricting the action of manipulations to a part of an expression is discussed extensively in a special section.

If you want to see if an equality is true, you might use other methods than symbolic manipulation. These are shown in a special section on this subject.

Examples are kept as small as possible. For such cases, manipulating by hand is much simpler. But the same ideas can be used for manipulating very large formulas and then using these tools can pay the efforts. Two sections each give an example of manipulating a not so simple expression.

In the Short Reference List at the start of this book is a survey of the main manipulation tools.

15.1 Options for `simplify` and `combine`

The general manipulation tools `simplify` and `combine` can do many things at once, for instance using properties of functions such as exp, ln, sin, cos, etc., and/or using properties of powers. In many cases it is desirable not to use all these properties, but to select the type of transformation wanted. Restricting the action of `simplify` or `combine` can be done by using options.
(In releases before V.4, `combine` must be activated with options; without options it can do only a few manipulations.)
Here is an example:

```
> testexpr:=exp(sin(x)^4)/exp(cos(x)^2);
```

$$testexpr := \frac{e^{(\sin(x)^4)}}{e^{(\cos(x)^2)}}$$

For instance, to restrict `combine` to applying the properties of the function exp, use the option `exp`:

```
> combine( testexpr , exp ) ;
```

$$e^{(\sin(x)^4-\cos(x)^2)}$$

Using the option `power` yields the same result. But using the options `trig` or `sin` or `cos` causes actions to trigonometric functions:

```
> combine( testexpr , trig );
```

$$\frac{e^{\left(\frac{3}{8}+\frac{1}{8}\cos(4x)-\frac{1}{2}\cos(2x)\right)}}{e^{\left(\frac{1}{2}\cos(2x)+\frac{1}{2}\right)}}$$

There are many more options available for `combine`. See the *Short Reference List* at the start of the book for a full list of options.

It is also possible to apply `combine` without options:

```
> combine( testexpr );
```

$$e^{\left(-\frac{1}{8}+\frac{1}{8}\cos(4x)-\cos(2x)\right)}$$

If you want to polish an expression without special purposes, you can experiment with `simplify`, `combine` without options and the other tools discussed in this chapter (especially `expand`, `normal`, `factor`, and `rationalize` and see what you get. If you are not satisfied with the result, look for patterns in the expression and decide what you want to do; then use the right procedure with the right option. In many cases, several steps in the right order are necessary. Some of these more complicated examples are demonstrated in this chapter.

To show the specific actions of procedures in combination with options, the relevant option is added in each example of this chapter, although in most of the basic examples you would get the same result without that option. In several cases, alternatives are shown.

15.2 Simplifications depending on conditions

Expected simplifications may not be executed. Here is another example:

```
> exp(X)^n*exp(Y);
```

$$\left(e^{X}\right)^{n} e^{Y}$$

Let's try to get $e^{n\,X+Y}$ by applying `combine` or `simplify`:

```
> combine(%);
```

$$\left(\mathrm{e}^{X}\right)^{n}\,\mathrm{e}^{Y}$$

```
> simplify(%);
```

$$\left(\mathrm{e}^{X}\right)^{n}\,\mathrm{e}^{Y}$$

No effects! There is a good reason for that: `X`, `Y`, and `n` are to be taken as unknown complex numbers. It may not be true that $\left(\mathrm{e}^{X}\right)^{n} = \mathrm{e}^{n\,X}$; for instance, if $X = \frac{3}{2}\pi$ and $n = \frac{1}{2}$, then

$$\exp(X)^{n} = \exp(\frac{3}{2}\pi)^{(1/2)} = (-i)^{(1/2)} = \frac{1}{2}\sqrt{2}(1-i)$$

$$\exp(n\,X) = \exp(\frac{3}{4}\pi) = \frac{1}{2}\sqrt{2}(-1+i)$$

But if `X` stands for a real number, then $\left(\mathrm{e}^{X}\right)^{n} = \mathrm{e}^{n\,X}$. In that case, we must tell Maple that `X` is a real variable; then `simplify` without option or with the option `power` can do the job:

```
> assume(X,real);
> simplify(%,power);
```

$$\mathrm{e}^{(n\,X\sim+Y)}$$

The trailing tilde indicates that `X` has a property; see section 3.4 on page 36, or section A.4 on page 278.

It is possible to make `simplify` really foolhardy by adding the (euphemistic) option **`symbolic`**:

```
> exp(x)^n*exp(y);
```

$$\left(\mathrm{e}^{x}\right)^{n}\,\mathrm{e}^{y}$$

```
> simplify(%,power,symbolic);
```

$$\mathrm{e}^{(n\,x+y)}$$

We could have used `simplify(%,symbolic)` as well. The option `symbolic` may be used also for `combine`, but only together with other options.

If you are trying to simplify some expression without success, you might use the option `symbolic` and, if it yields a nice result, check if this result is equivalent to the original expression (see section 15.12 on page 212), or, for instance, if you are trying to find an antiderivative, a solution of an equation, or of a differential equation, check if the result satisfies. An example can be found in section 15.8 on page 201.

We could also have told Maple that the variables are real:

```
> simplify(%%,power,assume=real);
```

$$\mathbf{e}^{(n\,x+y)}$$

This simplification rule can also be applied if the exponent is an integer:

```
> assume(n,integer);
> exp(x)^n*exp(y);
```

$$(\mathbf{e}^{x})^{n\sim}\ \mathbf{e}^{y}$$

```
> simplify(%,exp);
```

$$\mathbf{e}^{(n\sim\,x+y)}$$

If you are still using release V.3 or earlier, the suggestions above are not valid. The improvements of the reliability of expression manipulation from the first release of Maple V untill the present release V.5 are enormous, now making Maple V.5 a superior choice for reliable mathematical calculations. A system of properties has been built in; many procedures using rules that depend on properties can now check properties fast and efficiently and will apply these rules only when it can be determined that the necessary conditions are fulfilled. Up to a few marginal exceptions, mentioned in the last section of this chapter, you can trust the manipulations discussed in this chapter if you are using release V.5. If you are using release V.3 or even an earlier release, the best advice is to grade up.

15.3 Sums of exponents, products of powers with equal basis

For each complex number z, each a, each b, and each c

$$z^{a+b-c} = \frac{z^a\,z^b}{z^c}$$

Manipulation from the left-hand side to the right-hand side is possible with **expand**:

```
> z^(a+b-c);
```

$$z^{(a+b-c)}$$

```
> expand(%);
```

$$\frac{z^a\,z^b}{z^c}$$

However, `expand` does many things at once. There are facilities for restricting the action of `expand`, but it might be easier to use an alternative. You can get an only slightly different result with **simplify(,commonpow)** (not documented by Maple); this result can even be converted into the previous result with **normal(,expanded)**:

```
> simplify(%%,commonpow);
```

$$z^a\, z^b\, z^{(-c)}$$

```
> normal(%,expanded);
```

$$\frac{z^a\, z^b}{z^c}$$

The reverse direction can be produced with **simplify(,power)** or **combine (,power)**:

```
> simplify( % , power );
```

$$z^{(a+b-c)}$$

```
> combine( %% , power );
```

$$z^{(a+b-c)}$$

Halfway back brings us **combine(,cmbpwr)**:

```
> combine( %%% , cmbpwr );
```

$$z^a\, z^b\, z^{(-c)}$$

Although the expression `exp(x)` has the structure of a function call, not of a power, it is printed as e^x in windowing versions and it can be handled with some of the manipulation tools for powers: `simplify( ,power)`, `combine( ,power)`, and `expand`. For instance,

```
> exp(x)*exp(y)/exp(z);
```

$$\frac{\mathrm{e}^x\, \mathrm{e}^y}{\mathrm{e}^z}$$

```
> simplify(%,power);
```

$$\mathrm{e}^{(x+y-z)}$$

If action is to be restricted to exponential function calls, try the procedures **simplify(,exp)** and **combine(,exp)**:

```
> exp(a)*exp(b)+x^a*x^b; simplify(%,exp);
```

$$\mathrm{e}^a\, \mathrm{e}^b + x^a\, x^b$$

$$\mathrm{e}^{(a+b)} + x^a\, x^b$$

15.4 Powers of powers, products of exponents

The identity

$$(z^a)^b = z^{a\,b}$$

is true

- if b is an integer, or
- if z is positive and a is real

but in many other cases it is not true, for instance,

$$((-1)^2)^{(1/2)} \neq (-1)^{(2*1/2)}$$

Conversion from the left-hand side to the right-hand side is possible with **expand**, **simplify(,power)**, and **combine(,power)**, but only when Maple can see that the conversion is correct (from release V.3 for `simplify`, from release V.4 for `expand`). The following is correct because 5 is an integer:

```
> (z^a)^5; simplify(%,power);
```

$$(z^a)^5$$

$$z^{(5\,a)}$$

The following is correct because $\pi - 1$ is positive:

```
> ((Pi-1)^(7/4))^(1+I); simplify(%,power);
```

$$\left((\pi - 1)^{(7/4)}\right)^{(1+I)}$$

$$(\pi - 1)^{\left(\frac{7}{4}+\frac{7}{4}\,I\right)}$$

If we ask the same in a more general case, Maple refuses to perform this type of conversion:

```
> (z^x)^y; simplify(%,power);
```

$$(z^x)^y$$

$$(z^x)^y$$

But you can tell Maple that your variables have properties:

```
> assume(pos,positive,X,real,n,integer);
> (pos^X)^y; simplify(%,power);
```

$$(pos\sim^{X\sim})^y$$

$$pos\sim^{(X\sim\,y)}$$

```
> (z^x)^n; simplify(%,power);
```

$$(z^x)^{n\sim}$$

$$z^{(x\,n\sim)}$$

You can also tell Maple not to bother about correctness with the additional option `symbolic`:

```
> simplify((z^x)^y,power,symbolic);
```

$$z^{(x\,y)}$$

It is also possible to tell `simplify` that all variables are to be taken as positive numbers:

```
> simplify((z^x)^y,power,assume=positive);
```

$$z^{(x\,y)}$$

Exponential function calls can be handled with **simplify(,power)** as well:

```
> exp(X)^y + exp(a)^n; simplify(%,power);
```

$$\left(\mathbf{e}^{X\sim}\right)^{y} + \left(\mathbf{e}^{a}\right)^{n\sim}$$

$$\mathbf{e}^{(X\sim\,y)} + \mathbf{e}^{(n\sim\,a)}$$

If action is to be restricted to exponential function calls, the procedure `simplify( ,exp)` can be used instead of `simplify( ,power)`, but only for the case in which the exterior exponent is an integer:

```
> exp(X)^b + exp(a)^n; simplify(%,exp);
```

$$\left(\mathbf{e}^{X\sim}\right)^{b} + \left(\mathbf{e}^{a}\right)^{n\sim}$$

$$\left(\mathbf{e}^{X\sim}\right)^{b} + \mathbf{e}^{(n\sim\,a)}$$

Manipulation in the reverse direction is available only for cases such as:

```
> z^(5*a); simplify(%,commonpow);
```

$$z^{(5\,a)}$$

$$\left(z^{5}\right)^{a}$$

```
> expand(%%);
```

$$\left(z^{a}\right)^{5}$$

In the last case, `expand` acts as its own inverse:

```
> expand(%);
```

$$\left(z^{a}\right)^{5}$$

For more general cases, you can convert for instance $p^{X\,b}$ into $p^{X}\,p^{b}$, assuming that p is a positive number and X is real, as follows:

```
> p^(X*b); subs( p=pos^(1/X) , % );
```

$$p^{(X\sim b)}$$

$$\left(pos\sim^{\left(\frac{1}{X\sim}\right)}\right)^{(X\sim b)}$$

Now we need the assumptions on `X` and `pos`:

```
> expand( % );
```

$$pos\sim^{b}$$

Finish by substituting back:

```
> subs( pos=p^a , % );
```

$$(p^{a})^{b}$$

Obviously, the user must take responsibility for the correctness of this conversion.

In order to change $\mathbf{e}^{(a\,b)}$ into $(\mathbf{e}^{a})^{b}$ we can use the method above, but it can be done easier by manipulating `exp` and `ln`:

```
> exp(a*b); subs(a=ln(t),%);
```

$$\mathbf{e}^{(a\,b)}$$

$$\mathbf{e}^{(\ln(t)\,b)}$$

```
> expand(%);
```

$$t^{b}$$

```
> subs(t=exp(a),%);
```

$$(\mathbf{e}^{a})^{b}$$

15.5 Powers of products, products of powers with equal exponents

The identity

$$x^{a}\,y^{a} = (x\,y)^{a}$$

is true

- if x or y is positive or
- a is an integer

but not in general. For instance,

$$(-1)^{1/2}\,(-1)^{1/2} \neq 1^{1/2}$$

First, let's look at the case where the base of the power is assumed to be positive:

```
> assume(pos>0);
```

The procedure `simplify( ,power)` can use this property:

```
> (pos*z)^a; simplify(%,power);
```

$$(pos\sim\ z)^a$$

$$pos\sim^a\ z^a$$

In the present release V.5, no general tools are available for conversion of $(xy)^n$ into $x^n y^n$ for the case in which n is assumed to be an integer, (and x and y have no properties), apart from using the option `symbolic`. However, if the exponent is a concrete integer, automatic simplification does the job immediately:

```
> ( x*y ) ^ 5;
```

$$x^5\, y^5$$

The reverse is possible with the aid of **simplify(,commonpow)** (not documented):

```
> pos^a * z^a; simplify(%,commonpow);
```

$$pos\sim^a\ z^a$$

$$(pos\sim\ z)^a$$

```
> x^n * y^n ; simplify(%,commonpow);
```

$$x^{n\sim}\, y^{n\sim}$$

$$(x\, y)^{n\sim}$$

15.6 Radicals

In cases in which the exponent of a power is a concrete rational number, some conversions can be executed by the procedure **simplify(,radical)**. First, it splits off the integer powers in some cases:

```
> (x*y)^(22/7); simplify(%,radical);
```

$$(x\, y)^{(22/7)}$$

$$x^3\, y^3\ (x\, y)^{(1/7)}$$

Then it tries to use properties of the power operation, for instance here, where `pos` has been previously assumed to be a positive number:

```
> (pos^5)^(7/5); simplify(%,radical);
```

$$\left(pos\sim^{5}\right)^{(7/5)}$$

$$pos\sim^{7}$$

Then it tries to combine equal radicals:

```
> (a-b)*(z+1)^(1/4)+b*(z+1)^(1/4); simplify(%,radical);
```

$$(a-b)\ (z+1)^{(1/4)}+b\ (z+1)^{(1/4)}$$

$$(z+1)^{(1/4)}\ a$$

At last, `simplify( ,radical)` calls the procedure `normal`:

```
> pos^2*(z+1)^(2/5)+(z+1)^(-3/5); simplify(%,radical);
```

$$pos\sim^{2}\ (z+1)^{(2/5)}+\frac{1}{(z+1)^{(3/5)}}$$

$$\frac{pos\sim^{2}\ z+pos\sim^{2}+1}{(z+1)^{(3/5)}}$$

For square roots, you can use **simplify(,sqrt)**.

The results obtained with `simplify( ,radical)` are quite different from the results obtained with `simplify( ,power)`:

```
> ((p+1)^(7/4))^(5/3);
```

$$\left((p+1)^{(7/4)}\right)^{(5/3)}$$

```
> simplify(%,power); simplify(%%,radical);
```

$$\left((p+1)^{(7/4)}\right)^{(5/3)}$$

$$(p+1)^{(7/4)}\ \left((p+1)^{(7/4)}\right)^{(2/3)}$$

For elimination of radicals from the denominator of a quotient the procedure **rationalize** is available.

```
> (a+b)/(c+sqrt(d));
```

$$\frac{a+b}{c+\sqrt{d}}$$

```
> rationalize(%);
```

$$\frac{(a+b)\left(c-\sqrt{d}\right)}{c^{2}-d}$$

Here is another example of a quotient:

```
> (w-2)^(3/2)/(w^2-4*w+4)^(1/4);
```

$$\frac{(w-2)^{(3/2)}}{(w^2-4\,w+4)^{(1/4)}}$$

We can try to divide out the common factor from numerator and denominator. Usually such a job can be done with `normal`, but this is a conscientious procedure and cannot handle the present expression, as it does not know if `w` is a real number greater or less than 2. However, the procedure **radsimp** is not scrupulous: according to its on-line help, it chooses branches according to its own rules:

```
> radsimp( % );
```

$$w-2$$

The same result can be found with `simplify( ,symbolic)` or with `combine ( ,radical,symbolic)` followed by `normal`.
If you want to have control yourself, there are better ways of manipulation. Let's help Maple and apply `factor`:

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

$$\frac{(w-2)^{(3/2)}}{\left((w-2)^2\right)^{(1/4)}}$$

We get a product of two powers with equal basis. The procedure `simplify ( ,power)` refuses to work, unless we tell Maple for instance that $w < 2$:

```
> assume(w<2);
> simplify(%,power);
```

$$-I\,(w\sim -2)$$

(In release V.4 a wrong result is rendered: $I\,(w\sim -2)$.)

Manipulating the nth root of a complex number is discussed in section 12.5 on page 153.

15.7 Manipulating logarithmic expressions

If x, y, and z are positive, then $\ln(x*y/z) = \ln(x)+\ln(y)-\ln(z)$; it is even correct if two of the three variables are positive, but often this rule cannot be applied (from release V.4). For instance, if $x = y = -1 + I$ and $z = 1$:

```
> [ ln(-1+I) + ln(-1+I) , ln( (-1+I)*(-1+I) ) ];
```

$$[\,2\,\ln(-1+I),\ \ln(-2\,I)\,]$$

```
> evalc(%);
```

$$\left[\ln(2) + \frac{3}{2} I\pi,\ \ln(2) - \frac{1}{2} I\pi\right]$$

The procedure **simplify(,ln)** converts the logarithm of a product or quotient of two numbers into a sum of logarithms only if it can decide that one of the two numbers is a positive number or that it is a negative number (from release V.4).

```
> ln(pos*y); simplify(%,ln);
```

$$\ln(pos\sim\ y)$$

$$\ln(pos\sim) + \ln(y)$$

```
> assume(neg,negative);
> ln(x/neg); simplify(%,ln);
```

$$\ln\left(\frac{x}{neg\sim}\right)$$

$$-\ln(-neg\sim) + \ln(-x)$$

The same results can be found with **expand**; sometimes you also can use **simplify(,power)**.

For the opposite direction, you can use **combine(,ln)**,

```
> ln(pos)+ln(a);combine(%,ln);
```

$$\ln(pos\sim) + \ln(a)$$

$$\ln(pos\sim\ a)$$

```
> ln(a)-ln(pos);combine(%,ln);
```

$$\ln(a) - \ln(pos\sim)$$

$$\ln\left(\frac{a}{pos\sim}\right)$$

The following transformation can be handled only with the aid of the hazardous option `symbolic`:

```
> ln(pos)-ln(a);combine(%,ln,symbolic);
```

$$\ln(pos\sim) - \ln(a)$$

$$\ln\left(\frac{pos\sim}{a}\right)$$

The rule $\ln(a^b) = b\ln(a)$ is only true for some special cases. When Maple can see that the exponent b is an integer and that a is positive, then it converts $\ln(a^b)$ into

$b \ln(a)$ automatically:

```
> ln(pos^n);
```

$$n\sim \ln(pos\sim)$$

In other cases where this transformation from the left to the right is correct, you can use **simplify(,ln)** or **expand**:

```
> ln(pos^X); simplify(",ln);
```

$$\ln\left(pos\sim^{X\sim}\right)$$

$$X\sim \ln(pos\sim)$$

The other way around is available only for the case in which b is a concrete number:

```
> (2/3)*ln(pos+1)-5*ln(pos+2);
```

$$\frac{2}{3}\ln(pos\sim+1)-5\ln(pos\sim+2)$$

```
> combine( % ,ln);
```

$$\ln\left(\frac{(pos\sim+1)^{(2/3)}}{(pos\sim+2)^5}\right)$$

If you don't like exponents that are not an integer, you can request to avoid these with an extra option:

```
>  combine((2/3)*ln(pos+1)-5*ln(pos+2),ln,integer);
```

$$\frac{2}{3}\ln(pos\sim+1)-5\ln(pos\sim+2)$$

The function exp is the left inverse of ln; restricted to the real numbers it is the right inverse as well. Automatic simplification uses this:

```
> exp(ln(x));
```

$$x$$

```
> assume(X,real):
> ln(exp(X));
```

$$X\sim$$

For all complex numbers a, x, y, and z with x, y, and z unequal to 0:

$$\exp(a\ln(x)+\ln(y)-ln(z))=\frac{x^a y}{z}$$

$$\exp(y*\ln(x))=x^y$$

The conversion from the left to the right can be executed with `simplify` with the option `power` or `exp` or with `combine( ,exp)`:

```
> exp(a*ln(x) + ln(y) - ln(z));
```

$$\mathbf{e}^{(a\,\ln(x)+\ln(y)-\ln(z))}$$

```
> simplify(%,exp);
```

$$\frac{x^a\,y}{z}$$

For all real x: $\ln(\exp(x)) = x$, but, for instance, $\ln(\exp(1+2\pi i)) = \mathbf{e}$. Don't use the manipulations of this chapter for this; it can be found by **evalc**:

```
> ln(exp(1+2*I*Pi));
```

$$\ln\left(\mathbf{e}^{(1+2\,I\,\pi)}\right)$$

```
> simplify(%);
```

$$\ln\left(\mathbf{e}^{(1+2\,I\,\pi)}\right)$$

```
> evalc(%%);
```

$$1$$

However, `evalc` expects all variables to be real, so it does the same simplification as `simplify( ,ln)` for the case of undefined arguments:

```
> ln(exp(z)); evalc(%);
```

$$\ln(\mathbf{e}^z)$$

$$z$$

Not for all complex z is this correct, so don't use `evalc` if your variables are not all real!

15.8 An example of the use of the option symbolic

In Chapter 4, *Elementary calculus* we found:

```
> integrand:=1/(x^(1/2)+x^(1/3));
```

$$integrand := \frac{1}{\sqrt{x}+x^{(1/3)}}$$

```
> int( % , x );
```

$$2\sqrt{x}-2\,\operatorname{arctanh}\left(\sqrt{x}\right)-\ln(x-1)-3\,x^{(1/3)}-2\,\ln\left(x^{(1/3)}-1\right)+$$
$$\ln\left(x^{(2/3)}+x^{(1/3)}+1\right)+2\,\ln\left(x^{(1/6)}-1\right)-$$
$$\ln\left(x^{(1/3)}+x^{(1/6)}+1\right)-2\,\ln\left(x^{(1/6)}+1\right)+$$
$$\ln\left(x^{(1/3)}-x^{(1/6)}+1\right)+6\,x^{(1/6)}$$

The algorithm used yields a complex expression; if you have in mind an antiderivative on positive numbers, this expression yields a discontinuous function with values that are not real. Let's try to convert this to a real expression for x positive. First, let's make x a positive variable:

```
> assume(x,positive);
```

Now let's get rid of that arcsin:

```
> convert(%,ln);
```

$$2\sqrt{x\sim} - \ln\left(\sqrt{x\sim}+1\right) + \ln\left(1-\sqrt{x\sim}\right) - \ln(x\sim -1) -$$
$$3\,x\sim^{(1/3)} - 2\ln\left(x\sim^{(1/3)}-1\right) + \ln\left(x\sim^{(2/3)} + x\sim^{(1/3)} + 1\right) +$$
$$2\ln\left(x\sim^{(1/6)}-1\right) - \ln\left(x\sim^{(1/3)} + x\sim^{(1/6)} + 1\right) -$$
$$2\ln\left(x\sim^{(1/6)}+1\right) + \ln\left(x\sim^{(1/3)} - x\sim^{(1/6)} + 1\right) + 6\,x\sim^{(1/6)}$$

A sum of logs cannot be combined in general; that depends on the arguments. But now, we don't care for branch cuts of ln; to the contrary, by neglecting them, we might change the expression into a continuous one. So, let's use the option `symbolic`:

```
> combine(%,ln,symbolic);
```

$$2\sqrt{x\sim} - 3\,x\sim^{(1/3)} + 6\,x\sim^{(1/6)} +$$
$$\ln\left(\left(1-\sqrt{x\sim}\right)\left(x\sim^{(2/3)} + x\sim^{(1/3)} + 1\right)\left(x\sim^{(1/6)}-1\right)^2\right.$$
$$\left(x\sim^{(1/3)} - x\sim^{(1/6)} + 1\right) \;/\; \left(\sqrt{x\sim}+1\right)(x\sim -1)$$
$$\left(x\sim^{(1/3)}-1\right)^2\left(x\sim^{(1/3)} + x\sim^{(1/6)} + 1\right)$$
$$\left.\left(x\sim^{(1/6)}+1\right)^2\right)$$

If we apply `radsimp` to this expression directly, nothing happens: `radsimp` cannot reach the argument of ln. Let's analyze the expression. You can use `op` to find that out, but we can see it directly. The expression is a sum of four components; the fourth component is a function call, its argument being its sole regular component. Now `radsimp` must act on this subcomponent, so it must reach an operand of an operand of `ttt`. Let's apply it to all the subcomponents of the components of the expression with the aid of a double form of `map`:

```
> map( u->map(radsimp,u) , % );
```

$$2\sqrt{x\sim} - 3\,x\sim^{(1/3)} + 6\,x\sim^{(1/6)} + \ln\left(-\frac{1}{\left(x\sim^{(1/6)}+1\right)^6}\right)$$

This contains only a constant imaginary component:

```
> evalc(%);
```

$$2\sqrt{x\sim} - 3\,x\sim^{(1/3)} + 6\,x\sim^{(1/6)} - 6\,\ln\left(x\sim^{(1/6)} + 1\right) + I\,\pi$$

```
> evalc(Re(%));
```

$$2\sqrt{x\sim} - 3\,x\sim^{(1/3)} + 6\,x\sim^{(1/6)} - 6\,\ln\left(x\sim^{(1/6)} + 1\right)$$

We have used the option `symbolic` so we cannot be sure that this is a correct antiderivative (apart from this being the same result as in section 4.9 on page 52). So let's check it:

```
> normal(diff(%,x)-integrand);
```

$$0$$

That confirms the correctness.
It's a good habit to delete an assumption as soon as it becomes superfluous:

```
> x:='x';
```

$$x := x$$

15.9 Manipulating trigonometric expressions

The procedure `simplify( ,trig)` applies the rules

a. $\sin^2(\phi) + cos^2(\phi) = 1$

b. $\tan(\phi) = \frac{\sin(\phi)}{\cos(\phi)}$

to get rid of `tan` and `sin` as much as possible.

```
> 5*sin(x)^2 + 2*cos(x)^2; simplify(%,trig );
```

$$5\,\sin(x)^2 + 2\,\cos(x)^2$$

$$-3\,\cos(x)^2 + 5$$

```
> sin(x)^3+cos(x)^3; simplify(%,trig );
```

$$\sin(x)^3 + \cos(x)^3$$

$$\cos(x)^3 + \sin(x) - \sin(x)\,\cos(x)^2$$

```
> sin(t)+tan(t); simplify(%,trig );
```

$$\sin(t) + \tan(t)$$

$$\frac{\sin(t)\,(\cos(t)+1)}{\cos(t)}$$

If no sin or cos is present in the expression, no conversion of tan is applied.

