A little bit of help with Maple
Welcome!

The aim of this manual is to provide you with a little bit of help with Maple. Inside you will find brief descriptions on a lot of useful commands and packages that are commonly needed when using Maple. Full worked examples are presented to show you how to use the Maple commands, with different options given, where the aim is to teach you Maple by example – not by showing the full technical detail. If you want the full detail, you are encouraged to look at the Maple Help menu.

To use this manual, a basic understanding of mathematics and how to use a computer is assumed. While this manual is based on Version 10.01 of Maple on the PC platform on Windows XP, most of it should be applicable to other versions, platforms and operating systems.

This handbook is built upon the original version of this handbook by Maureen Edwards and Alex Antic. Further information has been gained from the Help menu in Maple itself.

If you have any suggestions or comments, please email them to Grant Cox (gcx@uow.edu.au).
# Contents

1 Introduction 4  
1.1 What is Maple? 4  
1.2 Getting started 4  
1.3 Operators 7  
  1.3.1 The Ditto Operator 9  
1.4 Assign and unassign 9  
1.5 Special constants 11  
1.6 Evaluation 11  
  1.6.1 Digits 13  
  1.6.2 The absolute value and the sign of a number 13  
1.7 Automatic simplifications 15  
1.8 Sets and lists 15  
1.9 Names, symbols and strings 18  

2 The basic commands 23  
2.1 Common mathematical functions 23  
2.2 Summation and products 23  
2.3 Limits 26  
2.4 Basic plotting 27  
  2.4.1 Basic two dimensional plots 27  
  2.4.2 Basic three dimensional plots 32  
2.5 Arrays, vectors and matrices 32  
2.6 Series expansions 39  
2.7 Simplifying and manipulation expressions 40  
2.8 Printing, reading and writing 47  

3 Calculus 52  
3.1 Differentiation 52  
3.2 Integration 54  

4 Solving equations 57  
4.1 Algebraic equations 57  
4.2 Differential equations 61  
4.3 Recurrence equations 64  

5 Some useful packages 65  
5.1 plots 66  
5.2 LinearAlgebra 69  
5.3 PDEtools 71
## 6 Programming in Maple

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Procedures</td>
<td>73</td>
</tr>
<tr>
<td>6.2 Modules</td>
<td>75</td>
</tr>
<tr>
<td>6.3 If statements</td>
<td>75</td>
</tr>
<tr>
<td>6.4 For statements</td>
<td>76</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this chapter, we will look at what Maple is, and how to start using it to solve many problems in mathematics.

1.1 What is Maple?

Maple is a powerful software program that can be used to solve general-purpose mathematical problems. Problems in the areas of mathematics, science and engineering (and many more) can be investigated using either Maple’s in-built commands, or by utilizing Maple’s powerful native programming language to create your own personalized programs.

While this handbook aims to provide you with everything you need to learn the basics of Maple, further information about Maple can be found either in numerous books in the library (library.uow.edu.au) or at Maple’s web site: www.maplesoft.com

1.2 Getting started

To use Maple, you need to type your commands in a Maple worksheet. There are two types of worksheet: a classic worksheet and a visually enhanced worksheet. Figures 1.1 and 1.2 show examples of a classic and visually enhanced worksheet, respectively. While the visually enhanced worksheet looks pretty, and presents the mathematics on the screen in a similar format to what you would write down on a sheet of paper, it is not that user friendly. On the other hand, the classic worksheet is very user friendly, but is not as visually pleasing. However, the author of this handbook highly recommends that you always use the classic worksheet - at least until you become good at using Maple. As such, everything in the handbook is based on the assumption that you are using the classic worksheet.

To start using Maple (in Microsoft Windows), go to:

Start → All programs → Maple 10 → Classic Maple Worksheet 10

Note:
Always save your Maple worksheet as a classic Maple worksheet - it should have the file type extension ‘.mws’, while the visually enhanced worksheet has the file type extension ‘.ms’.

□
A little bit of help with Maple

Figure 1.1: An example of a classic worksheet in Maple.

Figure 1.2: An example of a visually enhanced worksheet in Maple.
When you type Maple commands into the classic worksheet, the color of the text should be red. Thus, all Maple commands and inputs in this handbook will also be in red text. Maple commands must be typed after the prompt: >

Almost every statement entered into Maple must end with either a semicolon (i.e., ;) or a colon (i.e., :). If a semicolon is at the end of a statement, then the output will be displayed to the screen, while if a colon is used, then the output is suppressed - although the statement is still executed. To execute a command, press the Enter or Return key, while the blinking cursor is anywhere in the statement. The cursor does not need to be at the end of the statement in order to execute it. Once a command is executed, a new prompt is displayed, ready for your next command.

Example 1.1:

> 2;
2

However, if a colon, :, is used, then the statement is executed, but the result is not printed to the screen:

> 2:

Note:
If you have executed a statement, which has a semicolon at the end of the statement, but no output is given by Maple, then either:

- your statement has no output - indeed some Maple commands have no output, or
- Maple cannot find the answer to your statement.

Alternatively, you can execute a command by moving the blinking cursor to the appropriate command line and pressing the icon with a single exclamation mark, i.e., !. Further, you can execute the entire worksheet by pressing the icon with three exclamation marks, i.e., !!!.

If you want a new prompt above where the cursor is currently located, press Ctrl k, while if you want a new prompt below, press Ctrl j. In regards to the prompt, the [ shows the size of the execution group. If you have a multi-line input statement, then the [ will grow to encompass the entire statement. You can also add extra lines to an execution group, i.e., input statement, by pressing Shift Enter. All statements in the same execution group are executed at the same time - but in order of appearance.

When typing commands, please keep in mind that Maple is case sensitive. For example, pi and Pi are different commands.
mathematical | relational
---|---
+ | addition | < | less than
− | subtraction | > | greater than
* | multiplication | <= | less than or equal to
/ | division | >= | greater than or equal to
^ or ** | exponentiation | <> | not equal
= | equal | = | equal

<table>
<thead>
<tr>
<th>logical</th>
<th>set</th>
</tr>
</thead>
</table>
and | intersect
or | union
xor | minus
implies | subset
not | member

Table 1.1: The basic operators in Maple.

Finally, Maple has an in-built help support. You can either go to the Help menu and search for a desired topic or command, or if you already know the command, i.e., command_name, then you can instead type ? command_name to bring up the relevant help page. Notice that this is one of the very few situations where you do not need to have a colon or semicolon at the end of the statement. These help pages provide a detailed explanation on the commands and available options, usually with some basic examples.

1.3 Operators

The basic operators in Maple, including mathematical, relational, logical and set, are listed in Table 1.1. For more information about these operators, type ? operators.

Note:
When typing decimal numbers you need to be a little bit careful with blank spaces, because . is an operator in Maple that performs non-commutative, or dot product, multiplication on its arguments. Some examples of using the operator . in Maple are given below:

Example 1.2:
```
> 2 . 5; 10
> 2 .5; 10
> 2. 5; Error, unexpected number
> 2.5; 2.5
```

For more information on the . operator, see ? .

Table of contents: Index
We will see examples of how to use the relational, logical and set operators later. However, some examples of how to use the mathematical operators are presented in the following example:

**Example 1.3:**

\[
\begin{align*}
> 2 + 3; & \quad 5 \\
> 2 - 3; & \quad -1 \\
> 2 \times 3; & \quad 6 \\
> 2 / 3; & \quad \frac{2}{3} \\
> 2^{-12}; & \quad 4096
\end{align*}
\]

The blank spaces between the numbers and the operators in the above example can be neglected. For example, we could instead have written:

**Example 1.4:**

\[
\begin{align*}
> 2+3; & \quad 5 \\
> 2+3; & \quad 5 \\
> 2+3; & \quad 5 \\
> 2+3; & \quad 5 \\
> 2+3; & \quad 5 \\
> 2-3; & \quad -1 \\
> 2*3; & \quad 6 \\
> 2/3; & \quad \frac{2}{3} \\
> 2^12; & \quad 4096
\end{align*}
\]

In other words, Maple ignores spaces, *in general*. There are a small number of exceptions, where one is demonstrated in Example 1.2.
Note:
The standard order of operations holds true in Maple.

Example 1.5:

\[
\begin{align*}
> 2 + 6/2; & \quad 5 \\
> (2 + 6)/2; & \quad 4
\end{align*}
\]

1.3.1 The Ditto Operator

The ditto operator, \%, is a useful function within Maple that allows previously generated results to be accessed. The three ditto operators, \%, \%% and \%%% refer to the last, second last and third last non-null results computed in the worksheet, respectively.

For example, consider the following Maple worksheet:

Example 1.6:

\[
\begin{align*}
> 3; & \quad 3 \\
> x; & \quad x \\
> \text{hello}; & \quad \text{hello} \\
> \%; & \quad \text{hello} \\
> \text{%%;} & \quad x
\end{align*}
\]

Note:
The ditto operator, %, refers to the most recent statement executed – this need not be located directly above.

1.4 Assign and unassign

To assign a value to a variable, the command := is used. For example:
Example 1.7:

To assign the value of 2 to the variable \( x \), use

\[
> x := 2;
\]

\[ x := 2 \]

Thus, if \( x \) is used elsewhere, Maple knows its value to be 2, i.e.,

\[
> y := 3*x;
\]

\[ y := 6 \]

To \textit{unassign} a value to a previously assigned variable, use the \texttt{unassign} command. For example:

Example 1.8:

Assuming \( x \) is defined as above in Example 1.7, then to unassign \( x \), use

\[
> x;
\]

\[ 2 \]

\[
> \texttt{unassign('x')}:
\]

\[
> x;
\]

\[ x \]

Note:
The argument of the \texttt{unassign} command \textit{must} have the single quotes, ’ ’, around the variable to be unassigned. The single quotes have the effect of delaying the evaluation of the variable, so that the command \texttt{unassign('x')} unassigns the variable, \( x \), and not the value of the variable.

To clear \textit{all} stored information in a Maple worksheet, use the command \texttt{restart}. 

Example 1.9:

\[
> x := -2;
\]

\[ x := -2 \]

\[
> x;
\]

\[ -2 \]

\[
> \texttt{restart}:
\]

\[
> x;
\]

\[ x \]

Note:
The **restart** command will *not* be executed if it is read in from a file.

### 1.5 Special constants

Maple has a number of *protected* characters and words. For a complete list, see the help pages `? initialfunctions` and `? initialconstants`. Three of the most important constants are:

- **I**: complex symbol, where $I^2 = -1$.
- **infinity**: mathematical infinity, i.e., $\infty$.
- **Pi**: constant value of $\pi$.

`pi` is also used in Maple, but it is not a protected word. Instead, it is only used as a symbol to represent the constant value of $\pi$. Similarly, the characters `e` and `E` are not protected in Maple. To use the value of the exponential, `e`, consider the following example:

```
Example 1.10:

\[
\begin{align*}
> & \quad x := \exp(1); \\
& \quad x := e \\
> & \quad \ln(x); \\
& \quad 1
\end{align*}
\]
```

where `exp` is the exponential function and `ln` is the logarithmic function. See `? exp` and `? ln` for more information.

**Note:**

Any symbol or variable name beginning with an *underscore*, i.e., `_`, is reserved for use by Maple library code and as such should not be used by you for a variable name. Also, variables names should not begin or end with a *tilde*, i.e., `~`.

### 1.6 Evaluation

When using special constants, such as **Pi**, Maple will by default represent them by their symbol, $\pi$, and not by their numerical value, 3.141592654. To obtain the value of a constant, you need to ask Maple to evaluate it using the function `evalf`, which converts an expression from it’s symbolic form into it’s numeric floating-point form.
Example 1.11:

\[
\text{> Pi;} \quad \pi \\
\text{> evalf(Pi);} \quad 3.141592654
\]

As \text{pi} is used to denote the symbol of the constant \(\pi\), then you \textit{cannot} obtain the numerical value of \(\pi\) via the \texttt{evalf} function. If you do try, then you will only obtain the symbol back, i.e.,

\[
\text{> evalf(pi);} \quad \pi
\]

Maple represents \textit{fractions} in rational form. To obtain the numerical value of a fraction, use the \texttt{evalf} function. For example:

Example 1.12:

\[
\text{> 10/7;} \quad \frac{10}{7} \\
\text{> evalf(10/7);} \quad 1.428571429
\]

Maple automatically converts an expression to it’s numerical decimal value if the expression contains a decimal number, i.e.,

\[
\text{> 10./7;} \quad 1.428571429 \\
\text{> 10/7.0;} \quad 1.428571429 \\
\text{> 10.0/7.;} \quad 1.428571429
\]

[ where Maple treats 10. to be the same as 10.0.]

Further, \texttt{evalf} can be used to also find the decimal value of an irrational number, i.e., see \texttt{Pi} above.

Maple has other evaluation functions. The three most common other evaluation functions are:

- \texttt{eval}: evaluates an expression at a point. See \texttt{? eval} for more information.
- \texttt{evalb}: determines the \textit{boolean} value of an expression, i.e., true or false. See \texttt{? evalb} for more information.
- \texttt{evalc}: determines the \textit{complex value} of an expression. See \texttt{? evalc} for more information.
Example 1.13:

```
> eval(2*x+1,x=2);
5
> evalb(2>3);
false
> evalc(sqrt(1 + I));
\frac{1}{2}\sqrt{2 + 2\sqrt{2}} + \frac{1}{2}i\sqrt{-2 + 2\sqrt{2}}
```

Note:
The function `eval` also has other uses, which we will see later!

1.6.1 Digits

In floating-point calculations, the default number of digits that Maple returns is 10. To change this value to some number $n$, where $n$ is a natural number from 1 to `kernelopts(maxdigits)`, we need to change the predefined Maple variable `Digits`. For example:

Example 1.14:

```
> Digits;
10
> Digits := 5;
Digits := 5
> evalf(Pi);
3.1416
```

1.6.2 The absolute value and the sign of a number

To find the absolute value of a number or expression, use the `abs` command.
Example 1.15:

\[
\begin{align*}
> \text{abs}(3); & \quad 3 \\
> \text{abs}(-3); & \quad 3 \\
> \text{abs}(\cos(3)); & \quad -\cos(3) \\
a := \text{abs}(2x-3); & \quad a := |2x - 3| \\
> \text{abs}(1 - 2I); & \quad \sqrt{5}
\end{align*}
\]

See \texttt{abs} for more information.

To determine the sign of a number, use the \texttt{sign} command.

Example 1.16:

\[
\begin{align*}
> \text{sign}(0); & \quad 1 \\
> \text{sign}(-2/3); & \quad -1 \\
> \text{sign}(2 - 3I); & \quad 1 \\
> \text{sign}(-2 - 3I); & \quad -1
\end{align*}
\]

For more information, see \texttt{sign}. To determine the sign of a real or complex expression, use the \texttt{signum} command.

Example 1.17:

\[
\begin{align*}
> \text{signum}(0); & \quad 0 \\
> \text{signum}(-2/3); & \quad -1 \\
> \text{signum}(1 - 2I); & \quad \left(\frac{1}{5} - \frac{2}{5}i\right) \sqrt{5}
\end{align*}
\]

Note that \texttt{signum}(x) = x/abs(x) for \(x \neq 0\).

See \texttt{signum} for more information.
1.7 Automatic simplifications

There are a number of simplifications that Maple will perform by default for any calculation. For example, if \( x \) is an arbitrary variable, then:

\[
\begin{align*}
> x - x; & \quad 0 \\
> x + x; & \quad 2x \\
> x + 0; & \quad x \\
> x \times x; & \quad x^2 \\
> x/x; & \quad 1 \\
> x \times 1; & \quad x \\
> x^0; & \quad 1 \\
> x^1; & \quad x
\end{align*}
\]

Calculating \( \infty/\infty \), or \( \infty - \infty \), returns the value

\[ > \infty/\infty; \quad \text{undefined} \]
while trying to calculate \( 0/0 \) returns the error message

\[ > 0/0; \quad \text{Error, numeric exception: division by zero} \]

1.8 Sets and lists

In Maple, there are two different ways to order a sequence of expressions, by either using a list or a set.

A list is an ordered sequence of expressions of the form \([e_1, e_2, \ldots]\), where \( e_1, e_2, \ldots \) are expressions. In this case, the two lists \([a, b]\) and \([b, a]\) are not equal.

A set is an unordered sequence of expressions of the form \( \{e_1, e_2, \ldots\} \). In this case, the two sets \( \{a, b\} \) and \( \{b, a\} \) are equal. When using a set to denote a sequence of expressions, do not assume that Maple will preserve the same order.

Thus, if you want an ordered sequence of expressions, then you should use a list, with the
A little bit of help with Maple

Two *very* useful commands that are often used in conjunction with sets and lists, are the `nops` and `op` commands. The `nops` command returns the number of *operands* in an expression. For example:

**Example 1.19:**

```
> L := [1,2,3,4];
L := [1, 2, 3, 4]
> S := {5,6,7};
S := {5, 6, 7}
> nops(L);
4
> nops(S);
3
```

The `nops` command can be used with more than just sets and lists. For example, it can be used to tell you how many terms in an equation, i.e.,

```
> eqn := 1 + x + y + x*y - y^2;
eqn := 1 + x + y + xy - y^2
> nops(eqn);
5
```

The `op` command extracts the operands from an expression. For example:

**Example 1.20:**

```
> S := {5,6,7};
S := {5, 6, 7}
> op(S);
5, 6, 7
> op(2,S);
6
```

The `op` command can also be used to select individual particular operands, i.e.,

```
> eqn := 1 + x + y + x*y - y^2;
eqn := 1 + x + y + xy - y^2
> op(5,eqn);
−y^2
```

For more help on `nops` and `op`, see ? `nops` and ? `op`. There are many operations that you can do on both lists and sets. Some examples are given below:
Example 1.21:

Sets remove duplication, while lists don’t, i.e.,

```
> {a,a,b};
{a,b}
> [a,a,b];
[a, a, b]
```

You can convert from a list to a set, and vice-versa

```
> convert([a,b],set);
{a,b}
> convert({a,b},list);
[a, b]
```

For more information on convert, see `? convert`. Maple may change the order of elements in a set, i.e.,

```
> {b,a,c};
{a, b, c}
```

To select elements from a list or set, use `[#]`, i.e.,

```
> L := [1,2,3,4];
L := [1, 2, 3, 4]
> S := {5,6,7};
S := {5, 6, 7}
> L[1];
1
> S[2];
6
> L[2..4];
[2, 3, 4]
```

If you want to add a new element to a list, use

```
> L1 := [op(L),12];
L1 := [1, 2, 3, 4, 12]
```

Similarly for sets, or use `union`. To remove an element from a set, use

```
> S minus 6;
{5, 7}
```

To remove an element from a list, it is slightly harder to do. Instead of `minus`, use

```
> remove(has,L1,3);
[1, 2, 4, 12]
```

The command `remove`, and its counterpart `select`, are very useful commands that can be used in many situations. See later for more details.
Another common command used in conjunction with sets and lists is the `seq` command. For example:

```
Example 1.22:

[ If

> L1 := [1,2,3];
\hspace{1cm} L1 := [1,2,3]

> L2 := [4,5,6];
\hspace{1cm} L2 := [4,5,6]

then to multiply the corresponding elements of \(L1\) and \(L2\) together to form a new list \(L3\), we can use

> L3 := [seq(L1[i]*L2[i],i=1..nops(L1))];
\hspace{1cm} L3 := [4,10,18]
```

For more help on `seq`, see `? seq`. An alternate way to calculate the product of elements from two lists, is to use the `zip` command, i.e.,

```
Example 1.23:

[ If

> L1 := [1,2,3];
\hspace{1cm} L1 := [1,2,3]

> L2 := [4,5,6];
\hspace{1cm} L2 := [4,5,6]

[ then

> zip((x,y)-> x*y,L1,L2);
\hspace{1cm} [4,10,18]
```

For more information on the `zip` command, see `? zip`.

### 1.9 Names, symbols and strings

Any expression may be assigned a name. In the above example, the expression \([1,2,3]\) has assigned the name \(L1\). A valid name in Maple should satisfy the following criteria:

- Starts with a letter, followed by zero or more letters, digits, and underscore characters – uppercase and lowercase letters are distinct.
- You can form names by enclosing any sequence of characters in a pair of left single quotes, i.e., ‘This is a name’.
Any valid Maple name formed without using left single quotes is precisely the same as the name formed by surrounding the name with left single quotes. Therefore, \texttt{x} and \texttt{'x'} both refer to the same name \texttt{x}.

Names can be formed with the concatenation operator \texttt{||} or with the \texttt{cat} command (see example below). For more information see \texttt{? ||} and \texttt{? cat}.

Names should not start with an underscore, \texttt{_}, or a tilde, \texttt{~}, and should not contain a slash, \texttt{/}.

\begin{example}
\begin{verbatim}
> This_is_a_valid_name := 2;
This_is_a_valid_name := 2

> 'I like this number!' := 42;
I like this name!: = 42

> 'I like this number!';
42

You can join names together using \texttt{||} to form new names, i.e.,

> i := 5;
i := 5

> p || i;
p5

> p || 2*i;
p10

If the right-hand expression is a sequence or a range and the operands of the range are integers or character strings, then Maple returns a sequence of names.

> a || (a, b, 4, 67);
aa, ab, a4, a67

> "foo" || (1..3);
"foo1", "foo2", "foo3"

You can include \texttt{'} in names by using two of them.

> 'Trees aren't pink!';
Trees aren't pink!
\end{verbatim}
\end{example}

See \texttt{? name} for more information.

There are two types of names: indexed names and symbols (non-indexed names). All of the above names are symbols because they are not of the form: \texttt{name[expression]}. An indexed name may appear almost anywhere a name can appear.
Example 1.25:

[ The following are examples of symbol names.

> a := 2;
\> a := 2

> distance := 3.5;
\> distance := 3.5

[ The following are examples of indexed names.

> a[1] := 2;
\> a[1] := 2

> a[2] := 3;
\> a[2] := 3

> a[3] := 5;
\> a[3] := 5

> distance[0] := 2.7;
\> distance[0] := 2.7

> distance[1] := 3.5;
\> distance[1] := 3.5

> A[1,2,3][x,y][2,1] := 0;
\> A[1,2,3][x,y][2,1] := 0

It is strongly recommended that you don’t use the same name for indexed and symbol names – otherwise things may go wrong!

Example 1.26:

> a[1] := 2;
\> a[1] := 2

> a[1];
\> 2

> a;
\> a

> a := 3;
\> a := 3

> a[1];
\> 3

> b := 2;
\> b := 2

> b[1] := 3;
\> b[1] := 3

> b;
\> b
A *string* is a sequence of characters that has no value other than itself. It cannot be assigned to, and will always evaluate to itself.

A string is written by enclosing any sequence of characters within a pair of double quote characters, i.e., "This is a string". Strings may also be formed with the concatenation operator ||, or with the cat command.

**Example 1.27:**

```
> "The sky is blue";
"The sky is blue"

> "The sea is green";
"The sea is green"

If you try to assign to a string, the following error occurs:

```
> "A dog has how many legs?" := 4;
Error, invalid left hand side of assignment
```

Using the || operator, we find

```
> "This" || " " || "and" || " " || "that";
"This and that"
```

Strings are very useful when you want to write messages to the screen. In particular, as we will see later, when writing procedures it can be useful to output error messages to the screen using strings.

There are a number of very useful commands that can be used in conjunction with names, symbols and strings, including:

- **convert**: Can be used to convert from one Maple expression into another. This is one of the most useful and powerful Maple commands to know! Here, in particular, it can be used to convert a name into a string, and vice-versa. See ? convert for more information.

- **cat**: Can be used to combine strings and names together. This is useful when you want to write informed messages to the screen. See ? cat for more information.

- **parse**: Can be used to evaluate a Maple expression contained within a string (see example below). Note that the string must consist of exactly one Maple expression. By default, the expression is parsed, and returned unevaluated. If the option ‘statement’ is specified, the string is parsed and evaluated, and the result is returned. See ? parse for more information.

- **assign**: Can be used to assign one value to another. See ? assign for more information.

- **type**: Can be used to check they type of a Maple expression. This is particularly useful when you have assigned a value to a name and you want to check that the type is correct. See ? type for more information.
• **whattype**: Can be used to see what Maple type an expression is. If you do not know what type something this, this command comes in handy. See `? whattype` for more information.

Example 1.28:

```maple
> convert(a,string);
"a"

> convert("I want to be a string!",string);
"I want to be a string!"

> convert("I don't like being a name",name);
I don't like being a name
```

To assign the value 6 to the name converted from the string "2 times 3 equals", you need to use the `assign` command:

```maple
> assign(convert("2 times 3 equals",name),6);

> "2 times 3 equals";
"2 times 3 equals"

> '2 times 3 equals';
6
```

Note that the `assign` command above produces no output. The `parse` command can be used to evaluate the expression within a string.

```maple
> a := 3;
a := 3

> parse("a");
a

> parse("a",statement);
3
```

The `type` command can be used to check the type of a Maple expression.