If you prefer to remove cos as much as possible, apply simplify with **side relations** to the relation $\sin^2(x) + cos^2(x) = 1$. See section 11.11 on page 147.

```
> simplify( %% , {sin(x)^2+cos(x)^2=1} , [cos(x)] );
```

$$\sin(t) + \tan(t)$$

The other manipulation procedures have the same inclination to cos. For instance, the procedure **expand**, which can expand arguments of sin, cos, and tan:

```
> cos(5*x+y); expand(%);
```

$$\cos(5\,x + y)$$

$$16\,\cos(y)\,\cos(x)^5 - 20\,\cos(y)\,\cos(x)^3 + 5\,\cos(y)\,\cos(x) - 16\,\sin(y)\,\sin(x)\,\cos(x)^4 + 12\,\sin(y)\,\sin(x)\,\cos(x)^2 - \sin(y)\,\sin(x)$$

The reverse direction is handled with combine(,trig):

```
> combine(%,trig);
```

$$\cos(5\,x + y)$$

Trigonometric products can also be handled with combine(,trig):

```
> sin(x)*sin(y); combine(%,trig);
```

$$\sin(y)\,\sin(x)$$

$$\frac{1}{2}\cos(-y + x) - \frac{1}{2}\cos(y + x)$$

The procedure combine(,trig) tries to convert powers and products of sin and cos function calls into sums of sin and cos function calls. It does not handle tan function calls.

The reverse direction is handled with its counterpart **expand**, which, however, may have unwanted other effects:

```
> expand(%);
```

$$\sin(y)\,\sin(x)$$

The trigonometric functions incorporate some automatic simplifications. For instance,

```
> cos(11*Pi/2-x);
```

$$-\sin(x)$$

$\sin(x + n\pi)$ can be simplified to $\sin(x)$ easily when the name n has the *property* of being an integer:

```
> n^2 + sin(a+b+2*n*Pi) + cos(y-2*n*Pi)^3;
```

$$n\sim^2 + \sin(a+b+2\,n\sim\ \pi) + \cos(y-2\,n\sim\ \pi)^3$$

```
> expand(%);
```

$$n\sim^2 + \sin(a)\,\cos(b) + \cos(a)\,\sin(b) + \cos(y)^3$$

(In releases before V.4, use side relations.)

Conversion of `sin` and `cos` to `tan` is available:

```
> convert( cos(x) , tan );
```

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

Conversion to `sincos` does not yield the original expression:

```
> convert( % , sincos );
```

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

But `simplify( ,trig)` can do the job:

```
> simplify( % , trig );
```

$$\cos(x)$$

In most cases the procedure `convert( ,tan)` bisects the arguments of sin and cos, but unnecessary bisections are avoided if possible:

```
> sin(alpha)/cos(alpha)+cos(beta);
```

$$\frac{\sin(\alpha)}{\cos(\alpha)} + \cos(\beta)$$

```
> convert(%,tan);
```

$$\tan(\alpha) + \frac{1-\tan(\frac{1}{2}\,\beta)^2}{1+\tan(\frac{1}{2}\,\beta)^2}$$

Maple knows that compositions of trigonometric functions with their inverses should be handled with care:

```
> arcsin(sin(20));
```

$$20 - 6\,\pi$$

```
> arcsin(sin(x)); simplify(%);
```

$$\arcsin(\sin(x))$$

$$\arcsin(\sin(x))$$

But arcsin is the right inverse of sin. Maple handles this by automatic simplification:

```
> sin(arcsin(x));
```

$$x$$

Conversions of trigonometric expressions to complex expressions with exp are available:

```
> convert(arctan(x),expln);
```

$$\frac{1}{2} I \left(\ln(1 - I x) - \ln(1 + I x)\right)$$

```
> convert(sin(x),expln);
```

$$-\frac{1}{2} I \left(\mathbf{e}^{(I x)} - \frac{1}{\mathbf{e}^{(I x)}}\right)$$

The reverse direction:

```
> evalc(%);
```

$$\frac{1}{2}\sin(x) + \frac{1}{2}\frac{\sin(x)}{\cos(x)^2 + \sin(x)^2} + I\left(-\frac{1}{2}\cos(x) + \frac{1}{2}\frac{\cos(x)}{\cos(x)^2 + \sin(x)^2}\right)$$

```
> simplify(%,trig);
```

$$\sin(x)$$

As an aid for substitutions, the procedure **trigsubs** can show possibilities:

```
> readlib(trigsubs);
```

$$\mathbf{proc}(s, f) \ \ldots \ \mathbf{end}$$

```
> trigsubs( cos(2*x) );
```

$$\left[\cos(2x),\ \cos(2x),\ 2\cos(x)^2 - 1,\ 1 - 2\sin(x)^2,\ \cos(x)^2 - \sin(x)^2,\ \frac{1}{\sec(2x)},\ \frac{1}{\sec(2x)},\ \frac{1 - \tan(x)^2}{1 + \tan(x)^2},\ \frac{1}{2}\mathbf{e}^{(2 I x)} + \frac{1}{2}\mathbf{e}^{(-2 I x)}\right]$$

The **hyperbolic functions** can be handled in about the same way as the trigonometric functions, using the same option `trig`, except that `trigsubs` cannot be used.

15.10 Manipulating parts of expressions

Often, a part of a large expression must be manipulated without changing other parts to prevent unwanted, possibly questionable conversions. There are several standard tricks for this purpose, demonstrated here in examples where it would be easier to do mental calculations and type in the results, but these exercises show how to deal with much larger expressions.

- First, you can use `subs` for replacing subexpressions with others, or `subsop` for replacing a component. See Chapter 11, *Substitution and subexpressions*:

```
> exp(x+y)+cos(x+y);
```

$$\mathbf{e}^{(y+x)} + \cos(y+x)$$

Suppose that we want to expand the second term only. We can realize this with:

```
> subsop(2=expand(op(2,%)),%);
```

$$\mathbf{e}^{(y+x)} + \cos(y)\,\cos(x) - \sin(y)\,\sin(x)$$

- If each of the components of an expression is to be processed but the main structure of the expression should be kept, you can use **map**. See section 10.3 on page 127.

```
> (cos(x+y)-1)^2; map(expand,%);
```

$$(\cos(y+x)-1)^2$$

$$(\cos(y)\,\cos(x) - \sin(y)\,\sin(x) - 1)^2$$

```
> (x+1)/(x^2-1) + x/(x^2+3*x); map(normal,%);
```

$$\frac{x+1}{x^2-1} + \frac{x}{x^2+3\,x}$$

$$\frac{1}{x-1} + \frac{1}{x+3}$$

- The procedure **collect** can describe an expression as a polynomial in a variable or a function call; moreover, it can process the coefficients at the same time. For instance,

```
> a^2*cos(x)+a^2*cos(x)^3+(a*cos(x))^2-9*cos(x)^3
>      -2*a*cos(x)^2-cos(x);
```

$$a^2\,\cos(x) + a^2\,\cos(x)^3 + a^2\,\cos(x)^2 - 9\,\cos(x)^3 - 2\,a\,\cos(x)^2 - \cos(x)$$

You can interpret this expression as a polynomial in `cos(x)` and factor the coefficients with:

```
> collect(%,cos(x),factor);
```

$$(a-3)\,(a+3)\,\cos(x)^3 + a\,(a-2)\,\cos(x)^2 + (a-1)\,(a+1)\,\cos(x)$$

- You can take out a part of an expression in an algebraic way, process the remaining expression, then complete the result. In the following example, we want to combine the first and the third term into one quotient:

```
> (x^2-x-a^2+a)/(x^2-x-x*a+a)+1/sqrt(x+a)-1;
```

$$\frac{x^2-x-a^2+a}{x^2-x-x\,a+a}+\frac{1}{\sqrt{x+a}}-1$$

```
> % - op(2,%);
```

$$\frac{x^2-x-a^2+a}{x^2-x-x\,a+a}-1$$

```
> normal(%)+op(2,%%);
```

$$\frac{a}{x-1}+\frac{1}{\sqrt{x+a}}$$

- For conversion of trigonometric and hyperbolic functions into sin and cos, it is possible to indicate that only function calls with special arguments must be converted. For instance:

```
> tan(x+1)+sec(y-1);
```

$$\tan(x+1)+\sec(y-1)$$

```
> convert(%,sincos,y);
```

$$\tan(x+1)+\frac{1}{\cos(y-1)}$$

- It is possible to shield subexpressions from the action of **expand** by entering these subexpressions as additional arguments. For instance,

```
> (x^a)^n + cos(t+s);
```

$$(x^a)^{n\sim}+\cos(t+s)$$

```
> expand( % ); expand( %% , (x^a)^n );
```

$$x^{(n\sim a)}+\cos(t)\,\cos(s)-\sin(t)\,\sin(s)$$

$$(x^a)^{n\sim}+\cos(t)\,\cos(s)-\sin(t)\,\sin(s)$$

- Another trick is offered by the procedure **freeze**, which can freeze a subexpression into a temporary name; after processing you can use **thaw**. First these procedures must be read in from the library by the command

```
> readlib(freeze);
```

proc(e) ... **end**

Here is an example:

```
> exp(a+b)+cos(x+y);
```

$$\mathbf{e}^{(a+b)}+\cos(y+x)$$

Let's suppose that $cos(x+y)$ should be expanded, while e^{a+b} should be left the same. Then we can use:

```
> subsop(1=freeze(op(1,%)),%);
```

$$freeze/R0 + \cos(y+x)$$

```
> expand(%);
```

$$freeze/R0 + \cos(y)\ \cos(x) - \sin(y)\ \sin(x)$$

```
> thaw(%);
```

$$\mathbf{e}^{(a+b)} + \cos(y)\ \cos(x) - \sin(y)\ \sin(x)$$

If you want to denote `freeze/R0` in a command, don't forget to use back quotes, otherwise Maple divides `freeze` by `R0`:

```
> `freeze/R0` , freeze/R0 ;
```

$$freeze/R0,\ \frac{freeze}{R0}$$

- In the last example, the procedure **expandoff** can also be used. This procedure must be read in from the library in a specific way:

```
> expand( expandoff() );
```

$$\text{expandoff}()$$

Now we can handle the previous problem by disabling `expand` for the function `exp`:

```
> exp(a+b)+cos(x+y);
```

$$\mathbf{e}^{(a+b)} + \cos(y+x)$$

```
> expandoff(exp);
> expand(%);
```

$$\mathbf{e}^{(a+b)} + \cos(y)\ \cos(x) - \sin(y)\ \sin(x)$$

We can reset the expand facility for `exp` with:

```
> expand( expandon() );
```

$$\text{expandon}()$$

```
> expandon(exp);
```

However, the procedure `expand` has built-in memory, its so-called **remember table**, from which it recalls old results. Therefore, after the previous action it refuses to expand $\ln(x^t)$ as it recalls the result $\ln(x^t)$ from the former part of the session, but it expands $\ln(y^s)$ without problems:

```
> expand(ln(x^t)+ln(y^s));
```

$$\ln(x^t) + \ln(y^s)$$

The remedy is clearing the remember table of `expand` with **forget**. See section D.2 on page 295.

```
> readlib(forget)(expand);
> expand(ln(x^t)+ln(y^s));
```

$$\ln(x^t) + \ln(y^s)$$

- A rather complicated, but powerful and flexible facility is **frontend**, which can apply a procedure on an expression according to specifications. For instance,

```
> (cos(x+1)-1)^2;
```

$$(\cos(x+1) - 1)^2$$

We can ask to see this as a polynomial in `cos(x+1)` and apply `expand` on it as follows:

```
> frontend( expand , [%] );
```

$$\cos(x+1)^2 - 2\cos(x+1) + 1$$

The function call `cos(x+1)` has been frozen by `frontend`, then `expand` is called and the result is thawed.

A more complicated example is the following: we want to substitute a for x in the expression $x^2 + x\,y + cos(x)$, except for the x in the second term $x\,y$. This can be done with the substitution trick mentioned earlier in this section, but in more complicated cases this trick is less suitable than using `frontend`:

```
> x^2+x*y+cos(x);
```

$$x^2 + x\,y + \cos(x)$$

We want to modify the command `subs(x=a,%)`; therefore, the first argument to `frontend` should be `subs` and the second should be `[x=a,%]`: the list of arguments to `subs`. We can tell `frontend` with an option that expressions should be subdivided into subexpressions by splitting sums and powers, but not products. This freezes the product $x\,y$. Moreover, we want the function call `cos(x)` *not* to be frozen. Both requirements can be given together by the third argument, the option `[{'+','^'},{cos(x)}]`. But the second argument to `frontend` also contains `x=a`; this is not to be frozen either. We can reach this by extending the third argument to `[{'+','^'},{x=a,cos(x)}]` or `[{'+','^','='},{cos(x)}]`:

```
> frontend(subs,[x=a,%],[{'+','^'},{x=a,cos(x)}]);
```

$$a^2 + x\,y + \cos(a)$$

The procedure `frontend` expects two or three arguments: first the procedure that should be applied in a modified way, then a list of its arguments. As an optional last argument you can give a list of two sets: first the set of type names of subexpressions *not* to be frozen, second a set of subexpressions *not* to be frozen. For types, see section A.1 on page 275.

15.11 An example: converting a complex expression into a real expression

Often, the result of a computation in Maple is a complex expression There may be strong indications that this represents a real number. Here is an example:

```
> t1 := solve(x^6+2*x^4-4*x^2-7)[1];
```

$$t1 := \frac{1}{6}\,\frac{\sqrt{6}\sqrt{\%1^{(1/3)}\left(\%1^{(2/3)} + 64 - 4\ \%1^{(1/3)}\right)}}{\%1^{(1/3)}}$$

$$\%1 := 404 + 12\,I\sqrt{687}$$

A test with `evalf` suggests that this is a real number:

```
> evalf(%);
```

$$1.391198165 - .7928049728\,10^{-10}\,I$$

We want to convert the previous expression into an explicitly real one.
First, let's simplify the subexpression

$$\left(404 + 12\,I\sqrt{687}\right)^{(1/3)}$$

We can extract the basis of this power and then simplify its third root as follows:

```
> s1 := %1;
```

$$s1 := 404 + 12\,I\sqrt{687}$$

```
> evalc(convert(s1^(1/3),polar));
```

$$512^{(1/3)}\cos\left(\frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)\right) + I\,512^{(1/3)}\sin\left(\frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)\right)$$

```
> s2:=simplify(%);
```

$$s2 := 8\cos\left(\frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)\right) + 8\,I\sin\left(\frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)\right)$$

We can substitute the result into the main expression. It is not efficient to issue `subs(s1=s2^3,t1)`, because it is not easy to simplify $(s2^3)^{1/3}$ and $(s2^3)^{-(1/3)}$. Therefore, we can issue:

```
> t2:=subs( s1^(1/3)=s2 , s1^(-1/3)=1/s2 ,
>     s1^(2/3)=s2^2, t1 );
```

$$\frac{1}{6}\sqrt{6}\left((8\cos(\%1)+8I\sin(\%1))\left((8\cos(\%1)+8I\sin(\%1))^2+\right.\right.$$
$$\left.\left.64-32\cos(\%1)-32I\sin(\%1)\right)\right)^{(1/2)}\ /$$
$$(8\cos(\%1)+8I\sin(\%1))$$

$$\%1 := \frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)$$

Now the problem is that a complex part of this expression is contained in the argument of a square root. Let's see what happens when we square the whole thing and simplify with option `trig`:

```
> simplify(%^2,trig);
```

$$\frac{8}{3}\cos\left(\frac{1}{3}\arctan\left(\frac{3}{101}\sqrt{687}\right)\right)-\frac{2}{3}$$

The result is a real expression. Because of the result of `evalf(t1)` it is clear that `t1` equals the square root of this expression.

15.12 Verifying identities

Manipulation of expressions can be used to verify identities. For instance, we can try to check an antiderivative by verifying that its derivative equals the original function. Often this can be done by applying `normal` to the difference. See section 4.11 on page 53. But in many cases more manipulation is necessary for verifying identities. Sometimes, this proves to be difficult. If the result is numeric, you can use `evalf` to get an indication about it. If it is not, you can ask Maple to execute a numerical test with **testeq**.

```
> testeq( sin(x)/cos(x)=tan(x) );
```

$$true$$

Although this procedure uses numerical tricks, it does not walk into a trap such as the following:

```
> epsilon:=10^(-100):
> testeq( sin(x)/cos(x) + epsilon = tan(x) );
```

$$false$$

However, in other cases it does not decide, for instance,

```
> testeq( sin(x+epsilon)^2 + cos(x)^2 = 1 );
```

$$FAIL$$

```
> testeq( sin(x+epsilon)^2 + cos(x+epsilon)^2 = 1 );
```

$$FAIL$$

Another procedure **verify** can be used with some more comfort, but it relies on `testeq`.

If the expression contains radicals, `testeq` cannot be trusted. This is shown in the following example, where a solution of a polynomial equation is tested:

```
> pol := x^3+4*x^2+1:
> solve( pol=0 , x ):
> subs( x=%[2] , pol ):
> testeq( %=0 );
```

$$false$$

Here symbolic manipulation does the job well:

```
> simplify(expand(%%));
```

$$0$$

Moreover, `testeq` should not be applied to **RootOf** expressions. It checks equality of degrees and then uses `evalf`:

```
> testeq(RootOf(_Z^3-1)=RootOf(_Z^3+_Z^2-2));
```

$$true$$

Apart from the cases where the expression contains RootOf or radical subexpressions, `testeq` is a fast and rather reliable tool.

15.13 Reliability

The results of the manipulations discussed in this chapter are reliable in release V.5 and release V.4 (apart from possible bugs: nothing is perfect) with the following exceptions:

- `simplify` and `combine` with the option **`symbolic`** use simplification rules without checking conditions, as asked by this option;
- **evalc** expects all variables to be real; if that is not the case, you might get incorrect results;
- intentionally, **radsimp** does not respect the standard branch cuts; results must be checked.

In fact, these exceptions are choices of the developers of Maple. In the old **release V.3** it is more difficult to guarantee reliability; in the following cases conditions are not checked:

- **expand** in combination with powers, possibly powers of nonpositive numbers with undetermined exponents; moreover, it may convert expressions containing ln in a way that may not be correct if an argument to ln is not a real number. Remedies: `expandoff(ln)` and/or `frontend`: see earlier in this chapter
- **combine(** **,power)** with noninteger powers of nonpositive numbers
- **simplify(** **,ln)** with logarithms of products and powers. Moreover, it may convert the composition of ln and $x \to \mathrm{e}^x$ into the identity function when the argument is not a real number and not even an element of $\mathrm{R} \times\ <-\pi, \pi]$
- **simplify(** **,arctrig)** with compositions of arctrigonometric functions and the corresponding trigonometric functions, sometimes simplifying these to the identity function without checking that the arguments are in the correct range.

15.14 General advice for manipulating

In manipulating expressions with Maple, do not apply the general tools blindly. Many inexperienced users tend to use `simplify` first, and if the result is not satisfactory, they try to manipulate this result with other procedures. Generally, that is not a good method. First, think what you would do by hand. Then you may want to use more specific tools such as `factor`, `evalc`, or `expand`. If necessary, analyze an expression with `op`, `nops`, and `whattype`, possibly with `numboccur`.

If you want to use **simplify**, remember that `simplify` without a specific option is a very general tool, sometimes yielding the desired result, but often making things worse or even hopeless by doing *too much*. Often, **normal** is a better general tool, especially for checking if something equals zero.

chapter 16

Solving equations and inequalities in general

In Chapter 14, Polynomial equations and factoring polynomials, *many aspects of solving equations and sets of equations with Maple are discussed, but restricted to polynomial and rational equations. The present chapter examines more general types of equations and also inequalities.*

This vast field is difficult and full of pitfalls for a symbolic calculator. The present chapter shows how the user can cooperate with the system and how common and basic mathematics can help considerably.

Most of the examples are basic ones, but the same methods and ideas can be used in more complicated cases.

16.1 General principles in using Maple for solving equations and inequalities

Maple's powerful procedure **solve** can often find solutions. For polynomial equations the results are reliable, but in general you cannot be sure, so you better check the solutions.

No one expects an automatic solver to find all solutions of every equation, so the user must decide if all solutions have been found, and, if solutions are missing, try to help Maple in finding those missing solutions.

Generally, there are three tasks left to the user when solving equations with Maple:

1. The results of `solve` must usually be checked by substituting them into the equations and applying simplifications, approximations, or the procedure `testeq`. See section 15.12 on page 212. Be careful when the equations contain parameters: Maple interprets them as abstract objects and does not bother about special values of these parameters. See section 1.3 on page 6.
2. Try to discern whether all solutions have been found or not. Again, be careful when your equations contain parameters and you want to substitute values: think about special cases.
3. If `solve` cannot find all the solutions on its own, you can think about methods to be used by hand, for instance looking for patterns in the equations, guessing solutions, possibly substituting variables for subexpressions in order to obtain equations that are easier for Maple to handle, such as polynomial ones. (See again for instance section 1.3 on page 6.) Then try to execute your plan, using Maple as a tool.

E. Kamerich, *A Guide to Maple*

Generally, the procedure `solve` tries to find all complex solutions.
There are no facilities to restrict the **domain**, for instance to positive numbers. But there is a procedure `isolve` for solving equations over the integers.

16.2 An example: a trigonometric equation

Here is an example:

```
> eq := 6*sin(x)^3 + 11*sin(x)^2 - 3*sin(x)=2;
```

$$eq := 6\sin(x)^3 + 11\sin(x)^2 - 3\sin(x) = 2$$

```
> solve(eq,x);
```

$$\frac{1}{6}\pi, -\arcsin(\frac{1}{3}), -\arcsin(2)$$

As you might guess, Maple has solved this equation in two steps:

```
> solve( eq , sin(x) );
```

$$\frac{1}{2}, \frac{-1}{3}, -2$$

```
> map(arcsin,[%]);
```

$$\left[\frac{1}{6}\pi, -\arcsin(\frac{1}{3}), -\arcsin(2)\right]$$

From the first result, you can find easily all real solutions: $\frac{1}{6}\pi + 2k\pi$, $\frac{5}{6}\pi + 2k\pi$, $-\arcsin(\frac{1}{3}) + 2k\pi$ and $\pi - \arcsin(\frac{1}{3}) + 2k\pi$, for any integer k. But so can Maple, if you ask for that:

```
> _EnvAllSolutions:=true;
```

$$\_EnvAllSolutions := true$$

```
> solve( eq );
```

$$\frac{1}{6}\pi + \frac{2}{3}\pi \_B1\sim + 2\pi \_Z1\sim, -\arcsin(\frac{1}{3}) + 2\arcsin(\frac{1}{3}) \_B1\sim + 2\pi \_Z1\sim + \pi \_B1\sim, -\arcsin(2) + 2\arcsin(2) \_B1\sim + 2\pi \_Z1\sim + \pi \_B1\sim$$

This rather cryptic answer is to be read as follows:

- $\_B1\sim$ is to be understood as a binary: 0 or 1
- $\_Z1\sim$ is to be understood as an integer

Unfortunately, these variables are not easily accessible. (For programmers: in fact, they are local variables of procedures called by `solve`.) For instance, the following substitution does not work:

```
> subs( _Z1=4, _B1=0, [%] );
```

$$\left[\frac{1}{6}\pi + \frac{2}{3}\pi\,\_B1\sim + 2\pi\,\_Z1\sim,\right.$$
$$-\arcsin(\frac{1}{3}) + 2\arcsin(\frac{1}{3})\,\_B1\sim + 2\pi\,\_Z1\sim + \pi\,\_B1\sim,$$
$$\left.-\arcsin(2) + 2\arcsin(2)\,\_B1\sim + 2\pi\,\_Z1\sim + \pi\,\_B1\sim\right]$$

Here you can see how to handle this. First, it is practical to name the solution:

```
> sinsol:=%%:
```

The indeterminates in an expression can always be found with **indets**:

```
> pars:=indets([%]);
```

$$\{\_B1\sim, \_Z1\sim, \_B1\sim, \_Z1\sim, \_B1\sim, \_Z1\sim\}$$

You can see what these parameters are meant to be:

```
> about(pars);
{_B1, _Z1, _B1, _Z1, _B1, _Z1}:
  is used in the following assumed objects
  [_B1] assumed OrProp(0,1)
  [_B1] assumed OrProp(0,1)
  [_Z1] assumed integer
  [_B1] assumed OrProp(0,1)
  [_Z1] assumed integer
  [_Z1] assumed integer
```

but be careful: don't look for the order in the result of `about`. This is not necessarily equal to the order of the argument.
Now we can substitute, for instance:

```
> subs(pars[1]=0,pars[2]=3,pars[3]=0,pars[4]=-2,
>      pars[5]=1,pars[6]=0,[sinsol]);
```

$$\left[\frac{37}{6}\pi, -\arcsin(\frac{1}{3}) - 4\pi, \arcsin(2) + \pi\right]$$

You can find all real solutions in a certain range, say between -2π and 2π, by looking at the solution and making suitable choices for these parameters. In more complicated cases, you might prefer to leave this work to Maple as in the following. We will also pretend not to see that $\arcsin(2)$ is not a real number.