```maple
> type(b,name);
true

> b := 2;
b := 2

> type(b,name);
false

> whattype(b);
integer
```

For more information, see `? strings` and `? names`. 
Chapter 2

The basic commands

In this chapter we will go through some of the basic commands that you may need to use Maple.

2.1 Common mathematical functions

Maple has many built-in mathematical functions that are commonly used. These include: \( \sin, \cos, \tan, \exp, \ln \) and \( \sqrt{\cdot} \), among many more. For a full list of initially known functions in Maple, see \( ? \) initial functions.

For an example of how to use these functions, consider:

Example 2.1:

\[
\begin{align*}
&\text{\texttt{\textgreater} sin(Pi/2); 1} \\
&\text{\texttt{\textgreater} exp(x); e^x} \\
&\text{\texttt{\textgreater} exp(ln(x)); x} \\
&\text{\texttt{\textgreater} sqrt(4*x); 2\sqrt{x}}
\end{align*}
\]

2.2 Summation and products

In mathematics, there are many times when you either want to add or multiply a sequence of numbers or expressions.

In Maple, the two most commonly used functions for calculating summations are \texttt{sum} and \texttt{add}. The important difference between these two commands is that \texttt{sum} tries to calculate a formula for the sum, while \texttt{add} calculates the sum by simply adding the finite sequence of terms. Thus, it is strongly recommended that you use the \texttt{add} command if an explicit sum is needed, particularly when summing over all elements of a list, array, matrix, or similar data structure. See \( ? \) sum for a more detailed comparison between \texttt{sum} and \texttt{add}.
Similarly, there are two common functions used for calculating products, namely \texttt{product} and \texttt{mul}. The command \texttt{product} is used when you want to determine a formula that is equal to the product, whereas the \texttt{mul} command is used when you want to calculate the finite product of terms. Thus, \texttt{mul} is similar to \texttt{add}, in that it is \textit{strongly} recommended that you use \texttt{mul} if an explicit product of terms is needed.

For some examples of the summation and product commands, consider:

\begin{table}[h]
\begin{tabular}{|l|}
\hline
Example 2.2:
\hline
\texttt{The sum of the first 10 integers is calculated by} \\
\texttt{> add(i,i=1..10); 55} \\
\texttt{while the formula for the sum of the first } n \texttt{ integers is given by} \\
\texttt{> sum(i,i=1..n); } (n+1)^2/2 - n^2 - \frac{1}{2} \\
\texttt{Similarly, the product of the first 10 integers is found by} \\
\texttt{> mul(i,i=1..10); 3628800} \\
\texttt{while the formula for the product of the first } n \texttt{ integers is determined by} \\
\texttt{> product(i,i=1..n); } \Gamma(n+1) \\
\texttt{where } \Gamma \texttt{ denotes the GAMMA function. See } \texttt{? GAMMA} \texttt{ for more information.} \\
\hline
\end{tabular}
\end{table}

For all of the summation and product commands, the first argument in the command denotes the terms to be summed/multiplied, while the second argument denotes the index and the range of the index to be summed/multiplied over. For more information, see \texttt{? sum}, \texttt{? add}, \texttt{? product} and \texttt{? mul}.

For an example of how to add a sequence of numbers from a list (or set), consider:

\begin{table}[h]
\begin{tabular}{|l|}
\hline
Example 2.3:
\hline
\texttt{> L := [1,3,5,7]; }L := [1, 3, 5, 7] \\
\texttt{> add(L[i],i=1..4); 16} \\
\hline
\end{tabular}
\end{table}

\textbf{Note:}
If you are using either \texttt{sum} or \texttt{product} to determine a formula for the summation or product, and Maple is unable to calculate the answer, then the function call is returned.

\[ \square \]
For example:

**Example 2.4:**

\[
\sum_{n=2}^{k} \frac{n^k}{n-1}
\]

which shows that Maple is unable to calculate a closed form formula for the answer.

To calculate infinite sums (or products), consider:

**Example 2.5:**

\[
\sum_{x=1}^{\infty} \frac{1}{x^2} = \frac{\pi^2}{6}
\]

\[
\prod_{x=1}^{\infty} x^n
\]

If you want the sum of an expression, but you do not want Maple to calculate it, then you can use what is called the *inert* form of the `sum` command, i.e., `Sum`. For example:

**Example 2.6:**

\[
S := \sum_{k=0}^{n} x^k
\]

\[
X := 2 + 3S - S^3;
\]

If you want to determine the value of an inert `Sum`, use the `value` command, i.e.,

\[
\text{value}(S);
\]

**Note:**
The inert form of `product` is `Product`
2.3 Limits

Maple can calculate directional and bidirectional limits. To attempt to calculate the limiting value of an expression, say \( f(x) \), as \( x \) approaches \( a \), i.e., \( \lim_{x \to a} f(x) \), then we use the \texttt{limit} command.

The \texttt{limit} command has three arguments. The first argument is the expression that you want to take the limit of, say \( \frac{\sin(x-2)}{x-2} \). The second argument defines the limit you want to take, say as \( x \) approaches 2. And finally, the third argument, which is optional, tells Maple if you only want a directional limit from either the \texttt{left} or the \texttt{right}, or a \texttt{real} or \texttt{complex} limit. If no third argument is specified, then Maple assumes that you want the bidirectional real limit.

**Example 2.7:**

\begin{verbatim}
> limit(sin(x-2)/(x-2), x=2);
1
\end{verbatim}

If the bidirectional limit does not exist, then \texttt{undefined} is returned:

\begin{verbatim}
> limit(1/x, x=0);
undefined
\end{verbatim}

To calculate a directional limit, specify the direction as the third argument, i.e.,

\begin{verbatim}
> limit(1/x, x=0, left);
-\infty
> limit(1/x, x=0, right);
\infty
\end{verbatim}

You can take the limit at infinity, i.e.,

\begin{verbatim}
> limit(exp(-x), x=infinity);
0
\end{verbatim}

The inert form of \texttt{limit} is \texttt{Limit}. If the inert form of the limit is used, then Maple will not calculate the limit – until forced to do so using the \texttt{value} command.

**Example 2.8:**

\begin{verbatim}
> L := Limit(x^2, x=3);
L := \lim_{x \to 3} x^2
> value(L);
9
\end{verbatim}

See \texttt{? limit} for more information on the \texttt{limit} and \texttt{Limit} commands.
2.4 Basic plotting

You can use Maple to create two and three dimensional graphs easily. Here, we are going to look at some of the basic plotting commands.

2.4.1 Basic two dimensional plots

To create a two dimensional plot, the `plot` command is used. The `plot` command must have at least two arguments specified. The first argument is the expression to be plotted, while the second argument denotes the dependent variable and the range of which it is to be plotted over.

Example 2.9:

```maple
> plot(sin(x),x=0..2*Pi);
```

The `plot` command can have extra (optional) arguments. Some common options include:

- A vertical range – if this is specified then it must be the third argument. Unlike the horizontal range, when specifying the vertical range, you don’t specify a variable (see example below).

- `axes`: Specifies the type of axes to be drawn on the plot. `axes` can be set to the following: boxed, frame, none or normal, which can be specified in either upper or lower case letters. The default value is normal.

- `color`: Specifies the color of the lines. All lines in the plot will change to the color specified. See ? plot,color for more information of available colors.

- `labels`: Specifies the labels of the axes. By default, Maple will put the dependant variable (which from the specification of the horizontal range) on the horizontal axis, and leave the vertical axis with no label. To change this, you need to specify a string each for the horizontal and vertical axes (see example below).

- `labeldirections`: Specifies the directions of the axes labels. By default all labels will be written horizontally on the graphs. To change the direction, you need to set `labeldirections` equal to a list of two elements, where the first corresponds to the direction of the horizontal axis while the second element of the list corresponds to the direction of the vertical axis. These elements must only be equal to either horizontal or vertical, where case of letters does not matter (see example below).
• **línestyle**: Specifies the style of the lines. **línestyle** must be set equal to either be 1, 2, 3 or 4, or one of the following names: **SOLID, DOT, DASH, or DASHDOT**. The name must be in uppercase letters.

• **numpoints**: Specifies the minimum number of points used to generate the plot. If the lines on the graphs are looking jagged, you can increase the minimum number of points that Maple uses to draw the graphs by setting **numpoints** to a positive integer, where the default value is 50. By default, Maple uses adaptive techniques to draw graphs, so Maple may use more than the specified number of points.

• **thickness**: Specifies the thickness of the lines. **thickness** must be set equal to a non-negative integer, where by default it is equal to zero. The higher the number, the thicker the lines.

For more information of plotting options, see ? plot.options.

**Example 2.10:**

```maple
> plot(sin(x), x=0..2*Pi,-0.8..0.8,color=green,thickness=5,
    linestyle=4,labels=["x","\sin(x)"],
    labeldirections=[horizontal,vertical]);
```

If we change the axes to **boxed**, we find:

```maple
> plot(sin(x), x=0..2*Pi,-0.8..0.8,color=green,thickness=5,
    linestyle=4,labels=["x","\sin(x)"],
    labeldirections=[horizontal,vertical],axes=boxed);
```

You can also use the **plot** command to plot a list of points.
Example 2.11:

If you want to plot the points: \([0,0] , [1,1] , [1,0.5]\), then you need to create a list of the points, i.e.,

\[
\text{L := \{[0,0], [1,1], [1,0.5]\);
\]

\[
\text{plot(L, thickness=3);
}\]

To plot a closed shape, you need to add the first point to the end of the list:

\[
\text{plot([op(L), L[1]], thickness=3);
}\]

To fill in the closed shape with a color, set the option \text{filled} to true:

\[
\text{plot([op(L), L[1]], thickness=3, filled=true, color=blue);
}\]

To graph a parametric function, you can use the \text{plot} command as well.
Example 2.12:
If \( x = \sin t \) and \( y = \cos t \), for \( t \in [-\pi, \pi] \), then consider:

\[
> \text{plot}([\sin(t), \cos(t), t=-\Pi..\Pi], \text{color}=\text{blue}, \text{thickness}=4);
\]

It is possible to plot multiple functions on the same graph using the \texttt{plot} command.

Example 2.13:
If you want to plot both \( y = \sin x \) and \( y = \cos x \), over the same range of say, \( x \in [0, 2\pi] \), then consider:

\[
> \text{plot}([\sin(x), \cos(x)], x=0..2*\Pi, \text{thickness}=[4,2], \text{color}=[\text{blue}, \text{red}], \text{linestyle}=[1,3]);
\]

Note that the options for each line can be defined separately.

Note:
To plot multiple functions on the same plot, but with different ranges, you need to use the \texttt{plots[display]} command. This will be covered in more detail later.

To plot polar functions, you can use the \texttt{plot} command with the option \texttt{coords=polar}, and where function is defined for \( r \) and \( \theta \) parametrically.
Example 2.14:

To plot \( r = 3 \sin 3\theta \), consider:

three leaf rose

\[
\text{plot([3*sin(3*theta),theta,theta=0..2*Pi],coords=polar, thickness=4);}
\]

Some other interesting polar plots are:

limacon

\[
\text{plot([2-4*cos(theta),theta,theta=0..2*Pi],coords=polar, thickness=3,color=green);}\]

spiral of Archimedes

\[
\text{plot([exp(theta/3),theta,theta=0..4*Pi],coords=polar, thickness=3,color=blue);}\]

See \texttt{plot,polar} for more information on polar plotting, and see \texttt{plot,coords} for more information on plots in other coordinate systems.
2.4.2 Basic three dimensional plots

Three dimensional functions can be plotted in Maple in a similar manner as to two dimensional plots. However, instead of using the \texttt{plot} command, you use the \texttt{plot3d} command, and where you specify the range in two directions, not just one.

Example 2.15:

\begin{verbatim}
> plot3d(cos(x*y), x=-Pi..Pi, y=-Pi..Pi);
\end{verbatim}

Note that by default no axes are drawn in three dimensional plots! To have them drawn, you need to use the \texttt{axes} option.

\begin{verbatim}
> plot3d(cos(x*y), x=-Pi..Pi, y=-Pi..Pi, axes=frame, labels=[x,y,z]);
\end{verbatim}

[ Note: The options of \texttt{axes} in \texttt{plot3d} are the same as for \texttt{plot}.]

See \texttt{? plot3d, options} for more information about the possible options for \texttt{plot3d}.

Basically, all of the possibilities of what you can do in two dimensions with \texttt{plot} can be extended to do in three dimensions with \texttt{plot3d}, but where you have surfaces instead of lines. This raises the question of how do you draw three dimensional curves? This will be answered later, when we look at the \texttt{plots[spacecurve]} command.

2.5 Arrays, vectors and matrices

In early versions of Maple, before Maple 9, the way to create arrays, vectors and matrices was generally to use the \texttt{array} command (one alternative was to use the \texttt{linalg[matrix]} command). However, from Maple 9 onwards, the three data structures \texttt{Array}, \texttt{Vector} and \texttt{Matrix} were introduced, and are intended to replace the \texttt{array} command. Therefore, here
we look at how these new commands work in Maple.

In it’s simplest form, the **Array** commands works as follows:

**Example 2.16:**

You need to specify the number of dimensions of the **Array**, where there can be from 0 to 63 different dimensions. The range of each dimension must be defined. For a 1-dimensional **Array**:

```maple
> A := Array(1..10);
A := [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

while for a 2-dimensional **Array**:

```maple
> B := Array(1..2,1..3);
B :=

<table>
<thead>
<tr>
<th>0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
</tr>
</tbody>
</table>
```

For higher order dimensional **Array**s, Maple returns a summary of the **Array**:

```maple
> C := Array(1..2,1..3,0..512);  
C :=

1.2 x 1.3 x 0..512 3-D Array

Data Type: anything
Storage: rectangular
Order: Fortran_order
```

See [? Array](https://www.maplesoft.com/support/help/maple/viewer.aspx?view=5&guide=Array) for more information on the **Data Type**, **Storage**, **Order** and other options of an **Array**. You can refer to each element of an **Array** in the following way:

```maple
> B[1,2] := 3;
B[1,2] := 3
```

```maple
> B[2,3] := -1;
B[2,3] := -1
```

```maple
> B;

<table>
<thead>
<tr>
<th>0 3 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 -1</td>
</tr>
</tbody>
</table>
```

You can assign initial values to **Array**’s as follows:

```maple
> E := Array(1..2,1..3,2);
E :=

<table>
<thead>
<tr>
<th>2 2 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 2 2</td>
</tr>
</tbody>
</table>
```

```maple
> F := Array(1..3,1..4,[[1,2,1,2],[3,4,3,4],[5,6,5,6]]);
F :=

<table>
<thead>
<tr>
<th>1 2 1 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 4 3 4</td>
</tr>
<tr>
<td>5 6 5 6</td>
</tr>
</tbody>
</table>
```

You can perform simple arithmetic operations on **Array**’s provided they the **Array**’s are of same size, where the result is a new **Array** (of the same size) where the arithmetic operation has been applied to the corresponding elements of the **Array**’s.
Example 2.17:  

Assume the Array’s B, E and F are as defined in the previous example, i.e.,

```
> B;
0 3 0
0 0 -1

> E;
2 2 2
2 2 2

> F;
1 2 1 2
3 4 3 4
5 6 5 6
```

Then we can calculate:

```
> B - 3*E;
-6 -3 -6
-6 -6 -1

> B*E;
0 6 0
0 0 -2

> %/%%;
0 -2 0
0 0 2/7
```

Note that normal matrix multiplication does not work:

```
> B*F;
Error, (in rtable/Product) invalid arguments
```

Recall, that we also have the non-commutative multiplication operator ., i.e.,

```
> B . E;
0 6 0
0 0 -2
```

so in terms of Array’s, * and . are the same.

Array’s are useful when you want to store, manipulate and manage multi-dimensional data. However, when you have one or two dimensional data (i.e., something which can be expressed as a vector or a matrix), then it is better to use the Vector and Matrix data structures, as there exists many useful commands in the LinearAlgebra package that can be used to manipulate them.

The Vector data structure is useful when you want to perform matrix calculations on one-dimensional array’s. Vector’s differ to Array’s in that they are only one-dimensional and operations on Vector’s must obey the usual linear algebra rules for matrices.
Example 2.18:

The are a number of ways to define Vector’s, including:

```maple
> A := Vector(3);

> B := Vector(1..3,5);

> C := Vector[row]([1,2,3]);

> E := Vector[column]([4,5,6]);

> F := 2*B;

> B + F;

> B + C;

Error, (in rtable/Sum) invalid arguments

> C[2];

To convert a row vector into a column vector, or vice-versa, use the convert command:

> G := convert(C,Vector[column]);

> type(G,Vector[column]);
```

See ? Vector for some other ways to define a Vector and it’s elements. When specifying a Vector, you can add the option of either [row] or [column], as shown in the above example. In these cases, the Vector is outputted and treated as either a row or column vector. By default, Maple assumes the Vector is a column vector.
You can perform operations on Vector’s in the same way as Array’s – provided they are of the same dimension and type – except for multiplication.

Example 2.19:

\[
B := \text{Vector}(1..3, 5); \\
C := \text{Vector}[\text{row}](\{1, 2, 3\}); \\
E := \text{Vector}[\text{column}](\{4, 5, 6\}); \\
\]

The * multiplicative operator does not work for Vector’s, i.e.,

\[
B * E; \\
E * C;
\]

However, the . non-commutative multiplicative operator works are follows:

\[
B . E; \\
C . C;
\]

If you multiply two Vector’s of the same orientation, then the dot product of the two Vector’s is returned, i.e.,

\[
B . E; \\
C . C;
\]

If the two Vector’s are of different orientation, then the two vector’s are multiplied according to the standard rules of matrix multiplication, i.e.,

\[
E . C; \\
C . E;
\]

Note that the blank spaces around the . operator above do not matter, as Maple will ignore them when dealing with non-numbers.

To perform more complicated operations on Vector’s, you need to use commands from the LinearAlgebra package – we will see more about this later.

Note:
An alternate way to define a Vector is to use the < > notation.
Example 2.20:

```
> A := <1,0,1>
> ;
A :=
[1]
[0]
[1]
> type(A, Vector);
true
```

A Matrix is similar to a Vector, except that it can be one or two dimensional.

Example 2.21:

```
> M := Matrix(2);
M :=
[0 0]
[0 0]
> N := Matrix(2,3);
N :=
[0 0 0]
[0 0 0]
> P := Matrix(1..2,1..3,5);
P :=
[5 5 5]
[5 5 5]
> Q := Matrix([[1,2,3],[4,5,6]]);
Q :=
[1 2 3]
[4 5 6]
> R := Matrix(2,3, symbol=m);
R :=
[m1,1 m1,2 m1,3]
[m2,1 m2,2 m2,3]
```

Note:
To refer to an element of a Matrix, you need to specify both the row and the column of the element in the Matrix – even when the Matrix is one-dimensional.

Example 2.22:

```
> Q := Matrix([[1,2,3],[4,5,6]]);
Q :=
[1 2 3]
[4 5 6]
> Q[2,3];
6
> T := Matrix([3,5,7]);
T := [3 5 7]
> T[1,2];
5
```
You can perform simple arithmetic operations on Matrix’s in a similar manner to Array’s and Vector’s.

**Example 2.23:**

```
> P := Matrix(1..2,1..3,5);

P :=

\[
\begin{bmatrix}
5 & 5 & 5 \\
5 & 5 & 5 \\
\end{bmatrix}
\]

> Q := Matrix([[1,2,3],[4,5,6]]);

Q :=

\[
\begin{bmatrix}
1 & 2 & 3 \\
4 & 5 & 6 \\
\end{bmatrix}
\]

> P - 2*Q;

\[
\begin{bmatrix}
3 & 1 & -1 \\
-3 & -5 & -7 \\
\end{bmatrix}
\]

To perform matrix multiplication, you need to use the . operator:

```
> S := Matrix(3,2,[<1,-1>,<2,1>,<3,1>]);

S :=

\[
\begin{bmatrix}
1 & -1 \\
2 & 1 \\
3 & 1 \\
\end{bmatrix}
\]

> Q . S;

\[
\begin{bmatrix}
14 & 4 \\
32 & 7 \\
\end{bmatrix}
\]

> Q . P;

Error, (in LinearAlgebra:-MatrixMatrixMultiply) first matrix column dimension (3) <> second matrix row dimension (2)
```

To perform more complicated operations on Matrix’s, you need to use commands from the LinearAlgebra package – we will see more about this later.

**Note:**
An alternate way to define a Matrix is to use the \(< \quad \> | \quad > >\) notation, where each \(< \quad >\) defines a column of the Matrix.

```
> A := <<1,0,1> | <2,1,3>>;

A :=

\[
\begin{bmatrix}
1 & 2 \\
0 & 1 \\
1 & 3 \\
\end{bmatrix}
\]

> type(A,Matrix);

true
```

The Matrix command has many useful options, see ? Matrix for more information. A particularly useful option is the ability to define identity matrices, as given below:
Example 2.25:

```maple
M := Matrix(3,3,shape=identity);
M :=
[1 0 0]
[0 1 0]
[0 0 1]
```

Because we have specified that \(M\) has `shape=identity`, then we are unable to change the elements of \(M\), i.e.,

```maple
> M[1,1] := 2;
Error, invalid assignment to identity diagonal
> M[1,2] := 3;
Error, invalid assignment of non-zero to identity off-diagonal
```

2.6 Series expansions

Maple provides the ability to easily calculate different types of series expansions. Two of the most commonly used series expansions in Maple are `series` and `taylor`.

Example 2.26:

```maple
S := series(exp(x),x=0,8);
S := 1 + x + \frac{1}{2} x^2 + \frac{1}{6} x^3 + \frac{1}{24} x^4 + \frac{1}{120} x^5 + \frac{1}{720} x^6 + \frac{1}{5040} x^7 + O(x^8)
> series(cos(x),x=1,3);
\cos(1) - \sin(1)(x - 1) - \frac{1}{2} \cos(1)(x - 1)^2 + O((x - 1)^3)
```

The \(O()\) term denotes the order of the series expansion. To remove the order term, say in order to plot the expansion, use the `convert` command, i.e.,

```maple
T := convert(S,polynom);
1 + x + \frac{1}{2} x^2 + \frac{1}{6} x^3 + \frac{1}{24} x^4 + \frac{1}{120} x^5 + \frac{1}{720} x^6 + \frac{1}{5040} x^7
```

Note that if the series is expanded about \(infinity\), then an asymptotic expansion is generated.

```maple
> series(x^3/(x^4+4*x-5),x=infinity);
\frac{1}{x} - \frac{1}{2x^2} + \frac{3}{2x^3} + O\left(\frac{1}{x^4}\right)
```

For more information, see `? series`.

Table of contents : Index

39/81
2.7 Simplifying and manipulation expressions

There are many commands that can be used to simplify and manipulate expressions. In this section, some of the more common commands will be discussed.

Example 2.27:

[ Expressions can be factorized using the \texttt{factor} command.

\begin{verbatim}
> factor(6*x^2+18*x-24); 6(x + 4)(x - 1)
> factor((x^3-y^3)/(x^4-y^4)); (y + x)(x^2 + y^2)
\end{verbatim}

[ You can factorize an equation in the same manner:

\begin{verbatim}
> eqn := y^2 - 1 = x^2 + 4*x + 4;
> factor(eqn); (y - 1)(y + 1) = (x + 2)^2
\end{verbatim}

Alternatively, you can factorize the left hand or the right hand side of an equation separately, by using the \texttt{lhs} and \texttt{rhs} commands.

\begin{verbatim}
> lhs(eqn); y^2 - 1
> rhs(eqn); x^2 + 2*x + 4
> factor(lhs(eqn)) = factor(rhs(eqn)); (y - 1)(y + 1) = (x + 2)^2
\end{verbatim}

\textbf{Note:} The \texttt{factor} command cannot be used to factorize expressions that cannot be perfectly factorized.

Example 2.28:

\begin{verbatim}
> eq := x^2 + 2*x + 1 + y;
> factor(eq); x^2 + 2*x + 1 + y
\end{verbatim}

To factor the equation, we need to use the \texttt{student[completesquare]} command. We will see more on this later.
There are times when you want to factorize the denominator or numerator of an expression. In this case, the commands `denom` and `numer` come in handy.

Example 2.29:

```maple
> yeqn := (sin(x)^2 + 2*sin(x) + 1)/(x - 1/x); 
yeqn := \frac{\sin(x)^2 + 2\sin(x) + 1}{x - \frac{1}{x}}

> yeqn_denom := denom(yeqn);
\quad x - \frac{1}{x}

> yeqn_numer := numer(yeqn);
\quad \sin(x)^2 + 2\sin(x) + 1

> factor(yeqn_numer)/factor(yeqn_denom);
\frac{(\sin(x) + 1)^2x}{(x - 1)(x + 1)}
```

Of course, in this case, we can just use the `factor` command normally:

```maple
> factor(yeqn);
\quad \frac{(\sin(x) + 1)^2x}{(x - 1)(x + 1)}
```

See ? factor for more information.

The opposite of factorizing expressions is, of course, expanding expressions. To do this, use the `expand` command.

Example 2.30:

```maple
> expand((x - 1)*(x + 1));
\quad x^2 - 1

> expand((x+1)/(x+2));
\quad \frac{x}{x + 2} + \frac{1}{x + 2}
```

For quotients of polynomials, only sums in the numerator are expanded – products and powers are left alone. See the `normal` command for dealing with quotients of polynomials.