First, we make a sequence of solutions, then we will select the ones that are real and lie in the $[-2\pi, 2\pi]$. It is still necessary to choose good ranges for the parameters, but that can be done rather roughly:

```
> seq(seq(seq(seq(seq(seq(
>   subs(pars[1]=k1,pars[2]=k2,pars[3]=k3,
>        pars[4]=k4,pars[5]=k5,pars[6]=k6,
>   [sinsol]),
```

```
> k1=0..1),k2=-2..2),k3=0..1),k4=-2..2),
> k5=0..1),k6=-2..2):
```

This is a sequence of lists. Let's convert it into a set:

```
> map(op,{%});
```

$$\left\{ \arcsin(\frac{1}{3}) + 3\,\pi,\ \arcsin(\frac{1}{3}) - \pi,\ \arcsin(\frac{1}{3}) + 5\,\pi,\right.$$
$$\arcsin(2) - 3\,\pi,\ \arcsin(\frac{1}{3}) - 3\,\pi,\ \arcsin(2) + \pi,$$
$$\arcsin(\frac{1}{3}) + \pi,\ \frac{1}{6}\,\pi, -\arcsin(\frac{1}{3}),\ -\arcsin(2) + 2\,\pi,$$
$$-\arcsin(2) + 4\,\pi,\ -\frac{23}{6}\,\pi,\ -\frac{19}{6}\,\pi,\ -\frac{11}{6}\,\pi,\ -\frac{7}{6}\,\pi,\ \frac{5}{6}\,\pi,$$
$$\frac{17}{6}\,\pi,\ \frac{25}{6}\,\pi,\ \frac{29}{6}\,\pi,\ -\arcsin(\frac{1}{3}) - 4\,\pi,\ \frac{13}{6}\,\pi,$$
$$-\arcsin(2) - 4\,\pi,\ \arcsin(2) + 3\,\pi,$$
$$-\arcsin(2) - 2\,\pi,\ -\arcsin(\frac{1}{3}) - 2\,\pi,\ \arcsin(2) - \pi,$$
$$-\arcsin(\frac{1}{3}) + 2\,\pi,\ -\arcsin(\frac{1}{3}) + 4\,\pi,\ -\arcsin(2),$$
$$\left.\arcsin(2) + 5\,\pi \right\}$$

To avoid equal elements, we have made a set of all values. This set contains some nonreal elements. Let's select the real elements.

```
> select(x->(x=evalc(Re(x))),%):
```

In this simple case, we can trust the condition `x=evalc`, because `evalc` can handle simple `arcsin` expressions perfectly. In more complicated cases, you could use:

```
> select(x->(fnormal(evalf(x))=evalc(Re(evalf(x)))),%):
```

Now that all elements are real, we can remove the elements that are greater than 2π or smaller than -2π:

```
> remove(x->(evalf(x)>evalf(2*Pi)),%):
> remove(x->(evalf(x)<evalf(-2*Pi)),%);
```

$$\left\{ \arcsin(\frac{1}{3}) - \pi,\ \arcsin(\frac{1}{3}) + \pi,\ \frac{1}{6}\,\pi,\ -\arcsin(\frac{1}{3}),\ -\frac{11}{6}\,\pi,\ -\frac{7}{6}\,\pi,\right.$$
$$\left.\frac{5}{6}\,\pi,\ -\arcsin(\frac{1}{3}) + 2\,\pi \right\}$$

At last, you might want a list of the numeric approximations in ascending order:

```
> sort(evalf([op(%)],3),numeric);
```

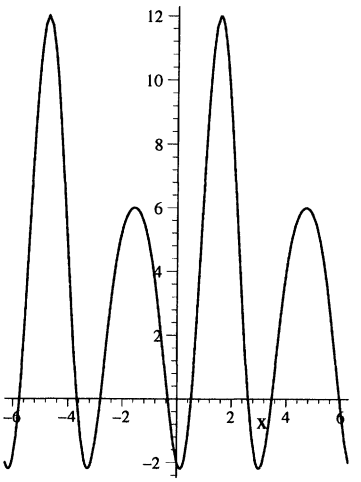
$$[-5.75,\ -3.67,\ -2.80,\ -.340,\ .524,\ 2.62,\ 3.48,\ 5.94]$$

Now let's test this result by a plot:

```
> lhs(eq)-rhs(eq);
```

$$6\,\sin(x)^3 + 11\,\sin(x)^2 - 3\,\sin(x) - 2$$

```
> plot(%,x=-2*Pi..2*Pi);
```



We can see that the zeros match the found solutions.

So we have checked in two ways that all solutions have been found:

a. by calculating the solutions in steps, where `solve` is applied only to a polynomial equation (in `sin(x)`);

b. by numeric approximation of the solutions and comparing them with a plot.

16.3 Another example: an exponential equation

Here is another example:

```
> eq := exp(3*x) = ((-9*q+27)*exp(2*x)-9*q*exp(x))/
>          (3*exp(2*x)+(3-q)*exp(x)-q-27);
```

$$eq := \mathbf{e}^{(3\,x)} = \frac{(-9\,q+27)\,\mathbf{e}^{(2\,x)} - 9\,q\,\mathbf{e}^{x}}{3\,\mathbf{e}^{(2\,x)} + (3-q)\,\mathbf{e}^{x} - q - 27}$$

```
> sol := solve(eq,x);
```

$$sol := \ln(-3),\ \ln(3),\ I\,\pi,\ \ln(\frac{1}{3}\,q)$$

We find a set of four solutions, which can be checked easily by substituting each of them into the equation and applying `simplify`.

```
> seq(simplify(subs(x=sol[k],lhs(eq)-rhs(eq))),k=1..4);
```

$$0,\ 0,\ 0,\ 0$$

Now the question must be asked if all solutions have been found. If you are only interested in real solutions, you could enter a value for q and try to check with a plot if all real solutions have been found, but that does not ensure that this is the case generally.
The other way is finding the solutions stepwise. That is easy here: you can do the same as what you would do by hand:

```
> expand(eq);
```

$$(\mathbf{e}^x)^3 = -9\,\frac{(\mathbf{e}^x)^2\,q}{\%1} + 27\,\frac{(\mathbf{e}^x)^2}{\%1} - 9\,\frac{q\,\mathbf{e}^x}{\%1}$$

$$\%1 := 3\,(\mathbf{e}^x)^2 + 3\,\mathbf{e}^x - q\,\mathbf{e}^x - q - 27$$

```
> solve(%,exp(x));
```

$$0,\ -3,\ 3,\ -1,\ \frac{1}{3}\,q$$

For no value of x the value of $\exp(x)$ can be zero. If we apply `ln` to the other solutions, we find the solutions rendered by `solve(eq,x)`. However, `solve` has not found all solutions, "forgetting" to add all integer multiples of $2\pi i$. The full solution is:

$$\ln(-3) + 2\,k\,\pi\,I,\ \ln(3) + 2\,k\,\pi\,I,\ I\,\pi + 2\,k\,\pi\,I,\ \ln(\frac{1}{3}\,q) + 2\,k\,\pi\,I$$

where k takes all integer values.

16.4 No solutions found

If you present a system of equations to Maple, you must provide all the variables for which it is to be solved. For instance:

```
> {x+y=3,x-y=5};
```

$$\{\,x + y = 3,\ x - y = 5\,\}$$

```
> solve( % , x );
```

No solution is found because there is no value for x that makes both equations identities in y. So no result is printed on the screen.

If Maple does not find a solution, it returns nothing at all and keeps silent. However, by putting

```
> infolevel[solve]:=1:
```

Maple tells if no solutions have been found. For instance:

```
> solve( cos(x) = ln(x^2) , x );
```

```
solve:   Warning: no solutions found
solve:   Warning: solutions may have been lost
```

The infolevel can be reset by

```
> infolevel[solve]:=0:
```

16.5 Inequalities and systems of inequalities

The procedure `solve` can handle some types of inequalities, for instance:

```
> solve(x^2<9,x);
```

$$\text{RealRange}(\text{Open}(-3), \text{Open}(3))$$

The solution above can be read as the open interval $< -3, 3 >$.

```
> solve(x^2>25,x);
```

$$\text{RealRange}(-\infty, \text{Open}(-5)) \,,\, \text{RealRange}(\text{Open}(5), \infty)$$

The solution of the last inequality can be read as a union of the open intervals: $< -\infty, -5 > \cup < 5, \infty >$.

```
> solve({x^2-x<1,x>-1,x<1},x);
```

$$\left\{ x < 1,\ \text{RootOf}\left(\_Z^2 - \_Z - 1, -.6180339887\right) < x \right\}$$

```
> convert(%,radical);
```

$$\left\{ x < 1,\ \frac{1}{2} - \frac{1}{2}\sqrt{5} < x \right\}$$

Here a different notation is used: $\frac{1}{2} - \frac{1}{2}\sqrt{5} < x$ AND $x < 1$.
The **RootOf** expression in the first result has a second argument, indicating which radical should be chosen:

```
> RootOf(_Z^2  - _Z - 1, -.6180339887);
```

$$\text{RootOf}\left(\_Z^2 - \_Z - 1, -.6180339887\right)$$

```
> convert(%,radical);
```

$$\frac{1}{2} - \frac{1}{2}\sqrt{5}$$

If this argument had been omitted, `convert(  ,radical)` would render the other radical:

```
> RootOf(_Z^2  - _Z - 1);
```

$$\text{RootOf}\left(\_Z^2 - \_Z - 1\right)$$

```
> convert(%,radical);
```

$$\frac{1}{2}+\frac{1}{2}\sqrt{5}$$

Don't try to use `allvalues` in this context, it makes no sense.

The following type of problem cannot be solved by `solve` (up to release V.5):

```
> solve({sin(x)<1/2,x>-3,x<3},x);
```

Here is another example:

```
> solve(1/x<1,x);
```

$$\text{RealRange}(-\infty, \text{Open}(0)),\ \text{RealRange}(\text{Open}(1), \infty)$$

Using parameters in inequalities is rather restricted.

Systems of linear inequalities and equalities can be handled as follows:

```
> solve({x+y<5,x-2*y>-7,y>-10},{x,y});
```

$$\{x < 15,\ -10 < y,\ x+y-5 < 0,\ -7-x+2y < 0,\ -27 < x,\ y < 4\}$$

```
> solve({x+y=5,x-2*y>-7,y>-10},{x,y});
```

$$\{-10 < y,\ x = -y+5,\ y < 4\}$$

For linear programming see the package `simplex`.

16.6 Manipulating equations and sets of equations

There are many possibilities for manipulating equations and systems of equations apart from the most obvious tool, substitution. The standard tools are demonstrated here. We apply these to the following two equations:

```
> x+1=y; eq1:=%:
```

$$x+1=y$$

```
> u+(x-1)^2=w; eq2:=%:
```

$$u+(x-1)^2=w$$

- Take the left-hand side and right-hand side with **lhs** and **rhs**:

```
> lhs(eq1); rhs(eq1);
```

$$x+1$$

$$y$$

- Add or subtract the same element to/from both sides:

```
> (x-1)^2;
```

$$(x-1)^2$$

```
> lhs(eq2)-%=rhs(eq2)-%;
```

$$u = w - (x-1)^2$$

The element added to both sides has been entered first, then used by quoting. This prevents typographical errors such as:

```
> lhs(eq2)-(x-1)^2=rhs(eq2)-(x-1^2);
```

$$u = w - x + 1$$

- Multiply both sides with the same factor:

```
> p*eq1;
```

$$p(x+1) = py$$

- Apply a function or other procedure to both sides with the aid of `map`:

```
> map(sqrt,eq1);
```

$$\sqrt{x+1} = \sqrt{y}$$

- Some procedures can be applied in a direct way:

```
> expand(eq2);
```

$$u + x^2 - 2x + 1 = w$$

- Add two equations:

```
> eq1 + eq2;
```

$$x + 1 + u + (x-1)^2 = y + w$$

- Divide (or multiply) two equations:

```
> lhs(eq1)/lhs(eq2)=rhs(eq1)/rhs(eq2);
```

$$\frac{x+1}{u+(x-1)^2} = \frac{y}{w}$$

- Combine two equations by simplification to **side relations**. See section 11.11 on page 147, section 14.11 on page 186, and section 15.9 on page 203. For instance, we can use `eq1` for simplification of `eq2`:

```
> simplify(eq2,{eq1});
```

$$u + 4 - 4y + y^2 = w$$

Here Maple has chosen to eliminate `x`. If we prefer elimination of `y` as far as possible, then we can add a third argument `[y]`:

```
> simplify(eq2,{eq1},[y]);
```

$$u + x^2 - 2x + 1 = w$$

This simplification uses Gröbner basis methods for polynomials. However, this method is not restricted to polynomials. Here is an example:

```
> exp(x)*exp(y) = exp(x)**2-t;
```

$$e^x e^y = (e^x)^2 - t$$

```
> exp(x)-1=u*exp(x);
```

$$e^x - 1 = u e^x$$

```
> simplify(%%,{%},[exp(x)]);
```

$$-\frac{e^y}{-1+u} = -\frac{t-2ut+u^2t-1}{1-2u+u^2}$$

- Sometimes, the procedure **isolate** can be used, which must be read from the library when it is used for the first time in a session:

```
> (a-b*cos(x))/(c-d*cos(x))=t-5;
```

$$\frac{a-b\cos(x)}{c-d\cos(x)} = t-5$$

```
> readlib(isolate)( % ,cos(x) );
```

$$\cos(x) = -\frac{a-tc+5c}{-b+td-5d}$$

Be careful with this procedure: if there is more than one possibility, `isolate` makes a choice from among them.

```
> 6*cos(x)^2 - 5*cos(x) - 1 = 0;
```

$$6\cos(x)^2 - 5\cos(x) - 1 = 0$$

```
> isolate(%,cos(x));
```

$$\cos(x) = \frac{-1}{6}$$

- You might want to assign the solutions to the variables. This can be done easily with **assign**:

```
> solve({X+2*Y=a,3*X-Y=b},{X,Y});
```

$$\left\{Y = \frac{3}{7}a - \frac{1}{7}b,\ X = \frac{1}{7}a + \frac{2}{7}b\right\}$$

```
> assign(%);
> X,Y;
```

$$\frac{1}{7}a + \frac{2}{7}b,\ \frac{3}{7}a - \frac{1}{7}b$$

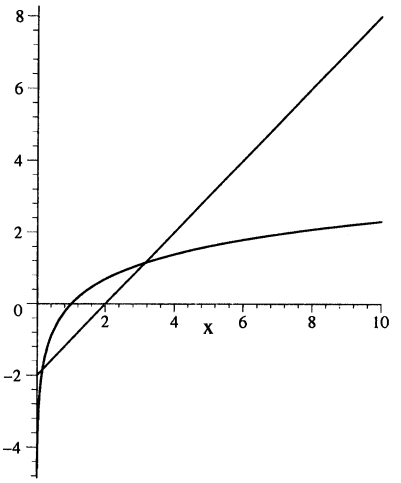
16.7 Solving equations numerically

For solving equations and sets of equations in a numerical way, the procedure **fsolve** can be used. When solving a polynomial equation, `fsolve` tries to find approximations to all real solutions (or, with option `complex`, all complex solutions) and generally succeeds. For more general equations and sets of equations `fsolve` is content with just *one* solution. For instance:

```
> eq := ln(x)=x-2;
```

$$eq := \ln(x) = x - 2$$

```
> plot( {lhs(%),rhs(%)} , x=0.01 .. 10 );
```




```
> fsolve(eq,x);
```

.1585943396

From the graph of ln it is clear that there should be a solution between 0 and 1. We can ask for this solution with the aid of an option `x=0..1`:

```
> fsolve( eq , x , x = 0 .. 1 );
```

.1585943396

There is no solution greater than 4, but we can ask for such a solution:

```
> fsolve( eq , x, x = 4 .. infinity );
```

$$\text{fsolve}(\ln(x) = x - 2, x, x = 4 \ldots \infty)$$

If `fsolve` cannot find a solution of a nonpolynomial equation, it may keep silent or return the command unevaluated.

It is possible that a root exists, although `fsolve` cannot find one, but that is a very rare exception.

16.8 Solving systems of equations numerically

Systems of equations can be given to `fsolve` in the same way as to `solve`. Here is an example that can be solved in an exact way as well.

```
> eq1 := cos(x)*sin(y)^2 + 6*sin(y) = 4*cos(x);
```

$$eq1 := \cos(x)\,\sin(y)^2 + 6\,\sin(y) = 4\,\cos(x)$$

```
> eq2 := 2*x+3*y=Pi;
```

$$eq2 := 2\,x + 3\,y = \pi$$

```
> fsolve( {eq1,eq2} , {x,y} );
```

$$\{y = -8.936914292,\ x = 14.97616777\}$$

Another solution `{y=0 , x=π/2}` can be seen directly. We can try to find this with `fsolve` by specifying ranges as a third option:

```
> fsolve( {eq1,eq2} , {x,y} , {x=1 .. 3, y=-1 .. 1} );
```

$$\{x = 1.571798612,\ y = -.0006681900122\}$$

This is the desired solution, but it is not very accurate. Trying to improve the accuracy by enlarging `Digits` is not always successful:

```
> Digits:=20:
> fsolve({eq1,eq2}, {x,y}, {x=1.5 .. 1.6, y=-.1 .. 0.1});
```

$$\{x = 1.5707967777287823174,\ y = -.30062259046544063546\,10^{-6}\}$$

We get no better result. Let's try to improve the performance by guiding Maple step by step, taking advantage of the fact that the second equation is easy.

```
> solve(eq2,x);
```

$$-\frac{3}{2}\,y + \frac{1}{2}\,\pi$$

There would be no reason to use `fsolve` in this first step; it would even be impossible, as `fsolve` cannot cope with an unspecified parameter `y`.

```
> subs(x=%,eq1);
```

$$\cos(-\frac{3}{2}\,y + \frac{1}{2}\,\pi)\,\sin(y)^2 + 6\,\sin(y) = 4\,\cos(-\frac{3}{2}\,y + \frac{1}{2}\,\pi)$$

```
> fsolve( % , y , y=-1..1 );
```

$$0$$

More details can be found in the on-line help for `fsolve`.

16.9 Series of an implicitly defined function

Suppose that we have a parameterized equation in x: $f_t(x) = 0$ and a value t_0 such that one or more solutions of the equation $f_{t_0}(x) = 0$ are available: say $f_{t_0}(x_0) = 0$. Then you might be interested in a parameterized extension $\mathrm{x}(t)$ of this solution. From an implicit function theorem we know that, if $f'_{t_0} \neq 0$ and $f_t(x)$ depends on x and t in a continuous way, then there is a parameterized solution $x(t)$ of $f_t(x) = 0$ for t in a neighborhood of t_0 such that x_{t_0}=x_0.

If the parameter t does not have a special value, it may be impossible to solve explicitly the equation for x, but you might be interested in a series approximation to $t \to x(t)$ in t_0. For instance:

```
> eq1:=exp(t*x)=x*exp(x)+exp(t);
```

$$eq1 := \mathrm{e}^{(t\,x)} = x\,\mathrm{e}^{x} + \mathrm{e}^{t}$$

It is easy to see that $\{x = 0, t = 0\}$ satisfies this equation. We can calculate a series expansion in $t = 0$ of a function $t \to x(t)$ on a neighborhood of 0, satisfying $\mathrm{e}^{(tx)} = x\,\mathrm{e}^{x} + \mathrm{e}^{t}$, as follows:

First we must convert the equation into a series:

```
> lhs(eq1)-rhs(eq1);
```

$$\mathrm{e}^{(t\,x)} - x\,\mathrm{e}^{x} - \mathrm{e}^{t}$$

```
> series( % , x=0 , 5 );
```

$$\left(1-\mathrm{e}^{t}\right) + (t-1)\,x + \left(\frac{1}{2}t^2 - 1\right)x^2 + \left(-\frac{1}{2} + \frac{1}{6}t^3\right)x^3 + \left(-\frac{1}{6} + \frac{1}{24}t^4\right)x^4 + O(x^5)$$

The procedure `solve` perceives this series as an equation (appending "=0") and it tries to find a series as a solution.

```
> solve(%,x);
```

$$-1t - \frac{5}{2}t^2 - \frac{43}{6}t^3 - \frac{569}{24}t^4 + O(t^5)$$

Let's check this result by substituting:

```
> convert(%,polynom);
```

$$-t - \frac{5}{2}t^2 - \frac{43}{6}t^3 - \frac{569}{24}t^4$$

```
> subs(x=%,eq1);
```

$$\mathrm{e}^{((-1/24\,t^2(24+60\,t+172\,t^2+569\,t^3)))} = \left(-t-\frac{5}{2}t^2-\frac{43}{6}t^3-\frac{569}{24}t^4\right)\mathrm{e}^{\left(-t-\frac{5}{2}t^2-\frac{43}{6}t^3-\frac{569}{24}t^4\right)}+ \mathrm{e}^t$$

This is not an exact identity, but the difference of the left-hand side and the right-hand one should be of order 5: the order of the series equation that has been solved. To show this more clearly we can ask for a series expansion of order 6.

```
> series( lhs(%)-rhs(%) , t=0 , 6 );
```

$$-\frac{2579}{30}t^5+O(t^6)$$

The result is of order 5, the same as the order of the original series. Apart from the order of the last series expansion there are two limiters to the order of the result:

1. the order of the series to which `solve` is applied
2. the value of the variable `Order`. See section 8.2 on page 109.

In order to find a solution series of order n, both limiters should be at least n. Here is a pitfall: if only the variable `Order` is raised, the order of the series expansion of the equation being high enough, the procedure `solve` yields no better result, because it only picks up the result from the remember table of one of its subprocedures, when it gets the old arguments. Therefore, in any case, raise both together before you ask Maple to solve again. For remember tables, see Appendix D, *Procedures remembering previous results.*

From the series expansion of $t \to \mathrm{x}(t)$, its *derivative* in 0 can be read as the coefficient of t, in this case -1. This can also be calculated with `diffimplicit`.

In the previous calculation we have used series expansion of the equation to `x`, but we can use series expansion to `t` as well:

```
> series(lhs(eq1)-rhs(eq1),t=0,5);
```

$$-x\,\mathrm{e}^x+(x-1)\,t+\left(-\frac{1}{2}+\frac{1}{2}x^2\right)t^2+\left(\frac{1}{6}x^3-\frac{1}{6}\right)t^3+\left(\frac{1}{24}x^4-\frac{1}{24}\right)t^4+O(t^5)$$

```
> solve(%,x);
```

$$-t-\frac{5}{2}t^2-\frac{43}{6}t^3-\frac{569}{24}t^4+O(t^5)$$

The result is the same.

Obviously, this method fails if no zeroth-order solution can be found. For instance, Maple cannot solve for x from the equation in the case $t = 2$:

```
> subs(t=2,eq1);
```

$$\mathbf{e}^{(2\,x)} = x\,\mathbf{e}^{x} + \mathbf{e}^{2}$$

```
> solve(%,x);
```

$$\mathrm{RootOf}\left(\_Z\,\mathbf{e}^{-Z} - \left(\mathbf{e}^{-Z}\right)^{2} + \mathbf{e}^{2}\right)$$

The problem is translated into a **RootOf** expression; Maple finds no solutions. This is why the following `solve` command has no result:

```
> series( lhs(eq1)-rhs(eq1) , t=2 , 5 );
```

$$\left(\mathbf{e}^{(2\,x)} - x\,\mathbf{e}^{x} - \mathbf{e}^{2}\right) + \left(\mathbf{e}^{(2\,x)}\,x - \mathbf{e}^{2}\right)(t-2) + \left(-\frac{1}{2}\,\mathbf{e}^{2} + \frac{1}{2}\,\mathbf{e}^{(2\,x)}\,x^{2}\right)(t-2)^{2} + \left(\frac{1}{6}\,\mathbf{e}^{(2\,x)}\,x^{3} - \frac{1}{6}\,\mathbf{e}^{2}\right)(t-2)^{3} + \left(\frac{1}{24}\,\mathbf{e}^{(2\,x)}\,x^{4} - \frac{1}{24}\,\mathbf{e}^{2}\right)(t-2)^{4} + O((t-2)^{5})$$

```
> solve( % , x );
```

Here is another example, this time with more than one solution:

```
> eq2 := x^2*exp(t)=2*t*x+1;
```

$$eq2 := x^{2}\,\mathbf{e}^{t} = 2\,t\,x + 1$$

If we want a series solution for x in a neighborhood of $t = 0$, Maple must solve the following equation:

```
> subs(t=0,eq2);
```

$$x^{2}\,1 = 1$$

```
> solve(%,x);
```

$$1,\ -1$$

So we may expect two series solutions.

```
> series(lhs(eq2)-rhs(eq2),t=0,5);
```

$$\left(x^{2}-1\right) + \left(x^{2}-2\,x\right)t + \frac{1}{2}\,x^{2}t^{2} + \frac{1}{6}\,x^{2}t^{3} + \frac{1}{24}\,x^{2}t^{4} + O(t^{5})$$

```
> solve(%,x);
```

$$1 + \frac{1}{2}t - \frac{3}{8}t^{2} - \frac{13}{48}t^{3} + \frac{35}{128}t^{4} + O(t^{5}),\ -1 + \frac{3}{2}t - \frac{13}{8}t^{2} + \frac{61}{48}t^{3} - \frac{233}{384}t^{4} + O(t^{5})$$

16.10 Recurrence relations

There is a procedure available for solving recurrence relations: **rsolve**. For instance, we can calculate formulas for the generalized Fibonacci sequence $a, b, a+b, a+2b, 2a+3b, \ldots$ as follows. Here is the defining equation:

```
> fibo := F(n)=F(n-1)+F(n-2);
```

$$fibo := \mathrm{F}(n) = \mathrm{F}(n-1) + \mathrm{F}(n-2)$$

and the initial values:

```
> fibinit :=  F(1)=a , F(2)=b;
```

$$fibinit := \mathrm{F}(1) = a,\ \mathrm{F}(2) = b$$

Now we can apply the procedure `rsolve` to obtain an explicit formula for the nth element of the sequence:

```
> rsolve({ fibo , fibinit } , F(n) );
```

$$\frac{1}{5}\frac{\left(2\sqrt{5}-5\right)\left(2a+b+\sqrt{5}\,b\right)\left(-2\,\frac{1}{1-\sqrt{5}}\right)^{n}}{1-\sqrt{5}} - \frac{1}{5}\frac{\left(5+2\sqrt{5}\right)\left(2a+b-\sqrt{5}\,b\right)\left(-2\,\frac{1}{1+\sqrt{5}}\right)^{n}}{1+\sqrt{5}}$$

Let's test the formula by calculating the fourth element:

```
> normal( subs(n=4,%) , expanded );
```

$$a+2b$$

The same procedure can also try to calculate the *generating function* of the solution of a recurrence relation. The necessary option `genfunc(t)` must be entered between forward quotes to avoid clashing with a package of the same name.

```
> rsolve({ fibo , fibinit }, F(n) , 'genfunc'(t) );
```

$$\frac{-at-bt^2+at^2}{-1+t+t^2}$$

As an example, let's calculate from this result the first seven elements of the sequence. These can be found as the coefficients in the series expansion of this result up to order 8:

```
> series( % , t , 8 );
```

$$at + bt^2 + (a+b)\,t^3 + (a+2b)\,t^4 + (2a+3b)\,t^5 + (5b+3a)\,t^6 + (5a+8b)\,t^7 + O(t^8)$$

More tools for linear recurrence relations are available in the package `LREtools`.

16.11 Solving identities, matching patterns

There are two procedures for solving identities and pattern matching in Maple: `solve` applied to an identity, and **match**.

First a simple problem: for which a and b is $x + c = \frac{x^2+a}{x+b}$?

```
> eq:=x+c=(x^2+a)/(x+b);
```

$$eq := x + c = \frac{x^2 + a}{x + b}$$

```
> solve(identity(eq, x),{a,b});
```

$$\{a = -c^2, b = -c\}$$

Let's solve the same problem with `match`:

```
> match(eq,x,'sol');
```

$$true$$

The result *true* affirms that Maple can match both sides of the equation by substituting suitable values for the variables in the right-hand side. The substitution chosen by Maple has been assigned to the third argument:

```
> sol;
```

$$\{a = -c^2, b = -c\}$$

The procedure `match` tries to find values for the parameters of the right-hand expression in order to make this identical to the left side. Therefore, we get no result if the left-hand side and the right one are exchanged in the present example:

```
> rhs(eq)=lhs(eq);
```

$$\frac{x^2 + a}{x + b} = x + c$$

```
> match( %, x , 'sol' );
```

$$false$$

Here `match` tries to find a value for `c` such that the left-hand side equals the right-hand side; this is impossible.

Apart from this question, there are cases where `match` is successful and `solve (identity( ), )` is not, and other cases where the reverse is true. If you don't succeed with one of them, try the other.

16.12 Other procedures for solving

Differential equations must be solved with the procedure `dsolve`, which is dealt with in Chapter 17, *Solving differential equations*.

Systems of linear equations can also be solved with the aid of linear algebra. See section 18.10 on page 257.

A procedure `isolve` is available for **solving equations in integer variables**, and a procedure `msolve` is available for **solving equations in variables over Z mod m**.

chapter 17

Solving differential equations

This chapter introduces the main tools for handling differential equations with exact or approximate methods, and for graphics in this field. This field is developing at a great pace; Maple now offers a lot more tools than the basic ones, demonstrated in this chapter.

17.1 Ordinary differential equations (ODEs): denoting, solving, checking solutions

Let's start with one of the most basic differential equations: $y'' = -y$. Both sides of the equation are functions. In Maple notation, both must be applied to an argument, say x:

```
> diff( y(x) , x , x ) = -y(x) ;
```

$$\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x) = -\,\mathrm{y}(x)$$

If y is entered without argument x, the output indicates clearly that something is wrong:

```
> diff( y , x , x ) = - y;
```

$$0 = -y$$

A recurring mistake is forgetting the argument only for the undifferentiated function, like this:

```
> diff( y(x) , x , x ) = -y ;
```

$$\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x) = -y$$

This is *not correct*: the left-hand side is the second derivative at x, while the right-hand side is a function, not the value of this function at x. Maple does not accept it:

```
> dsolve(%);
Error, (in ODEtools/info) y(x) and y cannot both appear in the
given ODE.
```

For checking purposes, let's assign the given differential equation to a name.

```
> deq1 := %%%;
```

$$deq1 := \frac{\partial^2}{\partial x^2}\,\mathrm{y}(x) = -\,\mathrm{y}(x)$$

Maple can solve this ODE with **dsolve**:

```
> dsolve(%);
```

$$\mathrm{y}(x) = \_C1\,\sin(x) + \_C2\,\cos(x)$$

In this case, `dsolve` can determine what to find, but generally we must tell that by a second argument:

```
> dsolve( deq1 , y(x) );
```

$$\mathrm{y}(x) = \_C1\,\sin(x) + \_C2\,\cos(x)$$

In fact, Maple tries to find a function y, but expresses the solution as $y(x)$. That is the reason that the second argument should not be y, but $y(x)$.

The result found by `dsolve` contains the expected integration constants: $\_C1$ and $\_C2$. Names starting with an underscore are used by Maple for special purposes and should never be assigned values by the user.

It is clear that this solution is correct and that all solutions have been found, but generally you can't be sure, and then you must check the found solutions and you must decide by mathematical arguments if a full set of solutions has been found. Solutions can be checked with Maple. In release V.5, you can use **odetest**:

```
> odetest(%,deq1);
```

$$0$$

The result 0 proves that the solution is correct. (Sometimes some extra manipulation is necessary to find 0; see also section 15.12 on page 212.)
In earlier releases, don't use **assign** in this case, as you could for solutions of systems of equations; if you want to assign the corresponding function to y, issue `y:=unapply(rhs(%),x);` (see section 6.7 on page 77). We will not assign the solution, only substitute it:

```
> subs( %% , deq1 );
```

$$\frac{\partial^2}{\partial x^2}\,\_C1\,\sin(x) + \_C2\,\cos(x) = -\,\_C1\,\sin(x) - \_C2\,\cos(x)$$

```
> normal( lhs(%)-rhs(%) );
```

$$0$$

Sometimes more manipulation is necessary, but the above demonstrates the standard first try.

17.2 Ordinary differential equations with initial conditions

Let's solve the same ODE with initial conditions. Suppose that $y'(0) = \frac{1}{2}$. This is expressed as `D(y)(0)=1/2`: take the derivative *function* of y and apply this to `1/2`. Moreover, let's suppose $y(0) = 1$.

```
> {diff(y(x),x,x) = - y(x) , D(y)(0)=1/2, y(0)=1};
```

$$\left\{ \mathrm{D}(y)(0) = \frac{1}{2}, \mathrm{y}(0) = 1, \frac{\partial^2}{\partial x^2}\,\mathrm{y}(x) = -\,\mathrm{y}(x) \right\}$$

An ODE with initial conditions is entered as a set of equations. For checking purposes let's assign this set to a name before solving it.

```
> deq2 := % :
> dsolve( % , {y(x)} );
```

$$\mathrm{y}(x) = \frac{1}{2}\sin(x) + \cos(x)$$

Checking this solution is a little more complicated as the operator `D` expects a function. So we must make a function of the result of `dsolve` before substituting. See section 6.7 on page 77.

```
> unapply( rhs(%%) , x );
```

$$x \to \frac{1}{2}\sin(x) + \cos(x)$$

```
> subs( y=% , deq2 );
```

$$\left\{ \mathrm{D}\left(x \to \frac{1}{2}\sin(x) + \cos(x)\right)(0) = \frac{1}{2}, \left(x \to \frac{1}{2}\sin(x) + \cos(x)\right)(0) = 1, \frac{\partial^2}{\partial x^2}\left(x \to \frac{1}{2}\sin(x) + \cos(x)\right)(x) = -\left(x \to \frac{1}{2}\sin(x) + \cos(x)\right)(x) \right\}$$

We can check the equations in the system with the same method as previously, with the aid of `map`:

```
> map( eq -> normal( lhs(eq) - rhs(eq) ) , % );
```

$$\{0\}$$

You may dislike the use of `D` and `diff` in combination. Alternatively, the previous system of equations can be denoted by

```
> { D(D(y))(x) = - y(x) , y(0)=1 , D(y)(0)=1/2 };
```

$$\left\{ \mathrm{D}(y)(0) = \frac{1}{2}, \mathrm{y}(0) = 1, \left(D^{(2)}\right)(y)(x) = -\,\mathrm{y}(x) \right\}$$

which makes a nicer input line but a less readable output. For detailed information about acceptable forms for initial conditions, see the on-line help for `dsolve,ICs`.

17.3 Implicit solutions and checking them

Here is another example, the differential equation:

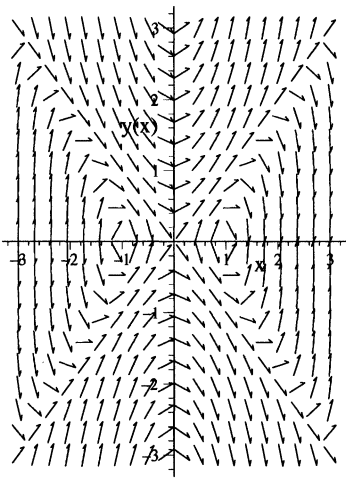
$$yy' = xy + x - x$$

```
> de3 := y(x) * diff(y(x),x) = x*y(x)^2 + x - x^3;
```

$$de3 := \mathrm{y}(x)\left(\frac{\partial}{\partial x}\,\mathrm{y}(x)\right) = x\,\mathrm{y}(x)^2 + x - x^3$$

In order to have an idea of the possible solutions let's plot the direction field of this differential equation:

```
> DEtools[dfieldplot](de3, y(x), x=-3..3, y(x)=-3..3);
```



Now let's solve it:

```
> dsolve( de3 , y(x) );
```

$$\mathrm{y}(x) = \sqrt{x^2 + \mathbf{e}^{(x^2)}\_C1}, \mathrm{y}(x) = -\sqrt{x^2 + \mathbf{e}^{(x^2)}\_C1}$$

We find two solutions; obviously, `dsolve` has found an implicit solution and then solved this equation to $y(x)$. If you prefer the **implicit solution**, you can try the option `'implicit'`:

```
> dsolve( de3 , y(x), 'implicit');
```

$$\mathrm{y}(x)^2 - x^2 - \mathbf{e}^{(x^2)}\_C1 = 0$$

If you want a **parametric solution**, you can try `dsolve( , ,'implicit', 'parametric')`. (In the present case, that is not successful.)

Sometimes, `dsolve` itself chooses to render an implicit solution. In such a case, you can ask for an **explicit solution**, if possible, with the option `'explicit'`.

In the present example, it is not difficult to see that all local solutions of the differential equations are represented in the result, if it is correct. Let's check it. In release V.5 you can check the results again with `odetest`:

```
> odetest( % , de3 );
```

$$0$$

For earlier releases, you can check implicit solutions by substituting and using **side relations** (see section 11.11 on page 147) as follows. In the set of relations we gather the solution and the derivative of the solution, because the ODE contains $\mathrm{y}(x)$ and $\frac{\partial}{\partial x}\,\mathrm{y}(x)$

```
> rels := { %% , diff(%%,x) };
```

$$rels := \left\{ 2\,\mathrm{y}(x)\left(\frac{\partial}{\partial x}\,\mathrm{y}(x)\right) - 2x - 2x\,\mathbf{e}^{(x^2)}\_C1 = 0,\ \mathrm{y}(x)^2 - x^2 - \mathbf{e}^{(x^2)}\_C1 = 0 \right\}$$

Then we can ask Maple to eliminate $y(x)$ and $\frac{\partial}{\partial x}y(x)$ as far as possible by applying simplification to the difference between the left-hand side and right-hand side.

```
> simplify( lhs(de3)-rhs(de3), rels ,
>     { y(x) , diff(y(x),x) } );
```

$$0$$

For a second-order ODE, the second derivative of the solution must also be put into the second argument set and `diff(y(x),x,x)` into the third argument set. In order to check a possible solution (`dsol`) of a 10th-order ODE (`deq`) in `y(x)`, we can try:

```
> simplify( lhs(deq)-rhs(deq) ,
>     { dsol(x) , seq(diff(dsol(x),x$k),k=1..10) },
>     { y(x) , seq(diff(y(x),x$k),k=1..10) } );
```

This method of checking works in many cases, but sometimes additional manipulations may be necessary.

17.4 DESol expressions appearing in solutions

Sometimes, when dsolve cannot find a solution, it presents an intermediate result with the aid of a DESol expression. For example:

```
> diff(y(x),x,x,x) + x*diff(y(x),x) + (x-1)*y(x) = 0;
```

$$\left(\frac{\partial^3}{\partial x^3}\,\mathrm{y}(x)\right) + x\left(\frac{\partial}{\partial x}\,\mathrm{y}(x)\right) + (x-1)\,\mathrm{y}(x) = 0$$

```
> dsolve( % , y(x) );
```

$$\mathrm{y}(x) = \_C1\left(6\,\mathrm{e}^{(-x)} + 6\,\mathrm{e}^{(-x)}\,x + \mathrm{e}^{(-x)}\,x^2\right) +$$
$$\left(6\,\mathrm{e}^{(-x)} + 6\,\mathrm{e}^{(-x)}\,x + \mathrm{e}^{(-x)}\,x^2\right) \int \mathrm{DESol}(\{\,(6+6\,x+x^2)$$
$$\left(\frac{\partial^2}{\partial x^2}\,\_\mathrm{Y}(x)\right) + (-12\,x - 3\,x^2)\left(\frac{\partial}{\partial x}\,\_\mathrm{Y}(x)\right) +$$
$$(12\,x + 9\,x^2 + x^3 - 12)\,\_\mathrm{Y}(x)\,\},\{\,\_\mathrm{Y}(x)\,\})\;dx$$

The DESol expression in the last result is a symbolic representation of the solutions of the differential equation described by the arguments to DESol. In this case, the differential equation is:

$$(6 + 6\,x + x^2)\left(\tfrac{\partial^2}{\partial x^2}\,\_\mathrm{Y}(x)\right) + (-12\,x - 3\,x^2)\left(\tfrac{\partial}{\partial x}\,\_\mathrm{Y}(x)\right)$$
$$+ (12\,x + 9\,x^2 + x^3 - 12)\,\_\mathrm{Y}(x) = 0$$

So the solution found with dsolve has the structure

$$\_C1\,f(x) + g(x)\int s(x)dx$$

where $s(x)$ is a symbolic representation of a solution of the last ODE in $\_Y$.

The DESol construction is discussed in section 17.9 on page 244.

In some cases the differential equation cannot be solved by dsolve, but it can be reduced with the aid of transformations. In such a case, an intermediate result may be presented containing the word **&where**. If you want to use or manipulate this, consult the on-line help for ODESolStruct.

17.5 Numerical approximations to solutions

The field of solving differential equations in an exact symbolic way is developing rapidly. With each new release of Maple, new algorithms for solving differential equations have been added. But there are limits; often, numerical approximations or series developments must be used.

Consider the differential equation: $u'(t) = \sin(t * u(t))$. As can be expected, `dsolve` cannot solve this in an exact way. The present section applies a numerical method to this ODE; the next section applies a series method to the same equation.

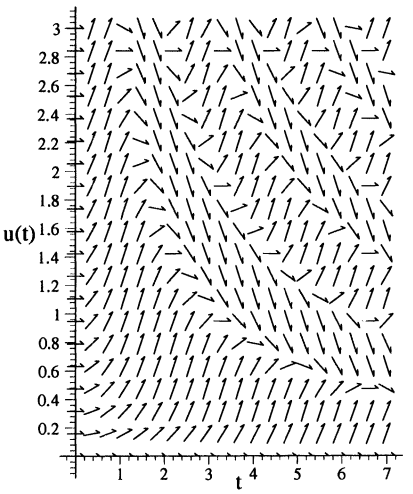
```
> diff(u(t),t) = sin(t*u(t));
```

$$\frac{\partial}{\partial t}\,\mathrm{u}(t) = \sin(t\ \mathrm{u}(t))$$

```
> deq4 := %:
```

Before solving, let's plot the direction field of this ODE with the procedure `DEtools[dfieldplot]`:

```
> DEtools[dfieldplot](deq4, u(t), t=0..7, u(t)=0..3);
```



Let's solve the equation in a numerical way with the command:

```
> Y := dsolve({deq4,u(0)=2},u(t),numeric);
```

$$Y := \mathbf{proc}(rkf45\_x)\ \ldots\ \mathbf{end}$$

This strange result says that a function has been assigned to `Y` that can calculate numerical approximations to function values of the solution. For instance:

```
> Y(0); Y(1); Y(3);
```

$$[t = 0,\ \mathrm{u}(t) = 2.]$$

$$[t = 1,\ \mathrm{u}(t) = 2.669716221353340]$$

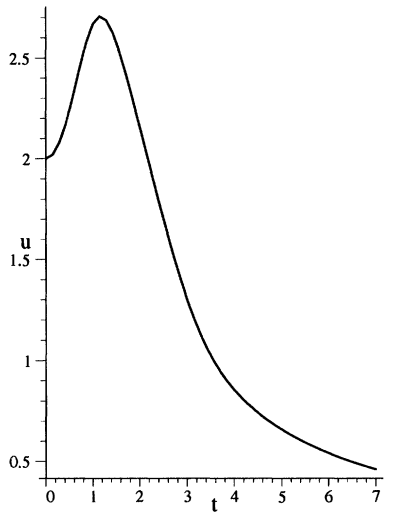
$$[t = 3,\ \mathrm{u}(t) = 1.297705136840902]$$

The function `Y` yields equations determining points of the numerical solution found with `dsolve`.

By additional arguments, the calculation method can be chosen and values of several variables controlling the calculations can be changed; see the on-line help to `dsolve,numeric`.

We can get a graph of the solution `Y` with the procedure `odeplot`, contained in the `plots` package.

```
> plots[odeplot]( Y , [t,u(t)] , 0..7 , labels=[t,u] );
```



The third argument determines the range of the first coordinate.

17.6 Series development of a solution

Another approach to approximation is a local Taylor series expansion. Let's apply this to the same ODE as in the previous section. First, let's set the value of `Order` to 20 to obtain a series expansion of order 20.

```
> Order:=20:
```

Let's apply `dsolve` with option `series`:

```
> dsolve( {deq4,u(0)=2} , u(t) , series );
```

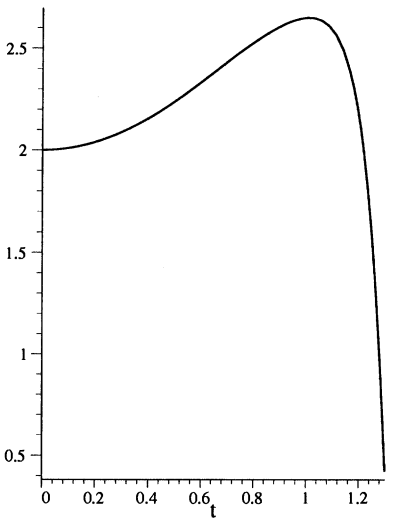
$$u(t) = 2 + t^2 - \frac{1}{12}t^4 - \frac{109}{360}t^6 - \frac{1247}{20160}t^8 + \frac{193657}{1814400}t^{10} + \frac{15397183}{239500800}t^{12} - \frac{1302729349}{43589145600}t^{14} - \frac{41217747869}{951035904000}t^{16} + \frac{78476591731}{291016986624000}t^{18} + O(t^{20})$$

Let's compare this solution with the numerical solution of the previous section by plotting it.

```
> convert( rhs(%) , polynom );
```

$$2 + t^2 - \frac{1}{12}t^4 - \frac{109}{360}t^6 - \frac{1247}{20160}t^8 + \frac{193657}{1814400}t^{10} + \frac{15397183}{239500800}t^{12} - \frac{1302729349}{43589145600}t^{14} - \frac{41217747869}{951035904000}t^{16} + \frac{78476591731}{291016986624000}t^{18}$$

```
> plot( % , t = 0 .. 1.3 );
```



In this picture we have taken a smaller range for `t` than previously in the numerical approach. From the graph of the direction field of the differential equation, it is clear that for $t > 1$ the series solution is losing relevance, while the numerical solution is a good approximation to the solution on a much wider range. In fact, there is a satisfying concordance between the numerical and the series solution for t between 0 and 0.5.

17.7 Systems of ODEs

Systems of differential equations are entered as sets. Here is an example, where `dsolve` can find a solution.

```
> deq1 := diff(f(t),t)=f(t)-g(t)-1;
```

$$deq1 := \frac{\partial}{\partial t}\,\mathrm{f}(t) = \mathrm{f}(t) - \mathrm{g}(t) - 1$$

```
> deq2 := diff(g(t),t)=f(t)-2*t;
```

$$deq2 := \frac{\partial}{\partial t}\,\mathrm{g}(t) = \mathrm{f}(t) - 2\,t$$

```
> dsolve( {deq1,deq2} , {f(t),g(t)} );
```

$$\left\{ g(t) = \frac{2}{3}\sqrt{3}\,e^{\left(\frac{1}{2}t\right)} \sin\left(\frac{1}{2}t\sqrt{3}\right) \_C1 + \_C2\, e^{\left(\frac{1}{2}t\right)} \cos\left(\frac{1}{2}t\sqrt{3}\right) - \frac{1}{3}\sqrt{3}\,e^{\left(\frac{1}{2}t\right)} \sin\left(\frac{1}{2}t\sqrt{3}\right) \_C2 - 1 + 2t,\right.$$

$$\left. f(t) = \_C1\, e^{\left(\frac{1}{2}t\right)} \cos\left(\frac{1}{2}t\sqrt{3}\right) + \frac{1}{3}\sqrt{3}\,e^{\left(\frac{1}{2}t\right)} \sin\left(\frac{1}{2}t\sqrt{3}\right) \_C1 - \frac{2}{3}\sqrt{3}\,e^{\left(\frac{1}{2}t\right)} \sin\left(\frac{1}{2}t\sqrt{3}\right) \_C2 + 2 + 2t \right\}$$

Let's check this solution, first for release V.5:

```
> odetest( % , {deq1,deq2} );
```

$$\{0\}$$

For earlier releases:

```
> subs( %% , {deq1,deq2} ):
> map( eq -> normal(lhs(eq)-rhs(eq)) , % );
```

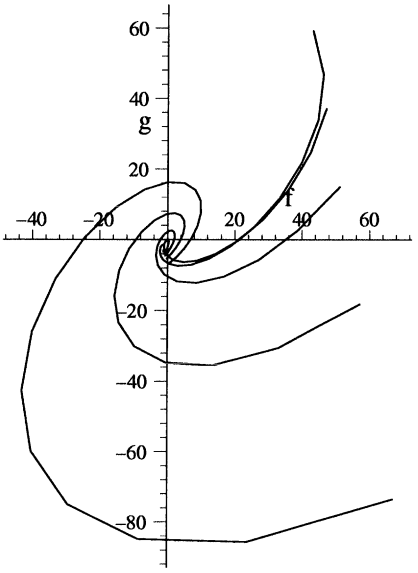
$$\{0\}$$

If Maple cannot find an exact solution of a system of ODEs, the system can be presented to `dsolve( ,series)`. Moreover, if it does not contain undetermined constants, then the set together with sufficient initial conditions can be handled with `dsolve( ,numeric)`.

Integral curves can be plotted with the procedure `phaseportrait` contained in the package `DEtools`. The arguments are here:

- a list of ODEs
- a list of the function names ("dependent variables")
- the range of the parameter t
- a set of initial conditions: lists of the form $[t, f(t), g(t)]$

```
> DEtools[phaseportrait]([deq1,deq2],[f,g],t=-2..5,
>    {[0,-2,-2],[0,0,0],[0,2,2],[0,5,5],[0,10,10]});
```



For a system of first-order ODEs with constant coefficients, the exponential of a matrix can be calculated with `linalg[exponential]`. This procedure is demonstrated in section 18.13 on page 263.

17.8 Helping Maple in solving ODEs

Because **piecewise** expressions can be handled well by `dsolve` (from release V.4), it can be useful to convert expressions and functions containing `abs`, `signum`, `Heaviside`, `max`, `min`, etc. to piecewise-defined functions with `convert( ,piecewise)`.

It can sometimes be helpful when **integral transforms** are applied in order to find a solution or a nicer solution. If you want the Laplace transform to be used, add the option `method=laplace`. Other methods are `fourier`, `fouriersin`, and `fouriercos`. These transforms are available separately in the `inttrans` package.

You may see a pattern in an ODE that Maple does not, perhaps a solution of its homogeneous variant. You can use this for manipulating that ODE, possibly resulting in an easier ODE that can be solved with `dsolve`. A special tool for manipulating differential equations is `PDEtools[dchange]`. This procedure can convert a differential equation by substituting variables. Here is an example:

```
> deq := x*diff(y(x),x,x) + (a*x^2-1)*diff(y(x),x) +
>     b*x^3 = 0;
```

$$deq := x\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) + \left(a\,x^2 - 1\right)\left(\frac{\partial}{\partial x}\,\mathrm{y}(x)\right) + b\,x^3 = 0$$

Let's substitute $\sqrt{t}$ for x and define $z(t) := y(\sqrt{t})$. This can be done with the procedure **PDEtools[dchange]**. We must indicate that a and b are constants by the third argument.

```
> PDEtools[dchange]({x=sqrt(t),y(x)=z(t)} , deq , {a,b});
```

$$2t\left(\frac{\frac{\partial}{\partial t}z(t)}{\sqrt{t}}+2\sqrt{t}\left(\frac{\partial^2}{\partial t^2}z(t)\right)\right)+2(at-1)\sqrt{t}\left(\frac{\partial}{\partial t}z(t)\right)+ bt^{(3/2)}=0$$

This result becomes a nice expression if it is multiplied by `sqrt(t)`.

The first argument to `PDEtools[dchange]` is a set of substitution rules, saying that the new variable is `t`, related to `x` by `x=sqrt(t)` and that the new function in `t` is `t -> z(t)`. The last argument says that `a` and `b` are constants.