```maple
> expand(1/(x+1)/x);
\quad \frac{1}{(x + 1)x}
```

The `expand` command can be used for more than just polynomials:

```maple
> expand(sin(x+y));
\quad \sin(x)\cos(y) + \cos(x)\sin(y)

> expand(exp(a+ln(b)));
\quad e^{ab}
```
If you have an expression that can be simplified using well-known simplification rules, then you can use the simplify command to do this. The simplify searches the expression for function calls, square roots, radicals, and powers, and then invokes the appropriate simplification procedures.

Example 2.31:

\[
\begin{array}{l}
> \text{simplify}(4^{(1/2)}+3); \\
5 \\
> \text{te1} := \exp(a+\ln(b*\exp(c))); \\
\text{te1} := e^{(a+\ln(b^e))} \\
> \text{simplify(te1)}; \\
b^e a + c
\end{array}
\]

There are times when using the simplify command causes unwanted and unnecessary difficulties, because Maple considers the analytical issue of branches for multi-valued functions. In situations where such branches are not of interest, then various options can be applied.

Example 2.32:

\[
\begin{array}{l}
> \text{g}:=\sqrt{x^2}; \\
g := \sqrt{x^2} \\
> \text{simplify(g)}; \\
csgn(x)x \\
\text{where csgn is the sign function for real and complex functions – see } ? \text{csgn} \text{ for more information. However, if we are only interested in the symbolic simplification, then:} \\
> \text{simplify(g,symbolic)}; \\
x \\
\text{Alternatively, we could have done:} \\
> \text{simplify(g,assume=real)}; \\
x \\
> \text{simplify(g,assume=positive)}; \\
x
\end{array}
\]

See ? simplify for more information.

To combine terms in sums, products, and powers into a single term, the combine command is used. For many functions, the transformations applied by the combine command are the inverse of the transformations that are applied by the expand command.
Example 2.33:

\[
\begin{align*}
> & \text{expand(sin(a+b));} \\
& \quad \sin(a) \cos(b) + \cos(a) \cos(b) \\
> & \text{combine(sin(a)*cos(b) + cos(a)*sin(b));} \\
& \quad \sin(a + b) \\
> & \text{E := exp(x)^2*exp(y);} \\
& \quad E := (e^x)^2e^y \\
> & \text{combine(E);} \\
& \quad e^{2x+y}
\end{align*}
\]

There may be times when you only want certain terms to be combined. To do this, you need to specify the appropriate options in the `combine` command.

Example 2.34:

\[
\begin{align*}
> & \text{eqn := 4*sin(x)^3+exp(y)*exp(x);} \\
& \quad eqn := 4 \sin(x)^3 + e^y e^x \\
> & \text{combine(eqn);} \\
& \quad -\sin(3x) + 3 \sin(x) + e^{x+y} \\
> & \text{combine(eqn,trig);} \\
& \quad -\sin(3x) + 3 \sin(x) + e^x e^y \\
> & \text{combine(eqn,exp);} \\
& \quad 4 \sin(x)^3 + e^{x+y}
\end{align*}
\]

When you want to collect together the coefficients of a variable or function in an equation, then you should use the `collect` command.

Example 2.35:

\[
\begin{align*}
> & \text{f := a*ln(x)-ln(x)*x-x;} \\
& \quad a \ln(x) - \ln(x) x - x \\
> & \text{collect(f,ln(x));} \\
& \quad (a - x) \ln(x) - x \\
\end{align*}
\]

When collecting coefficients, Maple does not automatically sort them into any particular order.

\[
\begin{align*}
> & \text{g := 3*(x - 2) + x*(2 - 3*x^2);} \\
& \quad g := 3x - 6 + x(2 - 3x^2) \\
> & \text{collect(g,x);} \\
& \quad -6 - 3x^3 + 5x \\
\end{align*}
\]

To do so, you need to use the `sort` command.

\[
\begin{align*}
> & \text{sort(collect(g,x));} \\
& \quad -3x^3 + 5x - 6
\end{align*}
\]
See ? sort for more information. When you want to collect more than one variable or function in an expression, you do so by specifying a list of objects to be collected.

Example 2.36:

```maple
> p := x*y+a*x*y+y*x^2-a*y*x^2+x+a*x;
p := xy + axy + yx^2 - ayx^2 + x + ax
> collect(p, x);
(y - ay)x^2 + (y + ay + 1 + a)x
> collect(p, [x,y]);
(1 - a)yx^2 + ((1 + a)y + 1 + a)x
> collect(p, [y,x]);
((1 - a)x^2 + (1 + a)x)y + (1 + a)x
```

Note that the order the objects are to be collected in will alter the output. If this is not desired, then use the distributed option.

```maple
> collect(p, [x,y], distributed);
(1 + a)x + (1 + a)xy + (1 - a)yx^2
```

See ? collect for more information. Additionally, if you have a polynomial, then you can ask Maple to pick out particular coefficients using the coeff command.

Example 2.37:

```maple
> p := 4*x^2 + 3*y^3 - 5;
p := 4x^2 + 3y^3 - 5
> coeff(p,x,2);
4
> coeff(p,x^2);
4
> coeff(p,x,0);
3y^3 - 5
```

See ? coeff for more information.

If you want all of the coefficients, then you can either use the coeff command manually for each coefficient, or you can use the coeffs command.
Example 2.38:

\[
> s := 3*v^2*y^2 + 2*v*y^3;
\]
\[
s := 3v^2y^2 + 2vy^3
\]
The coefficients of \( v \) can be picked out by:

\[
> \text{coeffs}(s, v, 'vt');
\]

\[
2y^3, 3y^2
\]
where \( \text{vt} \) specifies what the corresponding power of \( v \) is, i.e.,

\[
> \text{vt};
\]

\[
v, v^2
\]
We can pick out the coefficients of \( y \) in a similar manner:

\[
> \text{coeffs}(s, y, 'yt');
\]

\[
2v, 3v^2
\]
\[
> \text{yt};
\]

\[
y^3, y^2
\]
See \texttt{coeffs} for more information.

Alternatively, you may only want to select, or remove, certain terms from an expression. To do so, use the \texttt{select}, \texttt{remove} and \texttt{selectremove} commands.

Example 2.39:

\[
> f := 2*\exp(a*x)*\sin(x)*\ln(y);
\]
\[
f := 2e^{ax} \sin(x) \ln(y)
\]
\[
> \text{select}(\text{has}, f, x);
\]

\[
e^{ax} \sin(x)
\]
\[
> \text{remove}(\text{has}, f, x);
\]

\[
2 \ln(y)
\]
\[
> \text{selectremove}(\text{has}, f, x);
\]

\[
e^{ax} \sin(x), 2 \ln(y)
\]
\[
> g := 2*\exp(a*x)*\sin(x)*\ln(y) + \cos(x)*\exp(b*y) + 3;
\]
\[
g := 2e^{ax} \sin(x) \ln(y) + \cos(x)e^{by} + 3
\]
\[
> \text{select}(\text{has}, g, \cos);
\]

\[
\cos(x)e^{by}
\]
\[
> \text{remove}(\text{has}, g, [\cos, \ln]);
\]

\[
3
\]
For more information, see \texttt{?select}, \texttt{?remove} and \texttt{?selectremove}.

We have seen previously that we can use the \texttt{op} and \texttt{nops} commands to manipulate elements in a list. We can also use these commands in other expressions.
**Example 2.40:**

To know the number of terms in an equation, use the `nops` command, i.e.,

```maple
> f := expand(((x-4)*(x^2 + x - 3) + x^2*(x - 3))^2);
f := 4x^6 - 24x^5 + 8x^4 + 132x^3 - 95x^2 - 168x + 144
> nops(f);
7
```

To create a sequence of all the terms, use the `op` command, i.e.,

```maple
> op(f);
4x^6, -24x^5, 8x^4, 132x^3, -95x^2, -168x, 144
```

To select the $n$th term of the expression, use:

```maple
> op(1,f);
4x^6
> op(2,f);
-24x^5
> op(3,f);
8x^4
```

and so forth. Further, you can use the `op` command to find the next level of terms, i.e.,

```maple
> op(4,f);
123x^3
> op(op(4,f);
123, x^3
```

If you want to substitute values, or expressions, into an equation or expression, use the `subs` command.

**Example 2.41:**

```maple
> w := a*b + b - 3*c;
w := ab + b - 3c
> subs(a=2,w);
3b - 3c
> subs(a=0,w);
b - 3c
> subs([a=1],[b=2],w);
4 - 3c
> subs([a=1, b=3],[b=2],w);
6 - 3c
```
Example 2.42:

There is a difference between the \texttt{eval} and \texttt{subs} commands:

\begin{verbatim}
> z := 6*u*v^2 - 3*v + 3*exp(v - u);
  z := 6uv^2 - 3v + 3e^{v-u}
> subs(u=2,v=2,z);
  42 + 3e^0
> eval(z,[u=2,v=2]);
  45
\end{verbatim}

For more information, see ? \texttt{subs}.

2.8 Printing, reading and writing

When creating programs, you will often want to print messages to the screen, or to a file. In this section, we are going to look at how to print data to the screen and write data to files.

The \texttt{print} command displays the values of the arguments of the command to the screen, and returns NULL as the function value. The arguments are printed separated by a comma and a blank space.

Example 2.43:

\begin{verbatim}
> print(I,Love,Maths);
  I, Love, Maths
> print("This is a string");
  "This is a string"
> x := 42;
  x := 42
> print(x);
  42
> print(seq(i,i=1..10));
  1, 2, 3, 4, 5, 6, 7, 8, 9, 10
\end{verbatim}

The \texttt{print} commands returns no value.

\begin{verbatim}
> y := print(4*x^2);
  16
  y :=
> y;
\end{verbatim}

See ? \texttt{print} for more information.
The print command is fairly limited in the format of how the output appears. One alternative is to use the printf command. To do so, you need to specify the format string that defines the format of the output, where for full details see ? printf. Some examples of the printf command are given below.

Example 2.44:

[To output strings, use the %s format:

> printf("%s\n","This is cool!");
This is cool!
> printf("%s %s %s\n",This, is, ”cool!”);
This is cool!
> x := 42;
> printf("The answer is x = %s", convert(x,string));
The answer is x = 42
Alternatively, to print the number x = 42 without converting to a string, use
the %g format, i.e.,
> printf("The answer is x = %g", x);
The answer is x = 42

For full details on all the different types of formats, see ? printf. There a number of different ways to write data to a file. One of these is the fprintf command, which is related to the printf command.

Example 2.45:

> x := 23; y := -1/x;
> x := 23
> y := -1
> 23

You first need to use the fopen command to open the file you want to print to, i.e.,
> fd := fopen(”temp_file”, WRITE):
fprintf(fd, ”x = %d, y = %g”,x,y):
fclose(fd):
which will have written: x = 23, y = -0.0434783 to the file: temp_file. The location of this file will depends on a number of factors. To be certain of it’s location, use:
> fd := fopen(”C:/temp_file”, WRITE):
fprintf(fd, ”x = %d, y = %g”,x,y):
fclose(fd):
where the file is located on the C: drive. Note that you must close the file using the fclose command – otherwise the file will appear empty.
For more information, see ? fprintf, fopen and fclose. Some other useful commands to write data to a file include the writeln, writedata and writestat commands.

**Example 2.46:**

The writeln command writes a string to either the screen or to a file:

```maple
> writeln(terminal,"This is a test","So is this");
This is a test
So is this
```

The 26 is the number of characters returned from the writeln command. To suppress this, use the colon : instead. To write the strings to a file, use:

```maple
> writeln("C:/whatever.txt","This is a test","So is this");
```

The writedata command can be used to numerical data to a file.

```maple
> A := [[0,1,3.4,2,5.3],[-3,1,5.6,2,4.5]];
A := [[0, 1, 3.4, 2, 5.3], [-3, 1, 5.6, 2, 4.5]]
```

```maple
> writedata("C:/A_data.txt",A,float):
```

which writes

```
0 1 3.4 2 5.3
-3 1 5.6 2 4.5
```

to the file A_data.txt, and is located on the C: drive. If

```maple
> B := [[x,1,3.4,2,5.3],[y,1,5.6,2,4.5]]; 
B := [[x, 1, 3.4, 2, 5.3], [y, 1, 5.6, 2, 4.5]]
```

```maple
> writedata("C:/B_data.txt",B,float);
```

Error, (in writedata) Bad data found x because x and y are not numerical values. To get around this, you need to specify the individual formats for each data, i.e.,

```maple
> writedata("C:/B_data.txt",B,[string,integer,float,integer,float]);
```

fclose("C:/B_data.txt");

The writestat command writes a string or an expression to a file. In some ways, it is the more general form of both writeln and writedata.

For more information, see ? writedata, ? writeln and writestat. Another alternative on how to write data to a file, is to use the writeto and appendto commands. Simply put, you use the writeto command to specify what file to want to start writing to, and then from this point onwards any commands executed in the Maple worksheet will have their output written to the specified file. If you only want to append to a file that already has something in it, then use the appendto command, which will start writing at the end of the specified file.
Example 2.47:

```maple
> writeto("C:/C_data.txt");
> y := x^2 - 2*x + 1;
> 3*4;
> writeto(terminal);
> y;
```

where the command `writeto(terminal)` tells Maple to write back to the screen again.

See `? writeto` and `? appendto` for more information.

The opposite of writing is, of course, reading. To read data from a file, you can use `readline`, `readdata` or `readstat`, in a similar manner to how the write versions work.

Example 2.48:

Suppose that a file called SomeData.txt is located on the C: drive, and contains:

```
1 2 3 4 5
6 7 8 9 0
```

To read the first column of this data, use:

```maple
> readdata("C:/SomeData.txt",1);
```

where, unless told otherwise, Maple assumes the numbers are floats. Further, to read in all 5 columns:

```maple
> readdata("C:/SomeData.txt",1);
```

If instead SomeData.txt contains:

```
x 2 3 4 5
y 7 8 9 0
```

then to read in all the data, you need to specify the type of each column, i.e.,