In earlier releases, `PDEtools[dchangevar]` is not available. You might try `DEtools[Dchangevar]`. The same result could be generated in V.3 (but not in V.4) with:

```
> DEtools[Dchangevar]({x=sqrt(t),y(x)=z(t)} , " , {a,b});
```

$$2t\left(\frac{\frac{\partial}{\partial t}z(t)}{\sqrt{t}}+2\sqrt{t}\left(\frac{\partial^2}{\partial t^2}z(t)\right)\right)+2(at-1)\sqrt{t}\left(\frac{\partial}{\partial t}z(t)\right)+bt^{(3/2)}=0$$

This tool is much more restricted. For instance, the following cannot be done with `DEtools[Dchangevar]`:

```
> PDEtools[dchange]({x=exp(u),y(x)=p(u)} ,deq , {a,b} );
```

$$-\frac{\frac{\partial}{\partial u}p(u)}{e^u}+\frac{\frac{\partial^2}{\partial u^2}p(u)}{e^u}+\frac{\left(a\,(e^u)^2-1\right)\left(\frac{\partial}{\partial u}p(u)\right)}{e^u}+b\,(e^u)^3=0$$

17.9 Symbolic representations of solutions: DESol

Maple V can use symbolic representations of solutions of ODEs with the aid of `DESol`. First, let's use this for a basic standard equation:

```
> diff(y(x),x) = a*y(x);
```

$$\frac{\partial}{\partial x}y(x)=a\,y(x)$$

```
> DESol( % , y(x) );
```

$$\mathrm{DESol}\left(\left\{\left(\frac{\partial}{\partial x}y(x)\right)-a\,y(x)\right\},\{y(x)\}\right)$$

Such an expression can be manipulated with several procedures. For instance, it can be integrated:

```
> int(%,x);
```

$$\frac{\mathrm{DESol}(\{(\frac{\partial}{\partial x}\,\mathrm{y}(x)) - a\,\mathrm{y}(x)\},\{\mathrm{y}(x)\})}{a}$$

The last result says that the antiderivative of any solution of the present ODE is equal to this solution divided by a. In this case an explicit solution is available, by which we can affirm the last result.

```
> subs( DESol=dsolve , %% );
```

$$\mathrm{dsolve}\left(\left\{\left(\frac{\partial}{\partial x}\,\mathrm{y}(x)\right) - a\,\mathrm{y}(x)\right\},\{\mathrm{y}(x)\}\right)$$

```
> %;
```

$$\mathrm{y}(x) = \_C1\,\mathrm{e}^{(a\,x)}$$

Let's take another easy example, the ODE $y'' = a\,y$.

```
> DESol( diff(y(x),x,x)=a*y(x) , y(x) );
```

$$\mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) - a\,\mathrm{y}(x)\right\},\{\mathrm{y}(x)\}\right)$$

```
> des := %:
```

Let's differentiate `des`.

```
> diff( des , x );
```

$$\frac{\partial}{\partial x}\,\mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) - a\,\mathrm{y}(x)\right\},\{\mathrm{y}(x)\}\right)$$

Not very interesting. But if we differentiate this again, Maple can use the equation.

```
> diff( % , x );
```

$$a\,\mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) - a\,\mathrm{y}(x)\right\},\{\mathrm{y}(x)\}\right)$$

A `DESol` expression can be a part of an expression. For instance:

```
> des^3-1/des -5;
```

$$\mathrm{DESol}\left(\left\{\left(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)\right) - a\,\mathrm{y}(x)\right\},\{\mathrm{y}(x)\}\right)^3 - \frac{1}{\mathrm{DESol}(\{(\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)) - a\,\mathrm{y}(x)\},\{\mathrm{y}(x)\})} - 5$$

This can be integrated as well:

```
> int( % , x );
```

$$\mathrm{DESol}\left(\left\{ -\frac{2}{9}\frac{\frac{\partial^2}{\partial x^2}\,\mathrm{w}(x)^2}{\frac{\partial}{\partial x}\,\mathrm{w}(x)^{(5/3)}} + \frac{1}{3}\frac{\frac{\partial^3}{\partial x^3}\,\mathrm{w}(x)}{\frac{\partial}{\partial x}\,\mathrm{w}(x)^{(2/3)}} - a\frac{\partial}{\partial x}\,\mathrm{w}(x)^{(1/3)} \right\}, \{\,\mathrm{w}(x)\,\}\right) -$$
$$\mathrm{DESol}\left(\left\{ 2\frac{\frac{\partial^2}{\partial x^2}\,\mathrm{w}(x)^2}{\frac{\partial}{\partial x}\,\mathrm{w}(x)^3} - \frac{\frac{\partial^3}{\partial x^3}\,\mathrm{w}(x)}{\frac{\partial}{\partial x}\,\mathrm{w}(x)^2} - \frac{a}{\frac{\partial}{\partial x}\,\mathrm{w}(x)} \right\}, \{\,\mathrm{w}(x)\,\}\right) - 5\,x$$

The procedure `series` can be applied to a `DESol` expression.

```
> series( des , x , 10 );
```

$$\mathrm{y}(0) + \mathrm{D}(y)(0)\,x + \frac{1}{2}a\,\mathrm{y}(0)x^2 + \frac{1}{6}a\,\mathrm{D}(y)(0)\,x^3 + \frac{1}{24}a^2\,\mathrm{y}(0)x^4 + \frac{1}{120}a^2\,\mathrm{D}(y)(0)\,x^5 + \frac{1}{720}a^3\,\mathrm{y}(0)x^6 + \frac{1}{5040}a^3\,\mathrm{D}(y)(0)\,x^7 + \frac{1}{40320}a^4\,\mathrm{y}(0)x^8 + \frac{1}{362880}a^4\,\mathrm{D}(y)(0)\,x^9 + O(x^{10})$$

The `DESol` construction is an analogue to the `RootOf` construction. If there is no derivative, the first is converted to the second construction:

```
> DESol( y(x)^2=y(x)-1 , y(x) );
```

$$\mathrm{RootOf}(-\_Z + \_Z^2 + 1)$$

A set of initial conditions can be used as a third argument to `DESol`.

Sometimes results can be made more readable by using an alias for a `DESol` expression; see section B.1 on page 285.

17.10 Graphic tools for differential equations

The easiest method for plotting solutions is shown in section 17.5 on page 238 on numerical solutions: apply `dsolve( , ,numeric)` and then plot the solution with `plots[odeplot]`. For other purposes, the package `DEtools` offers `dfieldplot` and `phaseportrait`, which are shown earlier in this chapter.

17.11 More tools

In the present release 5 of Maple V, there is a new package `PDEtools` for manipulating and solving **partial differential equations** and for plotting solutions. See the on-line help for this package.

The package `linalg` contains several procedures for vector calculus, such as the procedure `linalg[potential]`, which calculates a function in several variables, if extant, when its first-order partial derivatives are given; and the procedure `linalg[vecpotent]`, which calculates a vector field, if extant, the curl of which is a given three-dimensional vector field.

chapter 18

Vectors and matrices

Maple supplies a special toolkit for linear algebra: the `linalg` *package. For matrices and vectors Maple uses a special data structure. This chapter explains how matrices and vectors can be created, changed, and handled.*

18.1 The linear algebra package

Before we can start calculations with vectors and matrices we must load the package `linalg`:

```
> with(linalg):
Warning, new definition for norm
Warning, new definition for trace
```

Maple's warnings indicate that the names `norm` and `trace` were referring to procedures, and that loading the `linalg` package made these names refer to procedures from the package. The "old" procedures can be reinstalled with the aid of `readlib`, for instance with `norm:=readlib('norm')`.

Asking for on-line help to the procedures from the `linalg` package is always possible in a way like "`?linalg[det]`" or "`?linalg,det`", but in most cases it is not necessary to use the name of the package; for instance, "`?det`" can be used as well.

Often, `Error` messages of `linalg` procedures are caused by applying them to singular matrices. The other important cause of error messages is this: if even one of the matrix elements contains a **floating-point number**, many of the procedures of this package try to switch to numerical algorithms. These can be applied only if all the elements of the matrix are numeric, and possibly complex. However, if the matrix contains indeterminates at the same time, only symbolic procedures can be used. In these cases, it might be necessary to convert the floating-point numbers contained in the matrix to rationals with the aid of `map`, as is shown in section 18.7 on page 254. The main cases of this question are discussed in this chapter.

Almost all symbolic algorithms used in this package are based on arithmetic operators and on handling polynomials. As Maple has a thorough command of these fields, the symbolic algorithms of `linalg` are **reliable**, generally. Only the test procedure `orthogonal` uses numerical testing, which might yield incorrect results in a rare exceptional case.

However, keep in mind that all unassigned names are interpreted as abstract numbers. Beware of the effects of substitution of special values for indeterminates, for instance causing matrices to be singular.

18.2 Creating vectors and matrices

A matrix can be created with the procedure `matrix`:

```
> A := matrix( [[ 5,4,1 ] , [ 3,-1,2 ] , [ -3,0,1 ]] ) ;
```

$$A := \begin{bmatrix} 5 & 4 & 1 \\ 3 & -1 & 2 \\ -3 & 0 & 1 \end{bmatrix}$$

The argument to `matrix` is a `list` of the rows of the matrix, each row being represented as the `list` of its elements. For a discussion of lists, see Chapter 10, *Manipulating several objects at once*. This list of lists can be derived back from a matrix: here for instance, with `convert( A , listlist )`.

A vector can be created with the procedure `vector`:

```
> cv := vector( [x,y,z] ) ;
```

$$cv := [x, y, z]$$

The procedure `vector` has been applied to the *list* of elements of the vector. This vector is printed *horizontally*, but it is interpreted by Maple as a **column vector** in matrix operations. It looks like a list. Compare:

```
> [x,y,z] , vector([x,y,z]);
```

$$[x, y, z], [x, y, z]$$

The corresponding **row vector** can be created with **transpose**:

```
> transpose( vector( [x,y,z] ) );
```

$$\mathrm{transpose}([x, y, z])$$

The result is to be read as the transpose of a column vector, thus as a row vector. It is also possible to think of a row vector as a matrix with one row:

```
> matrix( [[ x,y,z ]] );
```

$$\begin{bmatrix} x & y & z \end{bmatrix}$$

This one-by-three matrix is printed in exactly the same way as the *column* vector `vector([x,y,z])` in windowing versions. However, it is always possible to get extended information about a vector or matrix with `lprint`. For instance,

```
> lprint( % );
```

```
array(1 .. 1, 1 .. 3,[(1, 1)=x,(1, 2)=y,(1, 3)=z])
```

Vectors and matrices are one- and two-dimensional arrays, and here Maple explains that this one is an array with two indices, the first one ranging from 1 to 1, the second from 1 to 3; moreover, all its elements are presented.

Other ways of creating matrices and vectors are discussed later in this chapter.

18.3 Evaluation of vectors and matrices

In the previous section, we have assigned a matrix to the name `A`. If you simply type `A`, you don't see this matrix:

```
> A ;
```

$$A$$

It is the same as with procedures: a name referring directly to a matrix is not automatically evaluated further; for full evaluation, use `eval`:

```
> eval( A ) ;
```

$$\begin{bmatrix} 5 & 4 & 1 \\ 3 & -1 & 2 \\ -3 & 0 & 1 \end{bmatrix}$$

Later in this chapter, you will see that sometimes full evaluation requires a combination of `map` and `eval`: `map(eval,A)`.

By the same mechanism in Maple the name `A` at the right-hand side of the following assignment is evaluated to itself:

```
> B := A;
```

$$B := A$$

This makes `B` refer directly to `A` and not to the matrix itself:

```
> eval(B,1) ;
```

$$A$$

Automatic evaluation of `B` does not yield `B` itself, but the evaluation process reaches `A`, where it stops as this name is referring directly to a matrix:

```
> B;
```

$$A$$

The special rule of evaluation demonstrated here is:

> Automatic evaluation stops where it reaches
> a name that refers directly to a matrix or a vector.

This rule is called *the rule of evaluation up to the last name*. The same rule applies not only to vectors and matrices, but to tables in general and to procedures.

18.4 Elements of vectors and matrices

An element of a matrix can be selected with an index, using square brackets:

```
> A[2,1] ;
```

$$3$$

An element of the matrix can be changed with an assignment:

```
> A[1,2] := -2 ;
```

$$A_{1,2} := -2$$

More about what happens with such assignments can be found in section 18.8 on page 255. The input `A[1,2]` is printed as $A_{1,2}$ in windowing versions, but the square brackets are necessary in the input. Do not use as input `A12` instead, which yields the name $A12$, nor `A1,2`, which yields a sequence of the name $A1$ and the number 2. In section B.3 on page 287 some special aspects of so-called indexed names are discussed.

18.5 Matrix and vector arithmetic operators

For demonstration purposes in this section we use the following vector and matrices:

```
> vc := vector( [x,y] );
```

$$vc := [x, y]$$

```
> M1 := matrix( [[a,b],[c,d]] );
```

$$M1 := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> M2 := matrix( [[p,q],[r,s]] );
```

$$M2 := \begin{bmatrix} p & q \\ r & s \end{bmatrix}$$

In calculations with matrices and vectors the operators are lazy. For instance the sum:

```
> M1 + M2 ;
```

$$M1 + M2$$

is not calculated. You must execute matrix operations with the procedure **evalm**:

```
> evalm( % ) ;
```

$$\begin{bmatrix} a+p & b+q \\ c+r & d+s \end{bmatrix}$$

The product of `M1` and `M2` is calculated correctly with:

```
> evalm( M1 &* M2 );
```

$$\begin{bmatrix} ap+br & aq+bs \\ cp+dr & cq+ds \end{bmatrix}$$

> The product operator between matrices is to be denoted with **&***.
> Maple always interprets * as a commutative operator.

If `*` is used for a matrix product, the order of the factors is chosen by Maple before `evalm` comes into action, and so would yield a correct result only by chance.

The product operator between matrices and vectors must be denoted with `&*` as well:

```
> evalm( M1 &* vc );
```

$$[ax + by, cx + dy]$$

```
> evalm( transpose(vc) &* M1 );
```

$$\text{transpose}([ax + yc, xb + dy])$$

The product operator between a scalar and a vector or a matrix must be denoted with `*`:

```
> evalm( 3 * M1 );
```

$$\begin{bmatrix} 3a & 3b \\ 3c & 3d \end{bmatrix}$$

Scalar addition to a matrix may seem odd, but in such cases Maple interprets a scalar as a multiple of a suitable unit matrix:

```
> evalm(1000 + M1) ;
```

$$\begin{bmatrix} a+1000 & b \\ c & d+1000 \end{bmatrix}$$

Integer powers of matrices can be entered with `^`:

```
> evalm( M1 ^ 3 );
```

$$\begin{bmatrix} (a^2+bc)\,a+(ab+bd)\,c & (a^2+bc)\,b+(ab+bd)\,d \\ (ca+dc)\,a+(bc+d^2)\,c & (ca+dc)\,b+(bc+d^2)\,d \end{bmatrix}$$

```
> evalm( M1 ^ (-1) );
```

$$\begin{bmatrix} \frac{d}{a\,d-b\,c} & -\frac{b}{a\,d-b\,c} \\ -\frac{c}{a\,d-b\,c} & \frac{a}{a\,d-b\,c} \end{bmatrix}$$

The last result can also be calculated with the procedure `inverse`.

It is possible to substitute a matrix into a polynomial:

```
> x^2 - 5*x + 1;
```

$$x^2-5\,x+1$$

```
> subs(x=M1,%);
```

$$M1^2-5\,M1+1$$

```
> evalm(%);
```

$$\begin{bmatrix} a^2+bc-5\,a+1 & ab+bd-5\,b \\ ca+dc-5\,c & bc+d^2-5\,d+1 \end{bmatrix}$$

18.6 Manipulating all the elements of a matrix or vector at once

All elements of a matrix can be processed at once with the aid of the procedure **map**, discussed in section 10.3 on page 127. For instance, we have the matrix A:

```
> eval(A);
```

$$\begin{bmatrix} 5 & -2 & 1 \\ 3 & -1 & 2 \\ -3 & 0 & 1 \end{bmatrix}$$

We can square each element of A as follows:

```
> map( x -> x^2 , A );
```

$$\begin{bmatrix} 25 & 4 & 1 \\ 9 & 1 & 4 \\ 9 & 0 & 1 \end{bmatrix}$$

In fact, this is an exception to the general rule that `map` applies a procedure, the first argument to `map`, to the *operands* of the second argument; the operands of a matrix are not its elements:

```
> op( eval(A) );
```

$$1\ldots3, 1\ldots3, [2, 3 = 2, 3, 1 = -3, 3, 2 = 0, 3, 3 = 1, 1, 1 = 5, 1, 2 = -2, 1, 3 = 1, 2, 1 = 3, 2, 2 = -1]$$

but here map deviates from its normal rule in order to do what is useful for matrices in most cases.

In the same way, all the elements of a vector can be processed with the aid of map.

If a procedure such as simplify is to be applied to all the elements of a matrix, the procedure map can be used, too:

```
> matrix([[cos(phi),sin(phi)],[-sin(phi),cos(phi)]]);
```

$$\begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

```
> inverse(%);
```

$$\begin{bmatrix} \frac{\cos(\phi)}{\cos(\phi)^2+\sin(\phi)^2} & -\frac{\sin(\phi)}{\cos(\phi)^2+\sin(\phi)^2} \\ \frac{\sin(\phi)}{\cos(\phi)^2+\sin(\phi)^2} & \frac{\cos(\phi)}{\cos(\phi)^2+\sin(\phi)^2} \end{bmatrix}$$

```
> map( simplify , % , trig );
```

$$\begin{bmatrix} \cos(\phi) & -\sin(\phi) \\ \sin(\phi) & \cos(\phi) \end{bmatrix}$$

18.7 Processing a matrix that contains floating-point numbers

Several procedures in the linalg package switch over to numerical algorithms as soon as one of the elements of a matrix contains floating-point numbers, possibly complex. If the matrix contains indeterminates, this results in an error "matrix entries must all evaluate complex floats". For example:

```
> A:=matrix([[ 0.2 , a ],[ b/3. , 4 ]]);
```

$$A := \begin{bmatrix} .2 & a \\ .3333333333\,b & 4 \end{bmatrix}$$

```
> rank( % );
Error, matrix entries must all evaluate to complex floats
```

In such a case, the floats must be converted to rationals. See section 12.7 on page 156. See also the previous section for the use of map:

```
> map( convert , A , rational , exact );
```

$$\begin{bmatrix} \frac{1}{5} & a \\ \frac{3333333333}{10000000000} b & 4 \end{bmatrix}$$

```
> rank(%);
```

$$2$$

Observe that this result is correct only if a and b are interpreted as abstract items; substituting special values for a and b may result in rank equal to 1.

18.8 Names contained in elements of matrices and vectors

In substitutions, it is tempting to forget the special rule for evaluation of names referring to vectors and matrices. For instance,

```
> S := vector([ u^2 , u-1 , u+1 ]);
```

$$S := [u^2, u-1, u+1]$$

```
> subs(u=10,S);
```

$$S$$

This fails because S is not evaluated to the corresponding vector, so the procedure subs does not "see" the u contained in this vector. A correct method is:

```
> subs(u=10 , eval(S));
```

$$[100, 9, 11]$$

The result is a new vector object where u is exchanged for 10, as can be read from the result printed by Maple. Remember that the substitution has created a new vector and that S itself has not been changed by this command:

```
> eval(S) ;
```

$$[u^2, u-1, u+1]$$

If we *assign* a value to u we get the impression that this assignment does not influence the u in the matrix:

```
> u := 100;
```

$$u := 100$$

```
> eval(S);
```

$$[u^2, u-1, u+1]$$

This effect is caused by the fact that after the evaluation of a matrix or vector, the elements of the matrix are not evaluated. You can ask for this evaluation by applying eval to each of the elements of the matrix at once with the aid of map:

```
> map( eval , S );
```

$$[10000, 99, 101]$$

18.9 Determinant, basis, range, kernel, Gaussian elimination

Here is a short survey:

- `det` calculates the determinant of a matrix
- `Svd` calculates the singular values of a numeric matrix in combination with `evalf`, for example `evalf(Svd(A))`
- `trace` calculates the trace of a matrix
- `rank` calculates the rank of a matrix
- `basis` calculates a basis for the subspace spanned by the vectors in the list or set given as argument to `basis` as a sublist/subset of the given vectors
- `colspace` or `range` finds a basis for the range of a linear map described by a matrix as a left operator on column vectors. This basis is calculated by Gaussian elimination on the columns, resulting in a matrix in triangular form with leading entries equal to 1. The columns of this matrix that are not equal to zero yield a basis for the range of the given linear map.
- `colspan` does almost the same job as `colspace`, but it can only be used if all the items of the matrix are polynomials over the rationals. Generally the leading entries of the resulting matrix will not be equal to 1, but `colspan` has the advantage that the elements of the resulting vectors are also polynomials; it does not introduce quotients of polynomials.
- `rowspace` and `rowspan` do the same as `colspace` and `colspan`, respectively, for rows instead of columns
- `LUdecomp` computes the LU decomposition of a square matrix
- `intbasis` calculates a basis for the intersection of two linear subspaces
- `sumbasis` calculates a basis for the sum of two linear subspaces
- `gausselim` and `gaussjord` apply Gaussian elimination, yielding a matrix in row echelon form. In the case of `gaussjord` the nonzero leading entries in the resulting matrix are 1.
  These two procedures can be applied to a matrix containing quotients of polynomials over the rational complex numbers or only complex floating-point numbers and rational complex numbers. If necessary, floating-point numbers can be converted into rationals: see section 18.7 on page 254. The procedure `rref` is equal to `gaussjord`.
- `ffgausselim` applies fraction-free Gaussian elimination on a matrix of polynomials over the rationals

- `hermite` computes the reduced row echelon form of a matrix of polynomials in one variable with coefficients in the field of quotients of multivariate polynomials over the rationals
- `smith` computes the Smith normal form of a matrix of polynomials in one variable; `ismith` is a variant for calculations with integer matrices

18.10 Systems of linear equations

With the aid of Maple's **solve** procedure, large systems of linear equations can be solved in a fast and efficient way. Remember, that Maple conceives of variables as abstract items, so if a system contains parameters, and you want to substitute values for these parameters in the general solution, the result may not be the correct solution of the special problem. Here is an extremely easy example:

```
> sys := { a*x+3*y=7 , 6*x+2*a*y=14 };
```

$$sys := \left\{ a\,x + 3\,y = 7,\ 6\,x + 2\,a\,y = 14 \right\}$$

```
> solve( sys , {x,y} );
```

$$\left\{ x = 7\,\frac{1}{a+3},\ y = 7\,\frac{1}{a+3} \right\}$$

When you look at the result, you can see that it is not correct if `a=-3`. When you look at the system itself, you can see that the result is also not the correct solution if `a=3`. Let's pretend not to see this fact and use the methods from linear algebra. First, the homogeneous part of the system must be converted to a matrix by applying **genmatrix**:

```
> genmatrix( sys, [x,y] );
```

$$\begin{bmatrix} a & 3 \\ 6 & 2\,a \end{bmatrix}$$

The singular cases can be found by using the determinant:

```
> solve( det(%)=0 , a );
```

$$3,\ -3$$

These special cases can be handled by substituting 3 or -3 for `a` in the system and applying `solve` to these special systems. For instance,

```
> subs( a=3 , sys );
```

$$\{ 3\,x + 3\,y = 7,\ 6\,x + 6\,y = 14 \}$$

```
> solve( % , {x,y} );
```

$$\left\{ y = y,\ x = -y + \frac{7}{3} \right\}$$

The preceding example may show advantages of handling a system of linear equations in Maple with tools from the `linalg` package before `solve` is used. It is not necessary to use `solve` here; the package contains a special procedure for solving matrix equations: **linsolve**:

```
> B := matrix([[a,1,1],[1,a,1],[1,1,a]]);
```

$$B := \begin{bmatrix} a & 1 & 1 \\ 1 & a & 1 \\ 1 & 1 & a \end{bmatrix}$$

```
> w := vector([1,1,a^2]);
```

$$w := [1, 1, a^2]$$

The equation `B x = w` in vector `x` can be solved with:

```
> linsolve( B , w );
```

$$[-\frac{a}{a+2}, -\frac{a}{a+2}, \frac{a^2+2a+2}{a+2}]$$

Again it can be interesting to look at the cases where a special value of `a` causes `B` to be singular:

```
> solve(det(B)=0,a);
```

$$-2,\ 1,\ 1$$

Let's solve $B\,x = w$ in the case `a=1`. If we assign 1 to `a`, we get troubles: Maple calculates the solution before it evaluates `a` because of the special evaluation system of matrices and vectors:

```
> a:=1: linsolve(B,w);
Error, division by zero
```

Therefore, we substitute 1 for `a` in `B` and `w`:

```
> a:='a':
> subs( a=1 , [eval(B),eval(w)] );
```

$$\left[\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, [1, 1, 1] \right]$$

We have combined the matrix `B` and the vector `w` together in a list in order to apply substitution to both at once. The resulting list must be converted to a sequence with `op`, yielding the two arguments to `linsolve`:

```
> linsolve( op( % ) );
```

$$[1 - \_t_1 - \_t_2, \_t_1, \_t_2]$$

The result contains two parameters $\_t_1$ and $\_t_2$; the solution is two-dimensional. This is the standard way of parameterizing a solution found by `linsolve`. There-

fore, it is important to avoid assigning values to $\_t_1, \_t_2, \ldots$.

This same procedure `linsolve` can be used for solving equations of the following type: given two matrices A, where A is not singular, and B of suitable dimensions, for which matrix X is $AX = B$?

For **optimization problems in linear equations** the procedure `leastsqrs` is available: if `A` is a matrix and `w` is a vector, then `leastsqrs(A,w)` yields a vector `x` such that `A x - w` has minimal length, where the length is calculated as the square root of the sum of the squares of the coordinates.

For **optimization problems in linear inequalities** or **linear programming** the `simplex` package is available. Consult the on-line help about this package.

18.11 Characteristic polynomials and eigenvalues

The procedure `charpoly` computes the characteristic polynomial of a square matrix:

```
> M := matrix([[a,b^2],[25,a]]);
```

$$M := \begin{bmatrix} a & b^2 \\ 25 & a \end{bmatrix}$$

```
> charpoly(M,lambda);
```

$$\lambda^2 - 2\,\lambda\,a + a^2 - 25\,b^2$$

The procedure `eigenvals` calculates the eigenvalues of a square matrix:

```
> eigenvals(M);
```

$$-5\,b + a,\ 5\,b + a$$

The procedure `eigenvects` tries to calculate the eigenvectors of a square matrix:

```
> eigenvects(M);
```

$$[\ 5\,b + a,\ 1,\ \{\,[\frac{1}{5}\,b, 1]\,\}\,],\ [\,-5\,b + a,\ 1,\ \{\,[-\frac{1}{5}\,b, 1]\,\}\,]$$

The result is a sequence of two lists. The first element of each list is an eigenvalue of the matrix, followed by its multiplicity and a set of eigenvectors (a basis for the corresponding eigenspace). As usual, the variables `a` and `b` are considered by Maple as abstract items, not as unknown numbers, so the degenerated case where `b` equals zero is not included in the result.

When necessary, a `RootOf` expression is used in results of `eigenvals` and `eigenvects`. This is generally necessary if the dimension is higher than 3:

```
> hilbert(5);
```

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

```
> eigenvects(%);
```

$$\left[\frac{1}{5}\,\%1,\ 1,\ \left\{\left[\frac{1124794}{7875}\,\%1 - \frac{37956758}{15625}\,\%1^2 + \frac{310912}{125}\,\%1^3 - \frac{4346496}{15625}\,\%1^4 - \frac{81859}{7875000},\ \frac{5891}{33750} - \frac{6452734}{4725}\,\%1 + \frac{46132276}{1875}\,\%1^2 - \frac{1896832}{75}\,\%1^3 + \frac{1768704}{625}\,\%1^4,\ \frac{18880847}{7875}\,\%1 - \frac{4871447884}{109375}\,\%1^2 + \frac{5733056}{125}\,\%1^3 - \frac{80204544}{15625}\,\%1^4 - \frac{2752963}{3937500},\ 1,\ \frac{2720979364}{109375}\,\%1^2 - \frac{2040832}{1575}\,\%1 - \frac{1847177}{3937500} - \frac{641536}{25}\,\%1^3 + \frac{44884224}{15625}\,\%1^4\right]\right\}\right]$$

$$\%1 := \mathrm{RootOf}\left(85349376\_Z^5 - 762725376\_Z^4 + 741667248\_Z^3 - 40915248\_Z^2 + 61501\_Z - 1\right)$$

Here only one eigenvalue seems to be found, with multiplicity one, but this `RootOf` expression has two values that can be found with `allvalues( ,'d')`:

```
> allvalues(%);
```

$$\left[.3287928772\,10^{-5},\ 1,\ \{[-.008047359657,\ .1521038665,\ -.6597620813,\ 1,\ -.4904195315]\}\right],$$
$$\left[.0003058980402,\ 1,\ \{[.2023898935,\ -1.856745992,\ 2.863863673,\ 1,\ -2.392881055]\}\right],$$
$$\left[.01140749162,\ 1,\ \{[.6919626481,\ -2.339027826,\ -.3890937582,\ 1,\ 1.825713617]\}\right],$$
$$\left[.2085342186,\ 1,\ \{[-1.355862594,\ .621559,\ .9571437937,\ 1,\ .966454]\}\right],$$
$$\left[1.567050691,\ 1,\ \{[3.029605206,\ 1.76,\ 1.270834794,\ 1,\ .83]\}\right]$$

Apart from using `allvalues`, it is not convenient to manipulate the combinations of the eigenvalues and eigenvectors yielded by `eigenvects` as a whole; procedures such as `normal`, `expand`, and `simplify` must be applied separately to an eigenvalue, and, with the aid of the procedure `map`, to a corresponding eigenvector.

If a matrix contains floating-point numbers, the procedures `eigenvals` and `eigenvects` switch to numerical methods automatically. However, these numerical methods fail if the matrix contains indeterminates:

```
> M[2,1]:=.11:
> eval(M);
```

$$\begin{bmatrix} a & b^2 \\ .11 & a \end{bmatrix}$$

```
> eigenvects(M);
Error, matrix entries must all evaluate to float
```

In this case, all floating-point numbers must be converted to rationals. See section 18.7 on page 254.

```
> map(convert,M,rational);
```

$$\begin{bmatrix} a & b^2 \\ \frac{11}{100} & a \end{bmatrix}$$

```
> eigenvects(%);
```

$$\left[a+\frac{1}{10}\sqrt{11}\,b,\ 1,\ \left\{\left[\frac{10}{11}\sqrt{11}\,b,\ 1\right]\right\}\right],$$
$$\left[a-\frac{1}{10}\sqrt{11}\,b,\ 1,\ \left\{\left[-\frac{10}{11}\sqrt{11}\,b,\ 1\right]\right\}\right]$$

The procedure `eigenvects` for finding eigenvectors symbolically has restrictions on the type of the elements of the matrix: only algebraic expressions, possibly containing names, and `RootOf` expressions are allowed. Therefore, it fails in the following case:

```
> T:=matrix([[9*cos(phi)^2+43*cos(phi)+16,
>      50*(cos(phi)+1)^2],[-16+8*sin(phi)^2-16*cos(phi),
>      31*sin(phi)^2-37*cos(phi)-55]]);
```

$$T := \begin{bmatrix} 9\cos(\phi)^2+43\cos(\phi)+16 & 50\,(\cos(\phi)+1)^2 \\ -16+8\sin(\phi)^2-16\cos(\phi) & 31\sin(\phi)^2-37\cos(\phi)-55 \end{bmatrix}$$

```
> eigenvects(%);
Error, eigenvects only works for a matrix of rationals,
rational functions, algebraic numbers, or algebraic
functions at present
```

The procedure `eigenvects` cannot handle this matrix, as it contains `cos` and `sin`. Sometimes it is possible to avoid these restrictions by substituting. However, be on your guard. For instance, if `cos(phi)` and `sin(phi)` are replaced with `c` and `s` in the present case, then `eigenvects` finds two eigenvalues with two corresponding eigenvectors. However, both are equal if `c` and `s` are replaced with `cos(phi)` and `sin(phi)`. This is correct: T has only one eigenvalue and a one-dimensional eigenspace. Obviously, if you apply such a substitution trick, you must watch for possible relations between the elements of the matrix that might disappear with the substitution, leading to misleading results. A more sensible approach to the present problem is to apply `convert( ,tan)` or even only `simplify( ,trig)` to the matrix with the aid of `map`:

```
> map( simplify , T , trig );
```

$$\begin{bmatrix} 9\cos(\phi)^2 + 43\cos(\phi) + 16 & 50\cos(\phi)^2 + 100\cos(\phi) + 50 \\ -8 - 16\cos(\phi) - 8\cos(\phi)^2 & -24 - 37\cos(\phi) - 31\cos(\phi)^2 \end{bmatrix}$$

```
> subs( cos(phi)=c , % );
```

$$\begin{bmatrix} 9c^2 + 43c + 16 & 50c^2 + 100c + 50 \\ -8 - 16c - 8c^2 & -24 - 37c - 31c^2 \end{bmatrix}$$

```
> eigenvects( % );
```

$$\left[-4 + 3c - 11c^2,\ 2,\ \left\{\left[\tfrac{-5}{2} \quad 1\right]\right\}\right]$$

```
> subs( c=cos(phi) , % );
```

$$\left[-4 + 3\cos(\phi) - 11\cos(\phi)^2,\ 2,\ \left\{\left[\tfrac{-5}{2} \quad 1\right]\right\}\right]$$

The **Jordan matrix** of a matrix can be calculated with `jordan`, the **minimal polynomial** with `minpoly`, and the **companion matrix** with `companion`. Examples can be found in the on-line help of Maple.

18.12 Dot product, cross product, norms, and orthogonal systems

Here is a short survey:

- `dotprod` calculates the interior product of two vectors in complex space, supposing that each unassigned name occurring in the elements of the vectors stands for a real number
- `angle` computes the angle between two vectors
- `crossprod` calculates the cross product of two three-dimensional vectors
- `norm` computes the norm of a vector or matrix according to the norm definition *specified by the second argument.* If no second argument is present, the norm of a vector yields the maximum of the absolute values of its elements. For the square root of the sum of the squares of the elements use `norm( ,frobenius)`.

- `GramSchmidt` computes an orthogonal basis from a given basis
- `QRdecomp` computes the QR decomposition of a square matrix
- `orthog` tests a matrix for orthogonality, using a clever but numerical test on equalities

18.13 Vector calculus

The standard differential operators of vector calculus, divergence, Laplacian, gradient, curl, hessian, Jacobian, and Wronskian are available as `diverge`, `grad`, `curl`, `hessian`, `jacobian`, and `Wronskian`. Moreover, `potential` determines whether a given vector field is the gradient of an expression; if so, it can assign such an expression to a name given as a third argument. The procedure `vecpotent` determines whether a given three-dimensional vector (or list) of expressions is the curl of a vector field; if so, this field can be assigned to a name given as a third argument. Here is an example, where we indicate with an option that the coordinates are meant to be spherical:

```
> grad(cos(phi)/r,[r,phi,theta],coords=spherical);
```

$$\left[-\frac{\cos(\phi)}{r^2}, -\frac{\sin(\phi)}{r^2}, 0\right]$$

When the option `coords=spherical` had been omitted, the result is:

```
> grad(cos(phi)/r,[r,phi,theta]);
```

$$\left[-\frac{\cos(\phi)}{r^2}, -\frac{\sin(\phi)}{r}, 0\right]$$

Moreover, in release 5 there is a package `codegen`, which offers the procedures `GRADIENT`, `JACOBIAN`, and `HESSIAN`. These are meant to handle procedures, not expressions; the output is also a procedure. If you apply such a resulting procedure, you might need to apply `eval` to the values, due to a bug, that might be fixed later. These procedures can only work with the standard Cartesian coordinates (in release 5).

For differential equations, the **exp of a matrix** can be important:

$$exp(tA) := \sum_{n=0}^{\infty} (tA)^n$$

This can be calculated with the procedure **exponential**. Here is an example:

```
> M := matrix( [[a + 4, 10], [-2, a - 5]] );
```

$$M := \begin{bmatrix} a+4 & 10 \\ -2 & a-5 \end{bmatrix}$$

```
> exponential( M , t );
```

$$\begin{bmatrix} 5\,e^{(a\,t)} - 4\,e^{((a-1)\,t)} & -10\,e^{((a-1)\,t)} + 10\,e^{(a\,t)} \\ 2\,e^{((a-1)\,t)} - 2\,e^{(a\,t)} & -4\,e^{(a\,t)} + 5\,e^{((a-1)\,t)} \end{bmatrix}$$

As in many other cases, the matrix is not allowed to contain floating-point numbers if it also contains symbolic elements, so conversion of the floating-point numbers to rationals may be necessary. See section 18.7 on page 254.

18.14 Creating new vectors and matrices from old ones by changing elements

Suppose you have obtained a matrix, `M1`:

```
> eval(M1);
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

and you want to create another matrix, `M2`, by changing one or more elements of `M1`, while preserving the value of `M1`. The following assignment cannot fulfill these demands:

```
> M2 := M1;
```

$$M2 := M1$$

The right-hand side `M1` is not evaluated further, as you can see in the output of Maple, according to the rule of evaluation of matrices up to the last name, so `M2` is made to refer to `M1`. Changing an element of `M2` changes the corresponding element of `M1` as well:

```
> M2[1,1] := aaa;
```

$$M2_{1,1} := aaa$$

```
> eval(M1);
```

$$\begin{bmatrix} aaa & b \\ c & d \end{bmatrix}$$

That is not what we want.

Let's reset `M1` and then try another way by urging Maple to evaluate fully with the procedure `eval`.

```
> M1[1,1]:=a:
> M2 := eval(M1);
```

$$M2 := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The effect is that M2 refers to a matrix object directly. This can be seen by evaluating M2 just one step:

```
> eval(M2,1);
```

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Let's change an element of M2 and look at the result:

```
> M2[1,2] := bbb;
```

$$M2_{1,2} := bbb$$

```
> eval(M2) , eval(M1);
```

$$\begin{bmatrix} a & bbb \\ c & d \end{bmatrix}, \begin{bmatrix} a & bbb \\ c & d \end{bmatrix}$$

The output of Maple shows that M1 has been changed by changing M2. That is because both names are referring to the same matrix object in memory.

> Changing an element of a matrix (or vector)
> does not create a new matrix (or vector) object in the memory,
> but it changes only the existing object.

However, it is possible to create a new matrix object in the memory as a copy of an existing one by using **copy**. If this object is assigned to the name M2, we find two independent matrix objects:

```
> M1[1,2] := b:
> M2 := copy(M1) ;
```

$$M2 := \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

```
> M2[2,2] := 2222222;
```

$$M2_{2,2} := 2222222$$

```
> eval( M2 ) , eval( M1 ) ;
```

$$\begin{bmatrix} a & b \\ c & 2222222 \end{bmatrix}, \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Instead of copy the procedure evalm could have been used as well, but this is much slower than the special procedure copy.

18.15 Creating new matrices from old ones by transposing, cutting, and pasting

Here is a short survey of available procedures for transposing matrices, pasting matrices and vectors together, or cutting out parts of matrices. All these procedures do not change an existing matrix or vector, but create a new one that can be assigned to a name for later use.

- `transpose` transposes a matrix (or a column vector, yielding a row vector, or vice versa)
- `htranspose` (Hermitian transpose) transposes a matrix and takes the complex conjugate of the elements
- `concat` pastes matrices and/or vectors together side by side (horizontally)
- `stackmatrix` (older releases: `stack`) pastes matrices and/or vectors together bottom to top (vertically)
- `delcols` deletes columns of a matrix
- `delrows` deletes rows of a matrix
- `col` extracts a column of a matrix as a vector
- `row` extracts a row of a matrix; the result is a *column* vector
- `submatrix` extracts a submatrix from a matrix
- `subvector` extracts a vector from a matrix
- `copyinto` copies the entries of a matrix as a block into another matrix
- `extend` creates a matrix by adding columns and rows to the original matrix

18.16 Alternative ways of creating vectors and matrices

It is possible to create a matrix without assigning values to its elements. For instance,

```
> X := matrix(3,4);
```

$$X := \text{array}\,(1\ldots3, 1\ldots4, [\,])$$

The result is described as an **array**. In fact, matrices and vectors are special types of Maple `arrays`; the same object `X` can be created with the command `X := array(1 .. 3, 1 .. 4 ) ;`.

We can assign values to one or more elements of this matrix:

```
> X[1,1]:=0;
```

$$X_{1,1} := 0$$

If we evaluate `X`, we see:

```
> eval(X);
```

$$\begin{bmatrix} 0 & ?_{1,2} & ?_{1,3} & ?_{1,4} \\ ?_{2,1} & ?_{2,2} & ?_{2,3} & ?_{2,4} \\ ?_{3,1} & ?_{3,2} & ?_{3,3} & ?_{3,4} \end{bmatrix}$$

Printing `X` yields a slightly nicer picture:

```
> print(X);
```

$$\begin{bmatrix} 0 & X_{1,2} & X_{1,3} & X_{1,4} \\ X_{2,1} & X_{2,2} & X_{2,3} & X_{2,4} \\ X_{3,1} & X_{3,2} & X_{3,3} & X_{3,4} \end{bmatrix}$$

It is possible to create a matrix by using a *function* that calculates each element of the matrix from its coordinates. For instance, let's create the 8 by 8 matrix where the element on position $[i,j]$ is $\binom{i-1}{j-1} 2^{1-i}$ if $i \geq j$ and 0 otherwise:

```
> matrix( 8 , 8 , (i,j) ->
>     if i >= j then binomial(i-1,j-1)*2^(1-i)
>     else 0 fi );
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{4} & \frac{3}{8} & \frac{1}{4} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{32} & \frac{5}{32} & \frac{5}{16} & \frac{5}{16} & \frac{5}{32} & \frac{1}{32} & 0 & 0 \\ \frac{1}{64} & \frac{3}{32} & \frac{15}{64} & \frac{5}{16} & \frac{15}{64} & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{128} & \frac{7}{128} & \frac{21}{128} & \frac{35}{128} & \frac{35}{128} & \frac{21}{128} & \frac{7}{128} & \frac{1}{128} \end{bmatrix}$$

There are two procedures for creating **random matrices and vectors**: `randmatrix` and `randvector`.

18.17 Special types of matrices: (anti)symmetric, sparse, identity

Some special types of matrices can be created by the use of **index functions** and the procedure **array**:

- A **symmetric matrix** can be created with:

```
> S1 := array( 1..4 , 1..4 , symmetric );
```

$$S1 := \text{array}\,(symmetric, 1\ldots4, 1\ldots4, [\,])$$

Observe that the dimensions of an array are entered as ranges, in contrast to the procedures `matrix` and `vector` which require natural numbers for the dimensions.

We can see that `S1` is symmetric by printing it:

```
> print(S1);
```

$$\begin{bmatrix} S1_{1,1} & S1_{1,2} & S1_{1,3} & S1_{1,4} \\ S1_{1,2} & S1_{2,2} & S1_{2,3} & S1_{2,4} \\ S1_{1,3} & S1_{2,3} & S1_{3,3} & S1_{3,4} \\ S1_{1,4} & S1_{2,4} & S1_{3,4} & S1_{4,4} \end{bmatrix}$$

If elements are assigned to, symmetry is used, too:

```
> S1[2,3]:=0: S1[4,1]:=100:
> print(S1);
```

$$\begin{bmatrix} S1_{1,1} & S1_{1,2} & S1_{1,3} & 100 \\ S1_{1,2} & S1_{2,2} & 0 & S1_{2,4} \\ S1_{1,3} & 0 & S1_{3,3} & S1_{3,4} \\ 100 & S1_{2,4} & S1_{3,4} & S1_{4,4} \end{bmatrix}$$

Only the right upper triangle of the matrix is stored in memory; the indices are exchanged, if necessary.

It is also possible to enter the values of the elements in the invocation of `array`. As the elements are assigned row by row from the left, we should enter only the elements of the left under triangle of the matrix, so the first element of the first row, then the two first elements of the second row, and so on:

```
> array( symmetric , 1..4 , 1..4 ,
>     [ [1] , [2,3] , [4,5,6] , [7,8,9,10] ] );
```

$$\begin{bmatrix} 1 & 2 & 4 & 7 \\ 2 & 3 & 5 & 8 \\ 4 & 5 & 6 & 9 \\ 7 & 8 & 9 & 10 \end{bmatrix}$$

- An **antisymmetric matrix** can be created in a comparable way:

```
> print( array( antisymmetric , 1..4 , 1..4 ) );
```

$$\mathrm{array}(antisymmetric, 1\ldots4, 1\ldots4)$$

The elements of such a matrix are to be entered by assignments, possibly using the repetitional control structure discussed later in this section.

- A **sparse matrix** can be created as follows:

```
> Sp := array( 1..6 , 1..6 , sparse );
```

$$Sp := \mathrm{array}\,(sparse, 1\ldots6, 1\ldots6, [\,])$$

In a sparse matrix all elements that are not assigned to are zero:

```
> Sp[2,2]:=777;
```

$$Sp_{2,2} := 777$$

```
> print(Sp);
```

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 777 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

For sparse arrays, only the elements that are assigned to are stored in memory. If huge sparse matrices are to be handled, this feature can avoid unnecessary seizure of memory.

- An **identity matrix** can be created with:

```
> Id := array( 1..4 , 1..4 , identity );
```

$$Id := \text{array}\,(identity, 1\ldots4, 1\ldots4, [\,])$$

```
> print(Id);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- A diagonal matrix can be created with the index function `diagonal`, but the procedure `diag` is generally preferable.

In section 18.16 on page 266 we created a matrix with the command

```
> matrix( 8 , 8 , (i,j) -> if i >= j then
>     binomial(i-1,j-1)*2^(1-i) else 0 fi);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{4} & \frac{3}{8} & \frac{1}{4} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{32} & \frac{5}{32} & \frac{5}{16} & \frac{5}{16} & \frac{5}{32} & \frac{1}{32} & 0 & 0 \\ \frac{1}{64} & \frac{3}{32} & \frac{15}{64} & \frac{5}{16} & \frac{15}{64} & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{128} & \frac{7}{128} & \frac{21}{128} & \frac{35}{128} & \frac{35}{128} & \frac{21}{128} & \frac{7}{128} & \frac{1}{128} \end{bmatrix}$$

Such a construction is not available for arrays. However, it is easy to replace this with a **control structure, repetition**. For instance, a sparse matrix with the same elements can be created as follows:

```
> SB := array( 1..8 , 1..8 , sparse );
```

$$SB := \text{array}\,(sparse, 1\ldots8, 1\ldots8, [\,])$$

```
> for i to 8 do
>   for j to i do
>     SB[i,j] :=  binomial(i-1,j-1)*2^(1-i)
>     od
>   od:
> print(SB);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{4} & \frac{3}{8} & \frac{1}{4} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{32} & \frac{5}{32} & \frac{5}{16} & \frac{5}{16} & \frac{5}{32} & \frac{1}{32} & 0 & 0 \\ \frac{1}{64} & \frac{3}{32} & \frac{15}{64} & \frac{5}{16} & \frac{15}{64} & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{128} & \frac{7}{128} & \frac{21}{128} & \frac{35}{128} & \frac{35}{128} & \frac{21}{128} & \frac{7}{128} & \frac{1}{128} \end{bmatrix}$$

Do not forget to close a repetition structure with `od`. More information on this structure can be found by entering `?do`.

As an alternative we can use the procedure `seq`:

```
> SB := array( 1..8 , 1..8 , sparse ,
>     [seq(
>        [ seq(binomial(i-1,j-1)*2^(1-i),j=1..i) ],
>     i=1..8)]);
```

$$SB := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{1}{4} & \frac{3}{8} & \frac{1}{4} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{32} & \frac{5}{32} & \frac{5}{16} & \frac{5}{16} & \frac{5}{32} & \frac{1}{32} & 0 & 0 \\ \frac{1}{64} & \frac{3}{32} & \frac{15}{64} & \frac{5}{16} & \frac{15}{64} & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{128} & \frac{7}{128} & \frac{21}{128} & \frac{35}{128} & \frac{35}{128} & \frac{21}{128} & \frac{7}{128} & \frac{1}{128} \end{bmatrix}$$

18.18 Creating more special types of matrices

There are several procedures for creating matrices of a special type, for instance, the procedure **diag** for creating **diagonal matrices** and **block diagonal matrices**:

```
> diag(1,4,7);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 7 \end{bmatrix}$$

```
> diag( matrix([[a,b],[c,d]]) , p , q );
```

$$\begin{bmatrix} a & b & 0 & 0 \\ c & d & 0 & 0 \\ 0 & 0 & p & 0 \\ 0 & 0 & 0 & q \end{bmatrix}$$

```
> diag( JordanBlock(lambda,3) , JordanBlock(mu,2) , nu );
```

$$\begin{bmatrix} \lambda & 1 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 1 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 1 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \nu \end{bmatrix}$$

The procedure `band` creates **band matrices**:

```
> band( [1,2,3,4,5], 8 );
```

$$\begin{bmatrix} 3 & 4 & 5 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 4 & 5 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 4 & 5 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 4 & 5 & 0 \\ 0 & 0 & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{bmatrix}$$

The procedures `blockmatrix`, **hilbert**, **sylvester**, **toeplitz**, and **vandermonde** can create matrices corresponding to the names of these procedures.

18.19 Functions yielding vectors and matrices

A function yielding vectors or matrices can be created with the aid of the **arrow** ($\rightarrow$) when the dimensions are entered as additional arguments:

```
> vf := t -> vector( 3 , [t-1,t+1,t^2] );
```

$$vf := t \rightarrow [t-1, t+1, t^2]$$

```
> vf(100);
```

$$[99, 101, 10000]$$

In this example, the first argument to `vector` says that a three-element vector is to be created. Here is an example of a function yielding matrices:

```
> mf:= phi -> matrix( 2 , 2 ,
>     [[cos(phi),sin(phi)],[-sin(phi),cos(phi)]]);
```

$$mf := \phi \rightarrow \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix}$$

```
> mf(Pi/6);
```

$$\begin{bmatrix} \frac{1}{2}\sqrt{3} & \frac{1}{2} \\ \frac{-1}{2} & \frac{1}{2}\sqrt{3} \end{bmatrix}$$

Let's suppose that we have found a matrix containing a parameter:

```
> Ma := matrix([[a,a^2],[a^3,a^4]]);
```

$$Ma := \begin{bmatrix} a & a^2 \\ a^3 & a^4 \end{bmatrix}$$

Suppose that we want to create a function with parameter a yielding a matrix according to the next expression. We could do so with the arrow construction as in the previous example, but then the matrix must be typed in again; suppose that we want to avoid that. The usual trick is using the procedure **unapply**, but this does not work here without additional tricks. The easiest way to create a function from `Ma` is as follows:

```
> F := t -> subs(a=t,eval(Ma));
```

$$F := t \rightarrow \mathrm{subs}(a = t, \mathrm{eval}(Ma))$$

```
> F(3);
```

$$\begin{bmatrix} 3 & 9 \\ 27 & 81 \end{bmatrix}$$

The same trick can be used for vectors.

In release V.5 you can also use **codegen[makeproc]**, but at the moment of release it is not very suitable for this purpose; due to a bug the user must add `eval` for the results.

18.20 Vectors and matrices modulo an integer

Matrices with elements in Z **modulo** an integer can be used as in the following example:

```
> A := matrix([[22,-4,-55],[88,33,7777],[333,-8,-1111]]);
```

$$A := \begin{bmatrix} 22 & -4 & -55 \\ 88 & 33 & 7777 \\ 333 & -8 & -1111 \end{bmatrix}$$

```
> map(`mod`,A,5);
```

$$\begin{bmatrix} 2 & 1 & 0 \\ 3 & 3 & 2 \\ 3 & 2 & 4 \end{bmatrix}$$

The reason for the use of the **back quotes** here is that `mod` is a keyword, which can only be used as a name of a procedure by enclosing it between back quotes. The resulting matrix contains integers, not elements of Z mod 5, so each new operation is to be combined with `mod` again. For instance,

```
> evalm( %^2 );
```

$$\begin{bmatrix} 7 & 5 & 2 \\ 21 & 16 & 14 \\ 24 & 17 & 20 \end{bmatrix}$$

```
> map(`mod`,%,5);
```

$$\begin{bmatrix} 2 & 0 & 2 \\ 1 & 1 & 4 \\ 4 & 2 & 0 \end{bmatrix}$$

There are some special linear algebra procedures in connection with `mod`:

```
> Det(A) mod 5;
```

$$0$$

```
> Nullspace(A) mod 5;
```

$$\{[4, 2, 1]\}$$

The last two steps have called **inert procedures**, which are activated by `mod`. The other inert procedures for linear algebra that can be activated by `mod` are `Gausselim`, `Gaussjord`, `Smith`, and `Nullspace`.

For calculations with rings, Maple offers the **Gauss** package, which also can be used for calculations on matrices with elements in a ring created by `Gauss`.

18.21 Reading a matrix of data from a file

Suppose that a file is available that contains numerical data in some columns. Assume, as an easy example, that the file `exa` looks like:

```
 0.1   2    3.3
0.04  5.5   0.6
 77    8    9.99
 10   1.1   0.12
```

We can read these data into a matrix as follows:

```
> matrix( readdata(exa,3) );
```

$$\begin{bmatrix} .1 & 2. & 3.3 \\ .04 & 5.5 & .6 \\ 77. & 8. & 9.99 \\ 10. & 1.1 & .12 \end{bmatrix}$$

The number 3 in this command corresponds to the number of columns to be read. More details can be found in the on-line help.

18.22 Pedagogical facilities

As explained in section 4.18 on page 62, Maple can be used excellently for teaching mathematics. This is also the case for linear algebra. The `linalg` package even contains some special procedures for this purpose. For explanations and examples, consult the on-line help of Maple:

- `addcol`, `addrow`, `mulcol`, `mulrow`, `multiply`, `swapcol`, `swaprow`, and `pivot` for executing Gauss elimination step by step, where the user must choose the steps and can leave the calculations to Maple
- `backsub` for the last step after Gauss elimination in solving a system of linear equations by matrix operations
- `plots[matrixplot]` and `plots[sparsematrixplot]` for visualizing a matrix

.

appendix A

Types, properties, and domains

Types of objects are important in the Maple system. In the interactive use of Maple, some knowledge about types can be useful; types often appear in error messages. When you manipulate a very large Maple object, asking for the basic type of this object may help to reveal its structure; components of a special type can be singled out from an expression, set, or list with `select` *or* `remove`.

Don't get confused by the fact that some other computer languages have typed variables. In Maple, each name can refer to any other Maple object, whatever the type of the object, but the user can tell Maple to assume a special property *for a name or an expression. This facility can help Maple calculations and is discussed here.*

Mathematics owes much of its power to abstraction; in algebra especially, many theorems and algorithms are not restricted to a special number system, but can be used for a whole class of groups or rings. The `Domains` *package is a start in the direction of generalization to computations in classes of rings. The basic idea of this package is demonstrated at the end of this appendix.*

A.1 Basic types

Each Maple object has a basic type. This type can be found with the procedure **whattype**. Some of these basic types are discussed in section 11.3 on page 139 and section 11.4 on page 141. Here are some examples:

```
> x,whattype( x );
```

$$x, \mathit{symbol}$$

In releases before V.5, the last result would be $string$.