```maple
> readdata("C:/SomeData.txt",[string,integer,float,integer,float]);
```

For more information, see `? readdata`, `? readline` and `? readstat`.

Finally, if you have a file that contains Maple commands, then you can tell Maple to execute...
these commands by using the \texttt{read} command.

\begin{example}

Suppose that a file called \texttt{SomeCommands.txt} is located on the C: drive, and contains:

\begin{verbatim}
y := sin(x);
subs([z=y],z^2 - 1);
\end{verbatim}

To get Maple to read in and execute these commands, use:

\begin{verbatim}
> read "C:/SomeCommands.txt";
\end{verbatim}

\begin{verbatim}
y := sin(x)
sin(x)^2 - 1
\end{verbatim}

Note that the \texttt{restart} command will not be executed if Maple reads it in from a file.

\end{example}
Chapter 3
Calculus

In this chapter, we look at how to calculate derivatives and integrals of expressions.

3.1 Differentiation

To calculate the partial derivative of an expression, use the \texttt{diff} command.

Example 3.1:

\begin{verbatim}
[> diff(sin(x),x); cos(x)
[> diff(sin(x),y); 0

You can calculate higher order derivatives in a number of ways, i.e.,

[> diff(diff(x^2 - 3*x + 4,x),x); 2

or,

[> diff(x^2 - 3*x + 4,x,x); 2

or,

[> diff(x^2 - 3*x + 4,[x,x]); 2

or,

[> diff(x^2 - 3*x + 4,x$2); 2

where \$ is operator for forming an expression sequence, i.e.,

[> x$4; x, x, x, x
\end{verbatim}
For more information, see $\texttt{?} \$ \text{ and } \texttt{? diff}$. The $\$ operator is particularly useful when dealing with higher order derivatives.

**Example 3.2:**

\[
> \ z := \sin(x) \cdot \cos(y); \\
> \ 
\]

\[
> \text{diff}(z, x \cdot 42, y \cdot 76); \\
> \ 
\]

The inert form of the $\texttt{diff}$ command, is the $\texttt{Diff}$ command.

**Example 3.3:**

\[
> \text{Diff}(f(x,y), [x, y]) = \text{diff}(\sin(x \cdot y), [x, y]); \\
> \frac{\partial^2}{\partial y \partial x} f(x, y) = -\sin(xy)xy + \cos(xy) \\
\]

There is also a differential operator in Maple, namely the $\texttt{D}$ command.

**Example 3.4:**

\[
> \text{D} (\sin); \\
> \cos \\
> \text{D} (\cos); \\
> -\sin \\
[\text{To calculate higher order derivatives, use the } \texttt{@@} \text{ operator, i.e.,} \\
> \text{(D@@2)(sin)}; \\
> -\sin \\
The $\texttt{D}$ operator is particularly useful when you want to specify the derivative of an expression evaluated at a point, i.e., when you need to specify initial or boundary conditions:

\[
> \text{D} (f)(0) = 0; \\
> D(f)(0) = 0 \\
[\text{or, in other words,} \\
> \text{convert(D(f)(0) = 0,diff);} \\
> \left( \frac{d}{dt} f(t1) \right) \bigg|_{t1=0} = 0 \\
\]

The $\texttt{D}$ operator can also be used for partial derivatives, where you need to specify which independent variable you want to differentiate with respect to.
Example 3.5:

\[
\begin{align*}
> & \ D[1](F)(x,y) \\
& \text{convert(\%,diff);} \\
& D_1(F)(x,y) \\
& \frac{\partial}{\partial x} F(x,y) \\
> & \ D[2](F)(x,y) \\
& \text{convert(\%,diff);} \\
& D_2(F)(x,y) \\
& \frac{\partial}{\partial y} F(x,y)
\end{align*}
\]

To calculate higher order partial derivative operators, consider

\[
\begin{align*}
> & \ D[1,1,2](F)(x,y) \\
& \text{convert(\%,diff);} \\
& D_{1,1,2}(F)(x,y) \\
& \frac{\partial^3}{\partial y \partial x^2} F(x,y)
\end{align*}
\]

For more information, see \texttt{? D} and \texttt{? @@}.

3.2 Integration

The opposite of differentiation is, of course, integration. To integrate an expression, use the \texttt{int} command.

Example 3.6:

\[
\begin{align*}
> & \ \text{int}(\sin(x),x); \\
& -\cos(x)
\end{align*}
\]

Note that no constant of integration appears in the result. To calculate a definite integral:

\[
\begin{align*}
> & \ \text{int}(\sin(x),x=0..\Pi); \\
& 2 \\
> & \ \text{int}(\sin(x),x=0..\pi); \\
& 1 - \cos(\pi)
\end{align*}
\]

To calculate a double integral, you need to repeat the \texttt{int} command, i.e.,

\[
\begin{align*}
> & \ \text{int}(\text{int}(\sin(x),x),x); \\
& -\sin(x)
\end{align*}
\]
Note:
There is no shorthand way to represent double, triple and higher order integrals – you must repeat the int command for each integral.

Example 3.7:

```plaintext
> int(x,x);
\frac{x^2}{2}
```

If we try a shorthand way, similar to higher order derivatives, then Maple either ignores the extra input or returns an error, i.e.,

```plaintext
> int(x,x,x);
\frac{x^2}{2}
```

```plaintext
> int(x,x$3);
\frac{x^2}{2}
```

```plaintext
> int(x,[x,x,x,x]);
Error, (in int) wrong number (or type) of arguments
```

The inert form of int is the Int command.

Example 3.8:

```plaintext
> Int(f(x),x=a..b);
\int_a^b f(x)dx
```

```plaintext
> eqn := Int(Int(x*y,x),y);
eqn := \int \int xy \, dx \, dy
```

To get Maple to evaluate the Int command, use the value command, i.e.,

```plaintext
> value(eqn);
\frac{x^2 y^2}{4}
```

Note:
The value command can be used to force Maple to evaluate inert commands, such as Int, Diff, Limit, Product and Sum, by applying the lowercase versions, i.e., int, diff, limit, product and sum.

Sometimes Maple cannot evaluate an integral analytically. In this case, Maple will simply return your call. However, in such situations, if there are no non-numeric values then you can ask Maple to determine a numerical approximation to the integral using the evalf command.
Example 3.9:

\[
\begin{align*}
\text{MyInt} & := \text{int}(\sin(\cos(\sin(x))), x=0..\pi); \\
\text{MyInt} & := \int_{0}^{\pi} \sin(\cos(\sin(x))) \, dx \\
\text{To determine a numerical approximation to the above integral, use the \texttt{evalf} command, i.e.,} \\
\text{evalf(MyInt);} & = 2.147394008
\end{align*}
\]
Chapter 4

Solving equations

In this chapter, we look at how to use Maple to solve algebraic, differential and recurrence equations.

4.1 Algebraic equations

To solve algebraic equations, use the `solve` command.

Example 4.1:

```maple
> eqn := f = m*a;

To solve `eqn` for `a`:

```maple
> solve(eqn,a);

Note that Maple returns what `a` is equal to – it does not return an equation. If there is more than one solution, Maple will return a sequence of solutions, i.e.,

```maple
> eq := x^4-5*x^2+6*x=2;

> sols := solve(eq,x);

> sols[3];

You can use the `solve` command to solve a system of equations, i.e,

```maple
> e1 := 4*x - 5*y = 3;
> e2 := x + 7*y = 0;

> solve({e1,e2},{x,y});
```

```maple
{y = -1/11, x = 7/11}
```
Note:
If you specify the variables to be solved for using a set, as in the above example, then Maple will return the answer in any order. In the above example, the order of the variables were specified by \{x,y\}, but Maple returned the answer with \(y\) first, and then \(x\). However, if you execute the `solve` command again, then Maple may return the answers in a different order. This is particularly true for large systems of equations. If you want Maple to *always* return the solution in a desired order, then specify the order of the variables using a list (see example below).

**Example 4.2:**

```maple
*> eqn1 := 3.2*x + 1.3*y + 4.2*z = 5;
  eqn2 := 8.7*x + 19*y + 11.2*z = 94;
  eqn3 := x + y/4 + z = 1;

  eqn1 := 3.2*x + 1.3*y + 4.2*z = 5
  eqn2 := 8.7*x + 19*y + 11.2*z = 94
  eqn3 := x + y/4 + z = 1

*> solve({eqn1, eqn2, eqn3}, [x, y, z]);
[[x = 0.4969502408, y = 5.187800963, z = -0.7939004815]]

*> solve({eqn1, eqn2, eqn3}, [x, z, y]);
[[x = 0.4969502408, z = -0.7939004815, y = 5.187800963]]

*> solve({eqn1, eqn2, eqn3}, [y, x, z]);
[[y = 5.187800963, x = 0.4969502408, z = -0.7939004815]]
```

Note that because some of the coefficients of the variables in the above equations are floating point numbers, then Maple will automatically convert the answers to floating point numbers.

See ?`solve` for more information. If you have used the `solve` command to find the analytical solutions to an equation, but you want to know the floating point value of the solutions, then consider the following two methods.

**Example 4.3:**

```maple
*> eq := x^4-5*x^2+6*x-2;
  eq := x^4 - 5x^2 + 6x - 2

*> sols := solve(eq,x);
  sols := 1, 1, \sqrt{3} - 1, -1 - \sqrt{3}

If you want a floating point approximation of the solution, then you can either use `evalf`, i.e.,

*> evalf(solve(eq,x));
  1., 1., 0.732050808, -2.732050808

or you can use the `fsolve` command.

*> fsolve(eq,x);
  -2.732050808, 0.732050808, 1., 1.
```
The `fsolve` command works in pretty much the same way as the `solve` command. If the equation is a polynomial, then Maple will return all real roots, by default. If the equation is not a polynomial, then Maple will return a single root of the equation. There are some options to try to get Maple to solve for more roots.

**Example 4.4:**

```plaintext
> fsolve(sin(x) = 1, x);
1.570796327

To get Maple to solve for a different root, either specify a range, i.e.,

```plaintext
> fsolve(sin(x) = 1, x = -6..1);
−4.712388980
```

or use the `avoid` option and specify a starting point, i.e.,

```plaintext
> fsolve(sin(x) = 1, x = 8, avoid = {x = 1.570796327, x = -4.712388980});
7.853981636
```

If you want all the roots of a polynomial, where some of the roots are complex, then you need to specify the `complex` option, i.e.,

```plaintext
> polynomial := 23*x^5 + 105*x^4 - 10*x^2 + 17*x;
polynomial := 23x^5 + 105x^4 − 10x^2 + 17x

> fsolve(polynomial);
−4.536168981, −0.6371813185, 0.

> fsolve(polynomial, x, complex);
−4.536168981, −0.6371813185, 0., 0.3040664543 − 0.4040619058 I,
0.3040664543 + 0.4040619058 I
```

Note that if Maple cannot find a solution, then NULL is returned, i.e.,

```plaintext
> fsolve(x^2 + 1 = 0, x);
```

For more information, see `?fsolve`. An alternate to using the `solve` command to determine the analytical solution of a system of equations, is the `eliminate` command.

**Example 4.5:**

```plaintext
> eqn1 := x^2 + y^2 - 1;
eqn1 := x^2 + y^2 - 1

eqn2 := x^3 - y^2*x + x*y - 3;
eqn2 := x^3 − y^2x + xy − 3

> eliminate( { eqn1, eqn2 }, x);

\[ \begin{align*}
\{x &= \frac{3}{-2y^2 + y + 1} , nourm n y^4 + 4y^6 + 6y^3 − 4y^5 + 4y^2 − 2y + 8\} \\
\end{align*} \]

where Maple returns one possible expression for the desired variable and whatever equations are left when the elimination is made.
A little bit of help with Maple

See ? eliminate for more information.

To determine integer-valued solutions of an equation, use the isolve command.

**Example 4.6:**

```
> isolve(x^2 - x - 6);
{x = -2}, {x = 3}
> isolve(3*x-4*y=7);
{x = 5 + 4*Z1}, {y = 2 + 3*Z1}
```

where _Z1 is an arbitrary constant. If isolve cannot determine an integer-valued solution, then NULL is returned, i.e.,

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

For more information, see ? isolve. Additionally, if you want to find integer-valued solutions (mod n), then use the msolve command.

**Example 4.7:**

```
> msolve(3*x-4*y=1,7*x+y=2,19);
{y = 11}, {x = 15}
> msolve(2^i=3,19);
{i = 13}, {i = 18*Z1}
```

See ? msolve for more information.
4.2 Differential equations

In this section we will look at how to solve differential equations. For ordinary differential equations, use the `dsolve` command.

Example 4.8:

```maple
> ode := diff(y(x),x,x) = 2*y(x) + 1;
ode := \frac{d^2}{dx^2} y(x) = 2y(x) + 1

> dsolve(ode);
y(x) = e^{\sqrt{2}x}\cdot C2 + e^{-\sqrt{2}x}\cdot C1 - \frac{1}{2}
```

where \( C1 \) and \( C2 \) are arbitrary constants of integration. You can specify conditions to determine the constants of integration, i.e.,

```maple
> ics := y(0)=1, D(y)(0)=0;
ics := y(0) = 1, D(y)(0) = 0

> sol := dsolve({ode, ics});
sol := y(x) = \frac{3}{4}e^{\sqrt{2}x} + \frac{3}{4}e^{-\sqrt{2}x} - \frac{1}{2}
```

To test to see if the solution satisfies the ode, use the `odetest` command, i.e.,

```maple
> odetest(sol,ode);
0
```

where if zero is returned, then the solution satisfies the ode. If you also want to check that the solution satisfies the initial conditions, then specify a list of equations that you want to check whether the solution satisfies, i.e.,

```maple
> odetest(sol,[ode, ics]);
[0, 0, 0]
```

where the first zero corresponds to the ode, the second zero to the first initial condition and the third zero to the second initial condition, respectively.

Note:
In the above example, to specify initial conditions that contain a derivative, the `D` operator is used.

For more information, see `?dsolve`. Maple can also determine numerical solutions to odes using the `numeric` option in `dsolve`, i.e.,

Example 4.9:

```maple
> deq1 := diff(y(x),x) + sin(x)*y(x) = 0;
deq1 := \frac{d}{dx} y(x) + \sin(x)y(x) = 0
```
A little bit of help with Maple

\[
\text{ics} := y(0) = 1; \quad \text{ics} := y(0) = 1
\]

The analytical solution is given by

\[
\text{dsol} := y(x) = \frac{e^{\cos(x)}}{e}
\]

while the numerical solution is given by

\[
\text{dsol}_N := \text{proc}(x, \text{rkf}45)\ldots\text{end proc}
\]

where \( x, \text{rkf}45 \) denotes the numerical solution is using a Fehlberg fourth-fifth order Runge-Kutta method with degree four interpolant, and the \text{proc} \ldots \text{end proc} defines a procedure (which we will see more about later).

To see the numerical solution:

\[
\text{dsol}_N(0); \quad [x = 0., y(x) = 1.]
\]

\[
\text{dsol}_N(0.1); \quad [x = 0.1, y(x) = 0.995016612435748794]
\]

\[
\text{dsol}_N(2); \quad [x = 2., y(x) = 0.242647315785959495]
\]

Alternatively, you can use the \text{output=listprocedure} option, i.e.,

\[
\text{dsol}_N2 := \text{dsolve}(\text{deq1}, \text{ics}, \text{numeric}, \text{output=listprocedure}); \quad \text{dsol}_N2 := [x = \text{(proc}(x)\ldots\text{end proc}), y(x) = \text{(proc}(x)\ldots\text{end proc})]
\]

\[
\text{f} := \text{eval}(y(x), \text{dsol}_N2); \quad f := \text{proc}(x)\ldots\text{end proc}
\]

\[
\text{f}(2); \quad 0.242647315785959495
\]

Note:
To plot the numerical solution, Maple provides the \text{odeplot} command within the \text{plots} package (see next chapter).

To solve partial differential equations, use the \text{pdsolve} command.

\[
\text{PDE} := x \frac{\partial}{\partial y} f(x,y) - y \frac{\partial}{\partial x} f(x,y) = 0; \quad \text{PDE} := x \left( \frac{\partial}{\partial y} f(x,y) \right) - y \left( \frac{\partial}{\partial x} f(x,y) \right) = 0
\]

\[
\text{pdsolve}(\text{PDE}); \quad f(x,y) = \_F1(x^2 + y^2)
\]

where \( \_F1(x^2 + y^2) \) is an arbitrary function of \( x^2 + y^2 \).
The \texttt{pdsolve} command has many options to try to help Maple solve the pde. For example, you can suggest the form of the solution.

\begin{verbatim}
Example 4.11:
\begin{verbatim}
> PDE := S(x,y)*diff(S(x,y),y,x) + diff(S(x,y),x)*diff(S(x,y),y) = 1;
    PDE := S(x, y) \partial_x^2 S(x, y) + \partial_x S(x, y) \partial_y S(x, y) = 1

If we use \texttt{pdsolve} without any hints, then we get:
\begin{verbatim}
> pdsolve(PDE);
(S(x, y) = F1(x) - F2(y))\{\frac{d}{dx} F1(x) = \frac{c_1}{F1(x)}, \frac{d}{dy} F2(y) = \frac{1}{2 F2(y) x}\}
\end{verbatim}

However, if we suggest a form of the solution, then
\begin{verbatim}
> pdsolve(PDE,HINT=P(x,y)^(1/2));
(S(x, y) = \sqrt{F2(x) + F1(y) + 2xy})\text{ where } F1\text{ and } F2 \text{ are arbitrary functions of their arguments.}
\end{verbatim}
\end{verbatim}
\end{verbatim}

\textbf{Note:}
You can also use \texttt{pdsolve} to try to solve a system of partial differential equations.

\end{verbatim}

Maple can also be used to find a numerical solution to a partial differential equation.

\begin{verbatim}
Example 4.12:
\begin{verbatim}
> PDE := diff(u(x,t),t)=-diff(u(x,t),x);
    PDE := \frac{\partial}{\partial t} u(x, t) = - \frac{\partial}{\partial x} u(x, t)

subject to the initial-boundary conditions:
\begin{verbatim}
> IBC := u(x,0)=sin(2*Pi*x),u(0,t)=-sin(2*Pi*t);
    IBC := \{u(x, 0) = \sin(2\pi x), u(0, t) = - \sin(2\pi t)\}
\end{verbatim}

Then the numerical solution is given by
\begin{verbatim}
> pds := pdsolve(PDE,IBC,numeric,time=t,range=0..1);
\end{verbatim}

\texttt{Note that the numerical solution of a pde returns a module, not a proc like the numerical solution of an ode. Please see later for more information on module’s.}
\end{verbatim}
\end{verbatim}
To examine the numerical values of the solution, use the \texttt{values} option, i.e.,

\begin{verbatim}
> pds:-value(t=0)(0);
pds:-value(t=0)(1);
\end{verbatim}

\[
[x = 0., t = 0., u(x, t) = 0.]
[x = 1., t = 0., u(x, t) = -0.24492359829470494 \times 10^{-15}]
\]

The above values correspond to \(t = 0\) with \(x = 0\) and \(x = 1\). To plot the numerical solution, you don’t need the \texttt{odeplot} command, but instead you can use the \texttt{plot} option, i.e.,

\begin{verbatim}
> p1:=pds:-plot(t=0,numpoints=50):
p2:=pds:-plot(t=1/8,numpoints=50,color=blue):
p3:=pds:-plot(t=1/4,numpoints=50,color=green):
plots[display](p1,p2,p3);
\end{verbatim}

For more information, please see \texttt{?pdsolve}, \texttt{?pdsolve, system} and \texttt{?pdsolve, numeric}.

### 4.3 Recurrence equations

To solve recurrence relations, use the \texttt{rsolve} command.

\begin{verbatim}
Example 4.13:

\begin{verbatim}
> rsolve(f(n) = -3*f(n-1) - 2*f(n-2), f(k));
(2f(0) + f(1))(-1)^k + (-f(0) - f(1))(-2)^k
\end{verbatim}

You can apply initial conditions, i.e.,

\begin{verbatim}
> rsolve(f(n) = -3*f(n-1) - 2*f(n-2), f(1..2)=1, f);
f(n) = -3(-1)^n + (-2)^n
\end{verbatim}

You can make a procedure of the solution, i.e.,

\begin{verbatim}
> rr := rsolve(s(n) = 2*s(n-1), s(0)=1, s, 'makeproc');

rr := proc (n) if 1 < nargs then (procname)(args) else expand(2^n) end if end proc
\end{verbatim}

\begin{verbatim}
> rr(0), rr(1), rr(2), rr(3);
1, 2, 4, 8
\end{verbatim}
\end{verbatim}
Chapter 5

Some useful packages

In this chapter we will look at some packages that contain some commands that you may find useful in Maple.

To include a package into Maple, you need to use the `with` command.

```
Example 5.1:
[ To include the plots package, use

> with(plots);
   Warning, the name changecoords has been redefined

   [Interactive, animate, animate3d, animatecurve, arrow, changecoords,
    complexplot, complexplot3d, conformal, conformal3d, contourplot,
    contourplot3d, coordplot, coordplot3d, cylinderplot, densityplot, display,
    display3d, fieldplot, fieldplot3d, gradplot, gradplot3d, graphplot3d,
    implicitplot, implicitplot3d, inequal, interactive, interactiveparams,
    listcontplot, listcontplot3d, listdensityplot, listplot, listplot3d,
    loglogplot, logplot, matrixplot, multiple, odeplot, pareto, plotcompare,
    pointplot, pointplot3d, polarplot, polygonplot, polygonplot3d,
    polyhedra_supported, polyhedrplot, replot, rootlocus, semilogplot,
    setoptions, setoptions3d, spacecurve, sparsematrixplot, sphereplot,
    surfdata, textplot, textplot3d, tubeplot]

Note that if you use the semi colon ;, then you will see a list of all the commands available in the package. Also, in this case, a warning message is given saying that this package has redefined a command that already exists in Maple – this is something to be kept in mind if you need to use either the standard or the new definition of this command.

[ To include only a particular command, say the listplot command, use

> with(plots, listplot);
   [listplot]

Alternative, if you only want to use a command once or twice, then you can use the [] notation, i.e.,

> plots[listplot](seq([i,sin(i)],i=0..10));
```
5.1 plots

The plots package contains over fifty commands that can be used to plot various Maple expressions and data structures. A brief description of some of the more useful and commonly needed commands are presented below.

The display and display3d commands can be used to plot multiple plots in the same plot. Example 4.12 above provides an example of this for display.

Example 5.2:

```maple
with(plots):
F:=plot3d(sin(x*y), x=-Pi..Pi, y=-Pi..Pi):
G:=plot3d(x + y, x=-Pi..Pi, y=-Pi..Pi):
H:=plot3d([2*sin(t)*cos(s), 2*cos(t)*cos(s), 2*sin(s)], s=0..Pi, t=-Pi..Pi):
display([F, G, H]);
```

For more information, see ?plots[display]. To plot a list of points, or an array of data, the listplot command is very useful.

Example 5.3:

```maple
> t := Array([seq(i,i=1..10)]):
t := [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
> plots[listplot]([seq([t[i],sin(t[i])],i=1..10]),color=pink,thickness=3);
```

```tableofcontents
Index
```
See `plots[plots]` for more information. The command `dsolve` can be used to determine the numerical solution to and ordinary differential equation. To plot the resulting solution, you can use the `odeplot` command.

**Example 5.4:**

```maple
> with(plots, odeplot);

> p := dsolve(D(y)(x) = y(x), y(0)=1, type=numeric, range=-5..2):
> odeplot(p,thickness=3);
```

If the range is not specified in the `dsolve` command, then you need to do so in the `odeplot` command, i.e.,

```maple
> p2 := dsolve( diff(y(x),x) = sin(x*y(x)),y(0)=2,y(x),type=numeric):
> odeplot(p2,[x,y(x)],0..6,thickness=3);
```

See `plots[odeplot]` for more information. Another way to plot a polar curve, is to use the `polarplot` command.

**Example 5.5:**

```maple
> polarplot(1);
```

Table of contents: Index
To plot \( x = \cos \theta \) and \( y = \sin \theta \), use the similar format to the `plot` command with `coords=polar` option, i.e.,

\[
\text{polarplot}([\cos(t), \sin(t), t=0..2*\pi], \text{color}=\text{gold}, \text{thickness}=2);
\]

For more information, see `plots[polarplot]`. To plot a list of data in order to create a closed shaped, you can use the `polygonplot` and `polygonplot3d`.

Example 5.6:

\[
\text{onepoly := [[0,0],[0,1],[1,1],[1,0]];}
\]
\[
\text{polygonplot(onepoly, thickness=3);}
\]

To create a polygon, consider:

\[
\text{ngon := n -> [seq([\cos(2*Pi*i/n), \sin(2*Pi*i/n)], i = 1..n)];}
\]
\[
\text{polygonplot(ngon(5), \text{color}=\text{BLUE});}
\]

Similar closed shaped can be created in three-dimensions using `polygonplot3d`.

See `plots[polygonplot]` and `plots[polygonplot3d]` for more information. To write words, symbols and sentences onto plots, Maple provides the `textplot` command.
Example 5.7:

```maple
> p := plot(sin(x),x=-Pi..Pi):
delta := 0.05:
t1 := textplot([Pi/2,1+d,’Local Maxima (Pi/2, 1)’],align=ABOVE):
t2 := textplot([-Pi/2,-1,’Local Minima (-Pi/2, -1)’],align=BELOW):
display(p,t1,t2,thickness=3);
```

For further information, see `plots[textplot]`.

5.2 LinearAlgebra

We have seen how to perform simple operations on Vector’s and Matrix’s. Maple provides the LinearAlgebra package with over one hundred commands to manipulate Vector’s and Matrix’s.