```
> 5/3, whattype( 5/3 );
```

$$\frac{5}{3}, \mathit{fraction}$$

```
> sqrt(5)/3, whattype( sqrt(5)/3 );
```

$$\frac{1}{3}\sqrt{5}, *$$

Observe that a rational number is called a $fraction$, and that other quotients are stored as products.

```
> sqrt(5), whattype( sqrt(5) );
```

$$\sqrt{5},\ \hat{}$$

```
> exp(5), whattype( exp(5) );
```

$$e^5,\ function$$

Internally e^5 is an unevaluated function call of the procedure `exp`, so it has the Maple type `function`.

```
> whattype( 5 , sqrt(3) );
```

$$exprseq$$

```
> M := linalg[matrix]([[1,2],[3,4]]); whattype(M);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$symbol$$

```
> whattype( eval(M) );
```

$$array$$

```
> abc[3], whattype( abc[3] );
```

$$abc_3,\ indexed$$

The last type, `indexed`, is discussed in section B.3 on page 287. A list of the 28 basic types in Maple can be found in the on-line help to `whattype`. For each of these types, separate on-line help is available.

A.2 More types

Apart from these basic types, many more types are available. These can be used in the procedures `type` and `hastype`. For instance,

```
> type( 5/3 , numeric );
```

$$true$$

```
> type( 5/3 , polynom );
```

$$true$$

The following two results may make you wonder:

```
> type( sqrt(5) , numeric );
```

$$false$$

```
> type( sqrt(5) , positive );
```

$$false$$

Remember that $\sqrt{5}$ is only a symbolic object to Maple, but Maple *can* determine that $\sqrt{5}$ is a positive number, for instance, if the procedure `is` or `signum` is applied.

The name `M` has been assigned a matrix in the previous section. The procedure `type` is very compliant here:

```
> type( M , symbol );
```

$$true$$

```
> type( M , table );
```

$$true$$

```
> type( M , matrix );
```

$$true$$

```
> type( M , array );
```

$$true$$

(In releases before V.5, the type `string` is used instead of `symbol`; in release V.5 a new basic type `string` has been added.)
A special case is a name referring to a sequence. For instance,

```
> sq:=5,sqrt(3);
```

$$sq := 5, \sqrt{3}$$

```
> type( sq , exprseq );
Error, wrong number (or type) of parameters in function type
```

Here `type` reports an error, because it gets three arguments (`5,sqrt(3), exprseq`) instead of two. The procedure `type` can handle all basic types except for `exprseq`: names referring to a sequence can only be handled by `whattype`.

```
> whattype( sq );
```

$$exprseq$$

A list of the standard types that can be used with the procedures `type` and `hastype` can be found in the on-line help about `type`. More complicated types can be composed from these standard types as *structured types*; it is even possible to program new types.

A.3 Selection on type

All operands of a special type in a list, set, sum, or product can be singled out with the procedures **select** and **remove**. See section 10.6 on page 131.

```
> 2 + sqrt(3) - sqrt(5)*x + sqrt(7);
```

$$2+\sqrt{3}-\sqrt{5}\,x+\sqrt{7}$$

```
> select( x->type(x,radical) , % );
```

$$\sqrt{3}+\sqrt{7}$$

```
> remove( x->type(x,'*') , %% );
```

$$2+\sqrt{3}+\sqrt{7}$$

Observe that the first result is a sum and that $\sqrt{5}$ is not contained in the result as this is not one of the four operands of the sum.

A.4 Properties, the assume facility

A property can be given to a name or expression with the procedure **assume**. For instance,

```
> assume( t > 0 );
```

From now on, the name `t` is evaluated as being an undefined positive number. That property can be used in calculations:

```
> 'limit(exp(t*x),x=infinity),
>       limit(exp(-t*x),x=infinity)';
```

$$\lim_{x\to\infty} \mathbf{e}^{(t\,x)},\ \lim_{x\to\infty} \mathbf{e}^{(-t\,x)}$$

```
> %;
```

$$\infty,\ 0$$

It is also possible to assume a property for an expression:

```
> assume( r^2-1 >=0 );
```

After evaluation, the names `t` and `r` are printed with a **tilde**, indicating that they have some property.

```
> t,r;
```

$$t\sim,\ r\sim$$

An assumption concerning a name can be removed by unassigning:

```
> r:='r':
> about(r^2-1);
r^2-1:
  nothing known about this object
```

The procedure `assume` can be used for assuming properties for more than one expression at the same time, for instance, that $\ln(a) > 2$ and that b is an integer:

```
> assume( ln(a)>2 , b , integer );
```

This example also shows that there are two forms for assuming something: a relation, such as `(ln(a)>2)`, or a pairing of an expression and a property, such as `(b,integer)`.

A.5 Derived properties

Properties can be derived from a known property with the procedure **is**:

```
> is(t,real);
```

$$true$$

```
> is(t+1,positive);
```

$$true$$

```
> is(exp(t)>1);
```

$$true$$

```
> is(t-1,positive);
```

$$false$$

The first three results indicate that for each positive number t, this t is real, that $t + 1$ is positive, and that $\exp(t)$ is greater than 1. The last result indicates that there is a positive number t such that $t - 1$ is not positive.

A.6 Asking for the assumed properties

We can ask for the properties of `t` with the procedure **about**:

```
> about(t);
Originally t, renamed t~:
  is assumed to be: RealRange(Open(0),infinity)
```

The message `t~ is assumed to be: RealRange(Open(0),infinity)` means

$$t \in < 0, \infty$$

A.7 Adding properties

If you `assume` something for an expression or variable that has a property, the old property is discarded. It is also possible to add a property: if you want to assume that t is not only positive, but even a positive integer, then you can use the procedure **additionally**:

```
> additionally(t,integer);

> about(t);
Originally t, renamed t~:
  is assumed to be: AndProp(integer,RealRange(1,infinity))
```

A.8 Combining properties

In the last example you can see a way in which properties can be combined: with **AndProp**, indicating that `t` is assumed to be negative *and* integer.

There is another way of combining properties: **OrProp**. We can tell Maple that `t` is positive *or* negative with:

```
> assume(t,OrProp(positive,negative));
```

Essentially, this is the same as saying that `t` is a nonzero real number. We can ask if `t` is assumed to be real with the procedure `is`:

```
> is(t,real);
```

true

```
is(t,Non(0));
```

true

The method by which Maple can derive this is by applying a table of "parent properties": properties that are weaker than a given property. By tracing these tables Maple finds that both `RealRange(-infinity,Open(0))` and `RealRange(Open(0),infinity))` have `real` as a so-called ancestor, so it finds that `t` is a real number.

A survey of properties can be found by asking help for `property`. Moreover, a user can extend this system by other simple or parametric properties, using `addproperty`.

Information about properties can also be found with `about`. For instance, on `fraction`:

```
> about(fraction);
```

```
fraction:
  noninteger rational
  a known property having {rational, Non(0)} as immediate parents
  and {BottomProp} as immediate children.
  mutually exclusive with {0, 1, integer, prime, irrational,
    composite}
```

The first line explains the property. The next lines reveal the direct relations to other properties: its immediate parents (the next weaker properties), its children (the next stronger properties), and the next properties that are incompatible with it.

In this case, the property is also a type, so we can ask on-line help with `?type,fraction` as well. This explanation is more extensive than the one produced by `about`.

A.9 Properties and assigning

Assuming a property for a name is like assigning an unidentified object with the given property to that name. This can cause some strange effects. That is shown in the next examples:

```
> assume(t,positive);
> T:=t;
```

$$T := t\sim$$

Now T refers to a UPO, an Unidentified Positive number Object, indicated as $t\sim$. Observe the result when we give another property to `t`:

```
> assume(t,negative);
> about(t);
Originally t, renamed t~:
  is assumed to be: RealRange(-infinity,Open(0))
```

By this new assumption, the old property of `t` has been discarded, so `t` refers now to an Unidentified Negative number Object, again printed as $t\sim$. But the name T has obtained its value *before* the second assumption for `t`, so it still refers to that UPO:

```
> about(T);
Originally t, renamed t~:
  is assumed to be: RealRange(Open(0),infinity)
```

So at this moment T and `t` refer to different unidentified objects, both of which are indicated in the same way as $t\sim$.

Still more embarrassing is the result when we again assume t to be positive:

```
> assume(t,positive);
```

Now `t` refers to a *new* Unidentified Positive number Object, different from the one that it referred to previously. This explains the following result:

```
> T-t;
```

$$t\sim -t\sim$$

Advice: when you want to assign an object with a property to a name, use single quotes; for instance:

```
> T := ' t-3 '; is( T>7) ;
```

$$T := t - 3$$

$$false$$

```
> assume( t>10 ); is( T>7 );
```

$$true$$

Because of the single quotes, `T` does not refer to the evaluation result of `t-3` but to this expression itself; when something is changed in the properties of `t`, this change has an effect for `T` as well.

A.10 Properties and formal parameters

Assumed variables may not work as formal parameters. For instance:

```
> assume(n,integer);
> y:=n^2-1;
```

$$y := n\sim^2 -1$$

```
> seq(y,n=1..5);
```

$$n\sim^2 -1,\ n\sim^2 -1,\ n\sim^2 -1,\ n\sim^2 -1,\ n\sim^2 -1$$

Again the trick with back quotes works:

```
> y:='n^2-1';
```

$$y := n^2 - 1$$

```
> seq(y,n=1..5);
```

$$0,\ 3,\ 8,\ 15,\ 24$$

A.11 Domains, the Domains package

The Domains package is designed for calculations in rings, derived from the integers or the rationals by some standard steps, for instance taking the ring of polynomials over an already existing ring or the ring of n by n matrices over such a ring.

The Domains package creates representation methods, ring operations, and algorithms such as the Euclidean algorithm for greatest common divisor, if applicable, in an automatic and standardized way. The concept of this package originates from the symbolic computation system Axiom.
Here is an example:

```
> with(Domains);
---------------------- Domains version 1.0 ----------------------
Initially defined domains are Z and Q the integers and rationals
Abbreviations, e.g. DUP for DenseUnivariatePolynomial, also made
```

$$[init]$$

First, we create the ring of polynomials in X with rational coefficients:

```
> QX := DUP( Q , X ):
```

The domain QX was created with the Domains-procedure DUP (=DenseUnivariatePolynomial), using the field Q, which is known to Domains. Now we can use the domain QX. In fact, QX is a table of procedures. This table contains an input procedure QX[Input]:

```
> p := QX[Input] (4 - 7*X^2 + 5*X^3);
```

$$p := 4 - 7X^2 + 5X^3$$

The name p refers to a special internal structure representing the given polynomial of the ring represented by QX; this object can be handled by other procedures contained in the table QX. For instance, we can calculate p^2 and make $q := X^{10} + p^2$:

```
> QX['^'] (p , 2);
```

$$16 - 56X^2 + 40X^3 + 49X^4 - 70X^5 + 25X^6$$

```
> q := QX['+'] ( QX[Input](X^10) , % );
```

$$X^{10} + 16 - 56X^2 + 40X^3 + 49X^4 - 70X^5 + 25X^6$$

Let's calculate the remainder of the division of q by p:

```
> r := QX[Rem] (q , p);
```

$$\frac{667828}{390625} + \frac{38204}{78125}X - \frac{1379399}{390625}X^2$$

To see a survey of the available procedures in the table QX, enter `show (QX,operations)`.

To make things more efficient, we can define infix operators for QX. That is possible by using names starting with an **ampersand** (&). For instance,

```
> '&QXpr' := QX['*'];
```

$$\text{`\&QXpr`} := QX[\text{`*`}]$$

As the name &QXpr starts with an ampersand, we must surround it with back quotes to make Maple accept it as a name. It can be used without back quotes as an infix operator:

```
> QX[Output](p &QXpr q) ;
```

$$5X^{13} - 7X^{12} + 64 + 4X^{10} + 125X^9 - 525X^8 + 735X^7 - 43X^6 - 840X^5 + 588X^4 + 240X^3 - 336X^2$$

There are several other procedures in Domains for the creation of rings. For instance, we can create the quotient field of Q[X] by applying the Domains-command QF on the ring QX created earlier: `QF(QX)`:

```
> R := QF(QX):
> p1 := R[Input] (4 - 7*X^2 + 5*X^3);
```

$$p1 := 4 - 7X^2 + 5X^3$$

```
> R['/'](%,p1);
```

$$1$$

```
> R['/'](p1,R[Input](X^2-8));
```

$$\frac{4 - 7X^2 + 5X^3}{X^2 - 8}$$

The results are printed as the usual polynomials, but internally they have a quite different structure. The procedure `lprint` can reveal that:

```
> lprint(q);
'domains/DenseUnivariatePolynomial/badge0'(16,0,-56,40,49,-70,25,0,
   0,0,1)
```

The package Domains is available from release 4. In earlier versions of Domains (called Gauss), results are not printed in easily readable form automatically, but only with the aid of output procedures, for instance:

```
> QX[Output] (q);
```

$$X^{10} + 16 - 56X^2 + 40X^3 + 49X^4 - 70X^5 + 25X^6$$

For more information, consult the on-line help for Domains and `Domains, example`. For the special case of Galois fields, consult the on-line help to the GF package.

appendix B

Names and evaluation 3: some special features

In Chapters 1, 3, and 5, using names in Maple is discussed. This appendix is a supplement to this subject, discussing the `alias` *facility and several aspects of names and evaluation.*

B.1 Changing names, `alias`

If you want to use another name for an existing Maple object, you can assign this object to that other name. For instance, we can assign the procedure `BesselI` to the name `BI`. Let's use `eval` in order to make `BI` refer directly to that procedure.

```
> BI := eval(BesselI);
```

$$\mathbf{proc}(v :: algebraic, x :: algebraic) \ \ldots \ \mathbf{end}$$

Now we can use `BI` instead of `BesselI` in the input, but if Maple prints a result containing this object, it uses the standard Maple name:

```
> diff(BI(5,x),x);
```

$$\mathrm{BesselI}(4,x) - 5\,\frac{\mathrm{BesselI}(5,x)}{x}$$

To have Maple use `BI` in the output as well, apply the procedure **alias**:

```
> BI:='BI': alias( BI=BesselI );
```

$$I,\ BI$$

```
> diff(BI(5,x),x);
```

$$\mathrm{BI}(4,x) - 5\,\frac{\mathrm{BI}(5,x)}{x}$$

The `alias` facility is quite different from assigning; it only tells Maple to translate input before processing it and to translate output before printing it. The `alias` command works according to the syntax `alias(<username>=<object in Maple denotation, without the use of other aliases>)`. The result of an alias command is a sequence of all names currently aliased to Maple objects. Unassigning `BI` before the `alias` command is not necessary here, but generally an alias must be made only to unassigned names.

Entering the command `alias(BI(n,x) = BesselI(n,x))` would not be efficient; this would fail in the present example, as Maple would interpret `n` as the name `n`, not as a variable that could be 5, for instance.

Another example: we want to use the symbol `j` instead of `I` for $\sqrt{-1}$. You might think that it can be done in the following way:

```
> alias(j=I);
```

$$j,j$$

The result of this command is curious, but more so the effect:

```
> j^2 , sqrt(-1);
```

$$j^2, I$$

The reason is that it is not allowed to use an alias in the right-hand side of an alias declaration; `I` is an alias for $\sqrt{-1}$, so the command `alias(j=I)` is against the rules. Instead, you must unalias `I` with the command `alias(I=I)` first, and then alias `j` to $\sqrt{-1}$. Both can be combined in one command:

```
> alias(I=I,j=sqrt(-1));
```

$$BI, j$$

```
> expand((a+b*j)*(c+d*j));
```

$$ac + jad + jbc - bd$$

Now let's reset the aliases:

```
> alias(BI=BI,j=j,I=sqrt(-1));
```

$$I$$

For more examples of using `alias`, see Chapter 6, *Creating and using mathematical functions*, and Chapter 14, *Polynomial equations and factoring polynomials*.

Another way of using your own names is the procedure **macro**. This works only for translating input. It is even possible to use a macro for a call such as:

```
> macro( whoops =
>     convert( series( % , x=0 , 10 ) , polynom ) );
> sin(x) : whoops;
```

$$x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \frac{1}{362880}x^9$$

B.2 Finding names used

It is possible to obtain a sequence of all names that are assigned at that moment with the procedure **anames**:

```
> demovar:=13:
> anames():
```

The result that would be printed if the command was terminated with a semicolon contains a lot of uninteresting names, much more than that one name `demovar` that has been assigned to in the present session: Maple has read in procedures from its library and these names are presented here as well. Moreover, many tables have been made. It is, however, possible to obtain all names not referring to a procedure or a table with the aid of `remove` as follows (release V.5):

```
> remove(x->whattype(eval(x,2))=symbol,[anames()]);
```

$$[demovar]$$

For release V.4 use

```
> remove( x->whattype(eval(x,2))=string,[anames()]);
```

$$[demovar]$$

For release V.3, use

```
> select( x->whattype(eval(x)) <> procedure,[anames()] );
```

$$[demovar]$$

It is important that this command is not split into two commands, otherwise evaluation throws a spanner in the works.

If you wish to have this command easily available, make a macro of it and include this in your initialization file.

It is also possible to obtain the assigned names in **lexical order** with:

```
> sort([anames()],lexorder):
```

(The output has been suppressed.)

B.3 Indexed names

An element of a sequence, a vector, a matrix, or, more generally, an array or a table, can be selected by indexing:

```
> vc := linalg[vector]([1,2,3,5,7,11,13,17,19,23,29,31]);
```

$$vc := [1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11 \quad 13 \quad 17 \quad 19 \quad 23 \quad 29 \quad 31]$$

```
> vc[7];
```

Indexes are always given between square brackets. An index to an unassigned name is accepted with the idea that this name might refer to something else in the future:

```
> xyz[3];
```

$$xyz_3$$

This last object is called an **indexed name.** At this moment `xyz` is still an unassigned name:

```
> eval(xyz);
```

$$xyz$$

But as soon as something is assigned to `xyz[3]`, the name `xyz` refers to a table:

```
> xyz[3]:=0: eval(xyz);
```

table([

$3 = 0$

])

B.4 Quotes with table, arrays, vectors, and matrices

Suppose we want to unassign the last five elements of the vector `vc`. For that purpose, we can use the repetitional control structure. See section E.5 on page 302. Obviously, we cannot apply the command "`for i from 8 to 12 do vc[i]:='vc[i]' od;`", as the last `i` would not be evaluated. For such purposes the procedure **evaln** can be used instead of the **forward quotes**:

```
> for i from 8 to 12 do vc[i] := evaln(vc[i]) od:
> print(vc);
```

$$\left[1 \quad 2 \quad 3 \quad 5 \quad 7 \quad 11 \quad 13 \quad vc_8 \quad vc_9 \quad vc_{10} \quad vc_{11} \quad vc_{12}\right]$$

Using **back quotes** with tables, etc. can also be a source of mistakes. For instance,

```
> `some.thing`[1] := 2;
```

$$`some.thing`[1] := 2$$

```
> `other.thing[1]` := 3;
```

$$`other.thing[1]` := 3$$

In the first case, only the name is written between back quotes; in the second case, the index is included. Look at the difference:

```
> eval('some.thing');
```

$$table([\ 1 = 2\ ])$$

```
> eval('other.thing');
```

$$`other.thing`$$

Notice that the name `some.thing` refers to a table, but the name `other.thing` is unassigned; we have assigned 3 to the name `other.thing[1]`, which is a symbol, not an indexed name, because the whole is enclosed in back quotes.

B.5 Recovering lost procedures

If you might have lost a procedure from the library, it is possible to get it back, unless it is a so-called internal procedure. To see the list of internal procedures, enter `?index,internal`. In other cases, if the procedure is contained in the standard library, use `readlib` and assign the procedure to the name. For instance, in the following, we lose the standard procedure `norm`:

```
> with(linalg,norm):
Warning, new definition for norm
```

$$[norm]$$

We can recover this procedure and assign it to a name by:

```
> polynorm := readlib(norm);
```

$$polynorm := \mathbf{proc}(p,\ n,\ v)\ \ldots\ \mathbf{end}$$

If the procedure is contained in one of the packages, it can be recovered in the same way as it is obtained in other cases with the procedure `with`.

B.6 Exceptions to the rule of automatic full evaluation

If an expression enclosed between a pair of forward quotes is evaluated, no values of names are looked up by Maple. Instead, that pair of quotes is peeled of.

Obviously, left-hand sides of assignments are not evaluated.

In almost all cases, the arguments to a procedure are evaluated before the procedure comes into action. There are a few exceptions to this rule: `evaln`, `assigned`, `traperror`, `parse`, and `addressof`. These procedures take arguments literally. Moreover, the first argument in `eval` and the parameter in `seq` is taken literally, without evaluation.

For other procedures, evaluating arguments can be precluded with the aid of forward quotes, but you must not use forward quotes for arguments taken literally by the procedures mentioned in the above.

Names of procedures are evaluated to the last name, as explained in section 6.4 on page 75. However, if the procedure is applied to an argument, it is evaluated, unless this is prevented by another cause, mentioned in this section.

Names of tables, arrays, matrices, and vectors are evaluated to the last name, as explained in Chapter 18, *Vectors and matrices*. If such a name gets an index, it is evaluated, unless this is prevented by another cause, mentioned in this section.

Local variables are specific for programming procedures; these are evaluated only one step automatically within the procedure. See section E.5 on page 302.

appendix C

The user interface for text-only versions

Although the user interface for windowing systems offers many facilities, the text-only interface is attractive on its own: it is fast and efficient and uses much less memory. This appendix discusses general aspects of this user interface.

C.1 Starting, interrupting, and quitting Maple

You can start text-only Maple by clicking on `Command Line Maple` from Windows, but you can save more memory by starting from MS-DOS in the directory `bin\wnt` with the command `cmaple` if Maple has been installed under Windows 95 or Windows NT, else from the directory `win\win` with the command `dosmaple`.

On the Mac, click on the icon `Command Line Maple`.

Under Unix, give the command `maple`.

When you start Maple with a command, you can add options. Several of the effects of these options can also be produced with `interface`; moreover, this procedure offers many other possibilities for changing the user interface, etc. Details can be found in the on-line help for `maple`, `interface`, and `kernelopts`.

You can try to interrupt a process by pressing Control-C, but it may take some time before this is intercepted.

You can end the Maple session with the command `quit`, `stop`, or `done`.

C.2 Editing commands

A very nice feature of this interface is that a history of commands is available for reusing. To recall the previous command, press the up-arrow key, press again for the command before, etc. You can then change the command in the usual way. (The number of commands that is remembered depends on their lengths.)

If a command is entered without a colon or semicolon, you get a warning (from release V.4), and you can enter it on the next line. If you don't mind getting warned, you can enter a command on more than one line.

After a syntax error, Maple prints `syntax error:  ...` and indicates with a ^ the offending place. You have to enter the command again, but you may use the history facility.

C.3 Pictures

On text-only systems you still can make plots, even 3-dimensional, made by characters. But you can always export pictures to a file and view them with other software. Several formats are available. For instance, if you want to create a jpeg file, you can enter

```
> plotsetup(jpeg);
Warning, plotoutput file set to plot.jpg
```

Maple informs you that from now on the pictures will not be printed to the screen but to a file named `plot.jpg`. If you want to print more than one file, each time you may indicate a new name for the file, for instance:

```
> plotsetup(jpeg,plotoutput='figure7.jpg');
```

If you want to see pictures on the screen again, enter

```
> plotsetup(default);
```

In release V.3 the last command should be `plotsetup(terminal)`.