Example 5.8:

```maple
> with(LinearAlgebra):

> P := Matrix(1..2,1..3,5);

\[
P := \begin{bmatrix}
  5 & 5 & 5 \\
  5 & 5 & 5 
\end{bmatrix}
\]

> Q := Matrix([[1,2,3],[4,5,6]]);

\[
Q := \begin{bmatrix}
  1 & 2 & 3 \\
  4 & 5 & 6 
\end{bmatrix}
\]

To transpose a Matrix, use the Transpose command, i.e.,

```maple
> T := Transpose(Q);

\[
T := \begin{bmatrix}
  1 & 4 \\
  2 & 5 \\
  3 & 6 
\end{bmatrix}
\]
```

To multiply to Matrix’s together, use the Multiply command, i.e.,

```maple
> Multiply(Q,T);

\[
\begin{bmatrix}
  14 & 32 \\
  32 & 77 
\end{bmatrix}
\]
```
For more information, see ? LinearAlgebra[Transpose] and ? LinearAlgebra[Multiply]. To determine the eigenvalues and eigenvectors of a given Matrix, use the Eigenvalues and Eigenvectors commands.

Example 5.9:

\[
\begin{bmatrix}
-1 & -3 & -6 \\
3 & 5 & 6 \\
-3 & -3 & -4 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
-4 \\
2 \\
2 \\
\end{bmatrix}
\]

so the Eigenvectors command returns a Vector containing the eigenvalues and a Matrix containing the corresponding eigenvectors as the columns of the Matrix. Thus, checking that \( Ax = \lambda x \), i.e.,

\[
\begin{bmatrix}
-2 \\
-2 \\
4 \\
\end{bmatrix}
= \begin{bmatrix}
\frac{\lambda}{2} & -1 & 1 \\
\frac{1}{2} & 0 & 1 \\
1 & 1 & 0 \\
\end{bmatrix}
\]

where here \( \lambda \) is

\[
-4
\]

and \( x \) is the corresponding eigenvector, i.e.,

\[
\begin{bmatrix}
\frac{1}{2} \\
\frac{1}{2} \\
1 \\
\end{bmatrix}
\]

Note:
The Multiply command can also be used to multiply Matrix’s with Vector’s and scalars, as demonstrated above.

5.3 PDEtools

The PDEtools package is a collection of commands and routines to help find analytical solutions to partial differential equations. However, here we present a couple of the commands that can be utilized elsewhere in Maple.

The dchange command can be used to change the variable in mathematical expressions.

Example 5.10:

```maple
with(PDEtools):
tr := x = r + s, y = r - s:
dchange(tr, PDE);
```

For more information, see ?PDEtools[dchange]. Another convenient command is the declare command.

Example 5.11:

```maple
with(PDEtools):
declare(y(x), prime=x);
```

Table of contents: Index

71/81
You can tell Maple to toggle between using the compact form and the full form, using the \texttt{OFF} and \texttt{ON} options, i.e.,

\begin{verbatim}
> OFF; ode;
\left(\frac{d^2 y(x)}{dx^2}\right) y(x) f(x) - 2 \left(\frac{d y(x)}{dx}\right)^3 x^6 + 2 \left(\frac{d y(x)}{dx}\right)^2 y(x) \left(\frac{d y(x)}{dx}\right) + y(x)^5

> ON; ode;
\end{verbatim}

\[ y'' y' f(x) - 2 y'^3 x^6 + 2 y'^2 y g' + y^5 \]

For more information, see \texttt{PDEtools[declare]}. 
Chapter 6
Programming in Maple

In this chapter, we look at how to use Maple to create functions and programs. The simplest way to create a function is by using the \( \rightarrow \) operator.

Example 6.1:

\[
\begin{align*}
> f := x \rightarrow 3x + 5; \\
& f := x \rightarrow 3x + 5 \\
> f(2); \\
& 11 \\
> g := (x,y) \rightarrow \sin(x)\cos(y) + xy; \\
& g := (x, y) \rightarrow \sin(x) \cos(y) + xy \\
> g(\pi/2, \pi); \\
& -1 + \frac{1}{2}\pi^2
\end{align*}
\]

See \( ? \rightarrow \) for more information.

6.1 Procedures

More advanced functions can be created using the \texttt{proc} command. This command can have numerous arguments, but returns only one value (which may be assigned to some name).

Example 6.2:

\[
\begin{align*}
> lc := \text{proc}( s, u, t, v ) \\
& \text{description "linear combination of the arguments";} \\
& \text{local lincom;} \\
& \text{lincom := } s * u + t * v; \\
& \text{return lincom} \\
& \text{end proc;} \\
& lc := \text{proc}( s, u, t, v ) \text{ local lincom; description "linear combination of} \\
& \text{the arguments"; lincom := } s*u + t*v; \text{ return lincom end proc}
\end{align*}
\]

Notice that the last statement before the \texttt{end proc} does not need to have a colon : or a semi-colon ;.
Note:
When writing programs and procedures, it is *good programming practice* to indent your code to make it easier to read. Further, you can use blank lines to enhance the readability of your code. Finally, you should include comments in your code telling the reader what you are doing. Comments can be inserted using the `#` command.

### Example 6.3:

```
DiffProd := proc(func1, func2, var)
    description "Calculates the derivative of the product of the two functions, func1 and func2, with respect to 'var'":
    local Dfunc1, Dfunc2, answer:
    Dfunc1 := diff(func1, var): # derivative of func1 w.r.t var
    Dfunc2 := diff(func2, var): # derivative of func2 w.r.t var
    # calculate derivative of product using product rule
    answer := Dfunc1*func2 + func1*Dfunc2:
    return answer
end proc;

ans := DiffProd(sin(x),cos(x),x);
diff(sin(x)*cos(x),x);
combine(ans);
```

Note:
If `Dfunc1`, `Dfunc2` and `answer` had not been defined as local (or global) variables, then Maple would print a warning message to the screen saying that they have implicitly been defined as local variables. It is good programming practice to create code that does not produce any error or warning messages.

For more information, see `? proc`. 
6.2 Modules

 Procedures can generally only be used to return one result. If you want multiple results returned, then use the module command.

Example 6.4:

 If you have two numbers, \( a \) and \( b \), and you want to know \( a + b \mod 5 \) and \( a \times b \mod 5 \), then consider:

\[
\begin{aligned}
> z5 := \text{module()}
& \quad \text{export plus, times;}
& \quad \text{plus := (a,b) \rightarrow a + b \mod 5;}
& \quad \text{times := (a,b) \rightarrow a \times b \mod 5;}
& \quad \text{end module;}
\end{aligned}
\]

where \text{plus} and \text{times} have been defined as the results to be exported. To access these results, you need to use the := operator, i.e.,

\[
\begin{aligned}
> z5:-\text{plus}(2,4); \\
> z5:-\text{times}(2,4);
\end{aligned}
\]

1
3

Note:
We saw an example of a module, when we looked at the \text{pdsolve} command.

See ? \text{module} for more information.

6.3 If statements

When writing programs, it is common to want to test to see if an expression is true, and then execute some other statements depending on the value of the first expression. Maple provides the if statement for this purpose. There is more than one representation of the if statement, due to optional arguments.

Example 6.5:

\[
\begin{aligned}
> a := 3; \\
& \quad b := 5; \\
> \text{if } a > b \text{ then } a \text{ else } b \text{ end if;}
\end{aligned}
\]

\[
\begin{aligned}
& a := 3 \\
& b := 5 \\
& 5
\end{aligned}
\]
> if b - a <> 0 then
    c := b - a
else
    c := a - b
end if;
c := 2

> if c < a and c < b then
    a - c;
    b - c
elif c < a and c > b then
    a - c;
    c - b
else
    a + b - c
end if;

1
3

The elif command above stands for ‘else if’.

Note:
The last statement before either an else, elif or end if statement does not have to have a colon : or a semi-colon ;.

□

The operator form of the if statement is given by the ‘if’ operator.

Example 6.6:

> a := 3; b := 5;

> 5*(Pi + ‘if’(a > b,a,b));

For more information, see ? if.

6.4 For statements

To repeat an expression, use the for statement. There is more than one representation of the for statement, due to optional arguments.
Example 6.7:

```maple
> summ := 0;
  for i from 1 to 10 do
    summ := summ + i
  end do;

<table>
<thead>
<tr>
<th>Step</th>
<th>Value of summ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>45</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
</tr>
</tbody>
</table>
```

If you don’t want to see each step, then use a colon : at the end of the end do command.

Note:
If you nest for statements, then Maple will not show the output for each step – even if you use a semi-colon ;. If you want to see the output, you need to use commands like print, writeln, etc.

Example 6.8:

```maple
> for i from 1 by 2 while i <= 100 do
  for j in [1,24,3,-6,0,2] do
    i*j
  end do;
end do;
```

However, if you replace i*j by print(i*j), then each step will be printed to the screen (not shown here as there are 300 lines of output!).

See ? for for more information.
Index

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/, 6</td>
<td></td>
</tr>
<tr>
<td>//, 6</td>
<td></td>
</tr>
<tr>
<td>+, 7</td>
<td></td>
</tr>
<tr>
<td>−, 7</td>
<td></td>
</tr>
<tr>
<td>/, 7</td>
<td></td>
</tr>
<tr>
<td>&lt;, 7</td>
<td></td>
</tr>
<tr>
<td>&lt;=, 7</td>
<td></td>
</tr>
<tr>
<td>&gt;, 7</td>
<td></td>
</tr>
<tr>
<td>&gt;=, 7</td>
<td></td>
</tr>
<tr>
<td>Γ, see GAMMA</td>
<td></td>
</tr>
<tr>
<td>*, 7</td>
<td></td>
</tr>
<tr>
<td>**, 7</td>
<td></td>
</tr>
<tr>
<td>∞, 11</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>′, 10</td>
<td></td>
</tr>
<tr>
<td>→, 73</td>
<td></td>
</tr>
<tr>
<td>:, 75</td>
<td></td>
</tr>
<tr>
<td>:=, 9</td>
<td></td>
</tr>
<tr>
<td>#, 74</td>
<td></td>
</tr>
<tr>
<td>$, 52</td>
<td></td>
</tr>
<tr>
<td>%, see ditto</td>
<td></td>
</tr>
<tr>
<td>%%, see ditto</td>
<td></td>
</tr>
<tr>
<td>%%%, see ditto</td>
<td></td>
</tr>
<tr>
<td>^, 7</td>
<td></td>
</tr>
<tr>
<td>′, 18</td>
<td></td>
</tr>
</tbody>
</table>

abs, 13
add, 23
and, 7
appendto, 49
Array, 32
  Data Type, 33
  Order, 33
  Storage, 33
assign, see :=, 21
cat, 19, 21
coeff, 44
coeffs, 44
collect, 43
distributed, 44
combine, 42
command, 65
commands
  case sensitive, 6
  executing, 6
complex number, see I
convert, 21
  list, 17
  set, 17
cos, 23
csgn, 42
D
  differential operator, 53
  initial conditions, 61
dchange, 71
declare, 71
delay evaluation, see ’
denom, 41
Diff
  evaluating Diff, 55
  inert form of diff, 53
diff, 52
Digits, 13
display, 66
ditto, see operators
dsolve, 61
  numeric, 61
  plotting numerical solution, 67
Eigenvalues, 70
Eigenvectors, 70
eliminate, 59
end of statement
colon ;, 6
  semicolon ;, 6
eval, 12
evalb, 12
evalc, 12
evalf, 11
evaluating an integral numerically, 55
  numerical value of solve, 58
evaluation
  at a point, 12
  boolean, 12
  complex, 12
  floating point, see evalf
executing a statement
  how to, 6
  no output, 6
execution group, 6
exp, 11, 23
expand, 41
factor, 40
fclose, 48
fopen, 48
for, 76
fprintf, 48
fractions, 12
fsolve, 59
  avoid, 59
  complex, 59
GAMMA, 24
help
  ?, 7
  in-built support pages, 7
I, 11
if, 75
implies, 7
index, 32
indexed, see name
infinity, see \(\infty\)
initial functions and constants, 11
inserting new input prompts, 6
Int
  evaluating Int, 55
  inert form of int, 55
int, 54
intersect, 7
isolve, 60
lhs, 40
Limit
  evaluating Limit, 55
  inert form of limit, 26
limit, 26
linalg
  matrix, 32
LinearAlgebra, 34, 69
Eigenvalues, 70
Eigenvalues, 70
Multiply, 69
Transpose, 69
list, 15
ln, 11, 23
maple functions, see initial functions and constants
Matrix, 32, 37
  \(< < > | < >>\), 38
  identity, 38
member, 7
minus, 7
module, 63, 75
msolve, 60
mul, 24
Multiply, 69
name, 18
  indexed, 19
  symbol, 19
nops, 16, 45
normal, 41
not, 7
number of digits, see Digits
numer, 41
odeplot, 67
odetest, 61
op, 16, 45
operators
  ditto, 9
  logical, 7
  mathematical, 7
  relational, 7
  set, 7
  or, 7
parse, 21
PDEtools, 71
  dchange, 71
  declare, 71
pdsolve, 62
Pi, see pi
pi
  difference between pi and Pi, 11
plot, 27
  axes, 27
  boxed, 27
Table of contents : Index
frame, 27
none, 27
normal, 27
color, 27
coords, 31
labeldirections, 27
labels, 27
linestyle, 28
list of points, 28
multiple functions, 30
numpoints, 28
options, 28
parametric plots, 29
polar, 30
thickness, 28
plot3d, 32
plots, 62, 66
display, 30, 66
display3d, 66
odeplot, 62, 67
polygonplot, 68
polygonplot3d, 68
spacecurve, 32
textplot, 68
polygonplot, 68
polygonplot3d, 68
print, 47
printf, 48
proc, 73
Product
evaluating Product, 55
inert form of product, 25
product, 24
read, 51
readline, 50
remove, 17, 45
restart, 10
not