C.4 Maple system messages

In most Maple versions you can see the amount of workspace in memory used by Maple, and the computing time from the start of the session at the bottom of the screen. Only in character-based, full scrolling, terminal versions, messages on the state are printed on the screen *between* parts of the session. If you want to suppress such messages, enter `gc(0)` in release V.3 or `kernelopts(printbytes=false)` in later releases.

C.5 Saving a session and its results

If you intend to continue your Maple work later on and want to exit Maple or to perform other operations that might disturb your present results, you can preserve the present state of your Maple session in a file, say `'seeyou.m'`:

```
> save 'seeyou.m';
```

The extension `.m` causes fast and efficient saving in a special Maple format. Omitting this extension yields a readable file.
If you start Maple later on, you can enter

```
> read 'seeyou.m';
```

All variables at the moment of saving in the previous session are restored.

appendix D

Procedures remembering previous results

The remember tables are an essential aspect of the functioning of Maple procedures. Effects of remember tables are shown in Chapter 8 (`series`*), Chapter 14 (*`expand`*), and Chapter 15 (*`solve`*). This appendix shows how to handle such remember tables.*

D.1 Remember tables of procedures

For each procedure, individual results can be stored. For instance,

```
> sin(1) := h;
```

$$\sin(1) := h$$

```
> sin(5*Pi+1);
```

$$-h$$

We can even create a procedure in this way:

```
> f(3) := 5000;
```

$$\mathrm{f}(3) := 5000$$

```
> eval(f);
```

proc() **option** *remember*; 'procname(args)' **end**

This cryptic description of `f` means that the procedure `f` stores each "calculated" result in its remember table, and that the result of applying f is the function name f applied to the given arguments, except for the case(s) that the argument is contained in its remember table. This remember table can be obtained as follows:

```
> op( 4 , eval(f) );
```

table([3 = 5000
])

Another possibility (from release V.4) is as follows:

```
> interface(verboseproc=3)}:
> print(f):
```

proc() **option** *remember*; 'procname(args)' **end** # (3) = 5000

Let's calculate some function values of `f`:

```
> f(1),f(t),f(3),f(Pi);
```

$$\mathrm{f}(1),\ \mathrm{f}(t),\ 5000,\ \mathrm{f}(\pi)$$

The procedure returns the unevaluated function call, unless the argument is 3, in which case it yields 5,000.

For many procedures, the remember table is essential. For instance, here is the remember table of `sin`:

```
> op(4,eval(sin));
```

table([

$$\pi = 0,$$
$$\frac{1}{6}\pi = \frac{1}{2},$$
$$\frac{3}{10}\pi = \frac{1}{4}\sqrt{5} + \frac{1}{4},$$
$$\frac{1}{8}\pi = \frac{1}{2}\sqrt{2 - \sqrt{2}},$$
$$\frac{3}{8}\pi = \frac{1}{2}\sqrt{2 + \sqrt{2}},$$
$$\frac{1}{3}\pi = \frac{1}{2}\sqrt{3},$$
$$\frac{1}{4}\pi = \frac{1}{2}\sqrt{2},$$
$$0 = 0,$$
$$1 = h,$$
$$\frac{1}{2}\pi = 1,$$
$$\frac{1}{5}\pi = \frac{1}{4}\sqrt{2}\sqrt{5 - \sqrt{5}},$$
$$I = I\ \sinh(1),$$
$$\frac{1}{12}\pi = \frac{1}{4}\sqrt{6}\left(1 - \frac{1}{3}\sqrt{3}\right),$$
$$\frac{1}{10}\pi = \frac{1}{4}\sqrt{5} - \frac{1}{4},$$
$$\frac{5}{12}\pi = \frac{1}{4}\sqrt{6}\left(1 + \frac{1}{3}\sqrt{3}\right),$$
$$\frac{2}{5}\pi = \frac{1}{4}\sqrt{2}\sqrt{5 + \sqrt{5}}$$

])

In the table you can find the addition made at the start of this chapter: $\sin(1) = h$.

D.2 Clearing (parts of) the remember table

An item of a remember table can be removed with **forget**, which must be read from the library before it is used:

```
> readlib(forget);
```

$$\textbf{proc}(f)\ \ldots\ \textbf{end}$$

To have Maple forget that $f(3)$ should be 5, enter:

```
> forget(f,3);
> f(3);
```

$$\mathrm{f}(3)$$

If no arguments are specified, his command removes all additions to the remember table, resetting it to the original state:

```
> forget(sin);
> sin(1),sin(Pi);
```

$$\sin(1),\ 0$$

Some procedures store all results automatically. If a procedure is called, it looks up its present arguments in its remember table first; if it has calculated the same thing before, it uses the result from the table instead of calculating it.

If a procedure calls another procedure, it is possible that results are stored in the memory table of that subprocedure. For instance, if Maple encounters the command `int(sin(x),x)`, then it translates the parameter `x` into `_X` first, it passes the translation on to a subprocedure `'int/indef'`, which tries to calculate the integral and stores the result *in the memory table of that subprocedure*, and, at last, that result is translated back by the main procedure.

Although this calculation of an integral makes Maple store a result, it is not always easy to find which subprocedure has stored it. However, the procedure `forget` resets the remember tables of the specific subprocedures as well.

D.3 An example of side effects of the remember table: `infolevel`

The memory tables of procedures are essential for the efficiency of some of them, especially where procedures are programmed recursively. But sometimes, memory tables can have unexpected effects. Here is an example:

```
> int(exp(sin(x)),x);
```

$$\int e^{\sin(x)}\,dx$$

If we want to know what Maple has tried to do, it does not suffice to set **infolevel**:

```
> infolevel[int]:=1;
```

$$infolevel_{int} := 1$$

```
> int(exp(sin(x)),x);
```

$$\int e^{\sin(x)}\,dx$$

We don't get information about the calculation, because Maple did not calculate anything at all: it found an item "`exp(sin(x))=FAIL`" in the memory table of '`int/indef`' and immediately returned the call to `int` unevaluated. But when the memory table of `int` and its subprocedures is reset, we get the desired information:

```
> forget(int);
> int(exp(sin(x)),x);
int/indef:   first-stage indefinite integration
int/indef2:   second-stage indefinite integration
int/trigexp:   case of integrand containing exp and trigs
int/rischnorm:   enter Risch-Norman integrator
int/risch:   enter Risch integration
int/risch/algebraic1:   RootOfs should be algebraic numbers and
functions
int/indef:   first-stage indefinite integration
int/indef2:   second-stage indefinite integration
int/indef2:   applying derivative-divides
int/indef:   first-stage indefinite integration
int/indef:   first-stage indefinite integration
int/indef2:   second-stage indefinite integration
int/exp:   case of integrand containing exp
int/indef:   first-stage indefinite integration
int/indef2:   second-stage indefinite integration
int/exp:   case of integrand containing exp
int/prpexp:   case ratpoly*exp(arg)
int/risch:   exit Risch integration
```

$$\int e^{\sin(x)}\,dx$$

appendix E

Control structures

An extensive guide to programming in Maple is beyond the scope of this book. But sometimes it can be convenient to automate a task in order to prevent typing too much. For these cases, the present appendix can give you some support.

In some versions of Maple V.5 you can use a spreadsheet for automated execution of a system of commands. Another way of automating tasks in Maple is to write the commands in a file, check this file with `mint` *from outside of Maple (read your manual or the on-line help about that wonderful tool* `mint`*), and then read the file from within Maple with the command* `read`.

A more elegant way is to write procedures containing the desired commands. This has some advantages, especially the possibility of using arguments. Again, it is very much advisable to write such a procedure into a file, then check it with `mint` *and, after having fixed the possible syntax errors, read the file from within Maple with* `read`.

This appendix discusses the elementary use of procedures, choices, and repetitions. At the end, an extensive example is shown, concerning checking solutions of a set of polynomial equations.

If you want to know more on programming in Maple, you can read Maple V Programming Guide *by* Monagan, Geddes, Heal, Labahn, and Vorkoetter.

E.1 Procedures

In section 6.4 on page 75, you can read how mathematical functions can be created in Maple with the aid of the arrow. In fact, these functions are procedures, meant to be printed in the mathematical style. For instance,

```
> f := x -> x^2 - 1;
```

$$f := x \rightarrow x^2 - 1$$

This is really a procedure:

```
> lprint(eval(f));
```

proc(x) **options** *operator*, *arrow*; $x^2 - 1$ **end**

We could have defined `f` as well with the aid of the procedure **proc**:

```
> f := proc(x) x^2-1 end;
```

$$\mathbf{proc}(x)\ x^2 - 1\ \mathbf{end}$$

This last `f` calculates the same results as the previous one, but it is not printed in the mathematical style.

The construction with `proc` is more flexible than the construction with the arrow. An important advantage is the possibility of including more than one command. As a basic example, here is a procedure that calculates a series expansion and then converts the result into a polynomial.

```
> polseries:=proc( y , pos  , order )
>    series( y , pos , order );
>    convert( % , polynom );
>    end;
```

$$polseries := \mathbf{proc}(y, pos, order)$$
$$\quad \text{series}(y, pos, order); \text{convert}(\%, polynom)$$
$$\mathbf{end}$$

```
> polseries( cos(x) , x=Pi , 5 );
```

$$-1 + \frac{1}{2}(x-\pi)^2 - \frac{1}{24}(x-\pi)^4$$

```
> polseries( exp(y) , y=1 , 4 );
```

$$\mathbf{e} + \mathbf{e}(y-1) + \frac{1}{2}\mathbf{e}(y-1)^2 + \frac{1}{6}\mathbf{e}(y-1)^3$$

The last command shows that the y in the argument `exp(y)` does not interfere with the parameter y in the definition of `polseries`.

Procedure parameters should not be used as if they are variables that can obtain another value: assignments can cause strange effects. The value of such a procedure parameter is known only to the procedure itself.

The result of executing a procedure is the result of the last executed command. For instance, if the last executed command was something like `print( )`, then there would be no result that could be referred to by the ditto or used by another procedure, though something would be printed to the screen.

Observe that the intermediate results are not printed by Maple. That can be changed by raising the value of `printlevel`. See section E.5 on page 302.

E.2 Searching for causes of odd behavior with `trace` or `printlevel`

If a procedure yields an unexpected error message, first, check that the arguments given to that procedure are correct by evaluating them. If these are not suspect, and the `Error` message mentions problems with a specific procedure, try applying `trace` to the offending procedure. This causes Maple to print input and output for each call of this procedure, with intermediate results calculated by that procedure. There are several other tools for debugging available; if necessary, look at the on-line help for `debug`.

E.3 Using `if ... fi` for choices

Here is an example of the choice structure: we want to create a function f with $f(0) = a$, $f(1) = b$, $f(2) = c$, and $f(x) = 0$ for $x \neq 0, 1, 2$. We can write this procedure in a file:

```
f := proc(x)
    if x=0 then a
      elif x=1 then b
      elif x=2 then c
      else 0
    fi
end:
```

Let's suppose that this file is called "`demo1`". Now we can use **mint** to check this file for correct syntax by entering `mint demo1`. This command should not be given from Maple, but from the operating system of the computer. The result is:

```
Procedure  f( x ) on lines 1 to 7
  These names were used as global names:   a, b, c
```

There are no reports on syntax errors, nor cues that something might not be according to our plans, so we can start Maple and give the command

```
> read demo1;
```

Now `f` is available:

```
> f(-1), f(0), f(1), f(2), f(3);
```

$$0, a, b, c, 0$$

The last command executed after `then` or `else` yields the result of the `if ... fi` structure. This is the last executed command of the procedure as well, so it yields the result of the procedure.

After each occurrence of `then` and after `else`, several commands can be given, each closed by a colon or semicolon apart from the last one. There is no objection to using more (semi)colons, as in:

```
f := proc(x);
    if x=0 then a;
      elif x=1 then b;
      elif x=2 then c;
      else 0;
    fi;
end:
```

The present example shows a choice from several possibilities. If your choice is more restricted, you can omit `elif ...` and even omit `else ...`. But in any case, the choice structure must be closed by the word `fi`, otherwise you get a syntax error.

The procedure in the previous section requires an argument `order`. The standard procedure `series` can do without: if that order argument is omitted, it uses the value of the name `Order`. Such a functionality can also be used for modifying the procedure `polseries` created in the first section:

```
polseries:=proc( y , pos )
   if nargs>2 then series( y , pos , args[3]+1 )
     else series( y , pos ) fi;
   convert( % , polynom );
   end;
```

Here **nargs** and **args** are used; within a procedure, `nargs` is interpreted as the number of arguments in the present call. Moreover, `args` is interpreted as the actual sequence of arguments.
If we read in this procedure, we can use it, as in:

```
> polseries( ln(x) , x=Pi , 3 );
```

$$\ln(\pi) + \frac{x-\pi}{\pi} - \frac{1}{2}\frac{(x-\pi)^2}{\pi^2} + \frac{1}{3}\frac{(x-\pi)^3}{\pi^3}$$

```
> polseries( ln(x) , x=Pi );
```

$$\ln(\pi) + \frac{x-\pi}{\pi} - \frac{1}{2}\frac{(x-\pi)^2}{\pi^2} + \frac{1}{3}\frac{(x-\pi)^3}{\pi^3} - \frac{1}{4}\frac{(x-\pi)^4}{\pi^4} + \frac{1}{5}\frac{(x-\pi)^5}{\pi^5}$$

E.4 Recursion

Many mathematical objects are defined recursively. Such a definition can be used in a natural way for calculations. In Maple, $n!$ is available, but here is a simple procedure for it by way of an example:

```
fac := proc(n)
   if n=0 then 1
     elif type(n,posint) then n*fac(n-1)
     else 'procname(args)'
     fi
   end;
```

After having checked this with `mint` and having read it in Maple, you can use it. For instance,

```
> fac(7), fac(0), fac(-sqrt(3)), fac(t);
```

$$5040,\ 1,\ \mathrm{fac}\left(-\sqrt{3}\right),\ \mathrm{fac}(t)$$

A less trivial example is calculating Hermite polynomials. The `orthopoly` package contains a procedure for this purpose. The Hermite polynomials are defined by:

$$\mathrm{H}(n,x) := \begin{cases} 1 & \text{if } n = 0 \\ 2x & \text{if } n = 1 \\ 2x\,\mathrm{H}(n-1,x) - 2(n-1)\,\mathrm{H}(n-2,x) & \text{otherwise} \end{cases}$$

This can be programmed as follows:

```
H := proc(n::integer,x)
    if n>1 then 2*x*H(n-1,x)-2*(n-1)*H(n-2,x)
      elif n=1 then 2*x
      else 1
      fi
  end;
```

The first argument `n` is indicated as an integer by `n::integer`; the procedure will test if it is an integer automatically.

If this text is read in by Maple, the procedure `H` can be used, for instance,

```
> H(2,x); H(3,x); H(4,t);
```

$$4x^2 - 2$$

$$2x\left(4x^2 - 2\right) - 8x$$

$$2t\left(2t\left(4t^2 - 2\right) - 8t\right) - 24t^2 + 12$$

This seems to be fine. However, there are efficiency problems: if you ask for `H(5,x)`, Maple has to calculate H(4,x) and H(3,x). For the calculation of `H(4,x)`, Maple has to calculate H(3,x) a second time. For instance, if you ask for `H(10,x)`, then `H(3,x)` is calculated 21 times. We can avoid a procedure to calculate things more than once by adding the option `remember`. Then each result of the procedure is stored in its remember table automatically. See Appendix D, *Procedures remembering previous results*. In fact, the standard procedure in Maple that calculates Hermite polynomials uses this option `remember`. Here is the source:

```
'orthopoly/H' := proc(n,x)
option remember,
'Copyright (c) 1991 by the University of Waterloo.
All rights reserved.';
    if 1 < n then
        expand(2*x*'orthopoly/H'(n-1,x)
        -2*(n-1)*'orthopoly/H'(n-2,x))
    elif n = 1 then 2*x
    else 1
    fi
end:
```

Now the procedure stores each Hermite polynomial that it has calculated in its memory table. When the procedure is called, it looks first to see if the required Hermite polynomial is available in its memory table.

In fact, when you call `orthopoly[H]` for calculating a Hermite polynomial, Maple does more: it checks the first arguments (it should be an integer ≥ 0), then it calls the previous procedure with a second argument `_X`, and, at the end, it translates that `_X` into the second argument given by the user.

E.5 Using do ... od for repeating actions

The previous section shows an elegant method for repetition. There is also a special structure for this purpose, already shown partially in section 18.17 on page 267 and in section B.3 on page 287.
Here is an example: let's suppose that we want to factor $x^3 - 1$, $x^5 - 1$, $x^7 - 1$, $x^9 - 1$, and $x^{11} - 1$.

```
> for i from 3 by 2 to 11 do factor(x^i-1) od;
```

$$(x-1)\left(x^2+x+1\right)$$

$$(x-1)\left(x^4+x^3+x^2+x+1\right)$$

$$(x-1)\left(x^6+x^5+x^4+x^3+x^2+x+1\right)$$

$$(x-1)\left(x^2+x+1\right)\left(x^6+x^3+1\right)$$

$$(x-1)\left(x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1\right)$$

Maple can use defaults if not all elements are given. For instance, the same result is obtained with the following command:

```
> for i to 5 do factor( x^(2*i+1) - 1 ) od:
```

Here `i` takes the values 1, 2, 3, 4, and 5.

In many cases, such a repetition can also be generated efficiently by applying the procedure `seq`. See section 10.7 on page 132.

The previous section gives a procedure for calculating Hermite polynomials by recursion. The present construction can be used as well as follows:

```
H := proc(n::integer,x)
local h,i;
    h[0] := 1;
    h[1] := 2*x;
    for i from 2 to n do
      2*x*h[i-1] - 2*(i-1)*h[i-2];
      h[i] :=  expand( % )
      od;
    h[n]
end:
```

The second line declares the names `h` and `i` to be local variables. If a name is a local variable within a procedure, the values it gets in the execution of that procedure play no role outside that execution, so it cannot interfere with the session or with execution of other procedures. If we had not declared these as local, Maple would have done so automatically, but at the same time it would have issued a warning. It is also possible to declare a name as global. If you want to use local variables, you are advised to read more on this subject, for instance, on the special one-step evaluation rule, in the on-line help, or the book mentioned in the introduction of this appendix.

In the procedure, a table `h` is used. This is created automatically in the first assignment `h[0]:=1`. A table resembles a vector, but it has no restrictions in the type and the number of indices.

If you execute a repetition within a repetition interactively, results of the inner repetition are not usually shown on the screen:

```
> for i to 2 do
>   for j to 3 do 10*i+j od;
>   %*100
>   od;
```

$$1300$$

$$2300$$

This can be changed by the value of **printlevel**:

```
> printlevel := 2:
> for i to 2 do
>   for j to 3 do 10*i+j od;
>   %*100
>   od;
```

11

12

13

1300

21

22

23

2300

Generally, nesting choice and repetition structures prevents output of intermediate results unless the value of `printlevel` is high enough. Output generated by the procedure `print` is not influenced by this mechanism, but output from **plot** and **plot3d** is. Therefore, if you want to plot from inside nested choices and loops, you can better use `print(plot( ))` instead of `plot( )`.

If it is not clear in advance how many repetitions are necessary, use

`while do od`

For instance, if you try to find the smallest n such that factorization of $x^n - 1$ yields 10 or more factors, you can enter:

```
> n:=3: while nops(factor(x^n-1))<10 do n:=n+1 od:
> n;
```

48

The full repetition structure is:

`for from by to while do od`

where only the `do od` part is mandatory. There is a variant of this structure, demonstrated in the next section:

`for in while do od`

E.6 An example: checking the results of `solve` by substituting

This section shows how candidate solutions for a system of polynomial equations can be checked in an automated way. The system of equations in the example is chosen to be basic enough to be solved by hand, in order to help you understand the results.

```
> eq1 := 3*x^2 + 2*y^2 =  7*x*y;
```

$$eq1 := 3x^2 + 2y^2 = 7xy$$

```
> eq2 := x^2 + y^2+2*x = 20;
```

$$eq2 := x^2 + y^2 + 2x = 20$$

```
> sol := solve( {eq1,eq2} , {x,y} );
```

$$sol := \Big\{ x = \mathrm{RootOf}\big(5\_Z^2 + \_Z - 10\big),\ y = 3\ \mathrm{RootOf}\big(5\_Z^2 + \_Z - 10\big) \Big\},\ \Big\{ y = \frac{1}{2}\ \mathrm{RootOf}\big(5\_Z^2 + 8\_Z - 80\big),\ x = \mathrm{RootOf}\big(5\_Z^2 + 8\_Z - 80\big) \Big\}$$

Maple's solution consists of a sequence of two sets. It is easy to check these, for instance the first one:

```
> subs(%[1],{eq1,eq2});
```

$$\{10\,\%1^2 + 2\,\%1 = 20,\ 21\,\%1^2 = 21\,\%1^2\}$$

$$\%1 = \mathrm{RootOf}\big(5\,Z^2 + Z - 10\big)$$

```
>  map( eq->simplify(lhs(eq)-rhs(eq)),%);
```

$$\{0\}$$

This affirms the first solution. The second can be handled in the same way.

In fact, each of these two elements of *sol* represents some explicit solutions. We can find these with `allvalues`. See section 14.2 on page 171. We could use this procedure without option (or in release V.3 with the option `'d'`), so that all occurrences of the same `RootOf` are interpreted as the same number, after having argued that we would find all solutions that way. But suppose that we don't trust this idea. Then we can use the option `'independent'` and check all candidate solutions found.

If we try to apply `allvalues` directly to *sol*, we get a syntax error: it is a sequence of two objects, which yields two arguments for `allvalues`. So we must bundle them together, in a set (or a list):

```
> allvalues( {sol} , 'independent' );
```

$$\left\{ \left\{ x = -\frac{4}{5} + \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} + \frac{2}{5}\sqrt{26} \right\},\right.$$
$$\left\{ y = -\frac{3}{10} + \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} + \frac{1}{10}\sqrt{201} \right\} \Big\},$$
$$\left\{ \left\{ x = -\frac{4}{5} - \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} + \frac{2}{5}\sqrt{26} \right\},\right.$$
$$\left\{ y = -\frac{3}{10} + \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} + \frac{1}{10}\sqrt{201} \right\} \Big\},$$

... omitting a part of the output ...

$$\left\{ \left\{ x = -\frac{4}{5} - \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} - \frac{2}{5}\sqrt{26} \right\},\right.$$
$$\left\{ y = -\frac{3}{10} - \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} - \frac{1}{10}\sqrt{201} \right\} \Big\}$$

We have obtained a sequence consisting of sets, each containing two possible solutions. We must make a union of these sets. This can be done as follows by using the procedure **union**. See section 10.8 on page 133.

```
> 'union'(%);
```

$$\left\{ \left\{ x = -\frac{4}{5} + \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} + \frac{2}{5}\sqrt{26} \right\},\right.$$
$$\left\{ x = -\frac{4}{5} - \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} + \frac{2}{5}\sqrt{26} \right\},$$
$$\left\{ x = -\frac{4}{5} + \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} - \frac{2}{5}\sqrt{26} \right\},$$
$$\left\{ x = -\frac{4}{5} - \frac{4}{5}\sqrt{26},\ y = -\frac{2}{5} - \frac{2}{5}\sqrt{26} \right\},$$
$$\left\{ y = -\frac{3}{10} + \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} + \frac{1}{10}\sqrt{201} \right\},$$
$$\left\{ y = -\frac{3}{10} - \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} + \frac{1}{10}\sqrt{201} \right\},$$
$$\left\{ y = -\frac{3}{10} + \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} - \frac{1}{10}\sqrt{201} \right\},$$
$$\left\{ y = -\frac{3}{10} - \frac{3}{10}\sqrt{201},\ x = -\frac{1}{10} - \frac{1}{10}\sqrt{201} \right\} \Big\}$$

Each of the eight sets of potential solutions can be checked by substitution into the set of equations. After application of `expand` and `simplify` we must check if the result is {0=0}. In section 1.3 on page 6, a solution of a third-degree polynomial equation is checked. The same process is executed here in a more advanced way, using a `do` loop and a choice structure in order to handle all solutions.

First, let's give a name to the set of candidates for solutions:

```
> candidates := %:
```

Let's create a void sequence to which all the found solutions can be appended:

```
> solutions := NULL;
```

Now we start a repetition, using the construction

`for in do od`

We take each of the elements of `candidates`, naming that element "`testcand`" temporarily:

```
> for testcand in candidates do
```

We substitute `testcand` in the set of equations and apply `expand` and `simplify` on each of the two resulting equations.

```
>   map(eq->lhs(eq)-rhs(eq), {eq1,eq2}):
>   subs( testcand , % ):
>   map( expand , % ):
>   map( simplify , % ):
```

If the result is {0}, then `testcand` satisfies the system of equations and should be appended to the sequence of found solutions:

```
>   if %={0} then solutions:=solutions,testcand fi:
```

Now the repetition structure should be terminated:

```
>   od:
```

After this last line the loop is executed, but all the commands have been closed with a colon, so we do not see any result. We have obtained:

```
> solutions;
```

$$\left\{y=-\frac{2}{5}+\frac{2}{5}\sqrt{26},\ x=-\frac{4}{5}+\frac{4}{5}\sqrt{26}\right\},$$
$$\left\{y=-\frac{2}{5}-\frac{2}{5}\sqrt{26},\ x=-\frac{4}{5}-\frac{4}{5}\sqrt{26}\right\},$$
$$\left\{y=-\frac{3}{10}+\frac{3}{10}\sqrt{201},\ x=-\frac{1}{10}+\frac{1}{10}\sqrt{201}\right\},$$
$$\left\{y=-\frac{3}{10}-\frac{3}{10}\sqrt{201},\ x=-\frac{1}{10}-\frac{1}{10}\sqrt{201}\right\}$$

In fact, this is the same result as yielded by `allvalues` without option, but the previous demonstration may show how you can check solutions of a system of polynomial equations, for instance, in a case where it is not clear if solutions are omitted by `allvalues`.

The method for testing candidates for solutions that is shown here can be used for sets of polynomial equations if all of these equations are reduced to the form `polynomial=0`. Our checking method relies fully on the power of `simplify`. In order to minimize the chance that solutions get lost by a failing simplification, an alternative check with `testeq` could be used or added. As always, choosing methods and judging results is the task of the user.

Error messages and warnings

Catchword Index

notes:

notes:

notes:

notes:

notes: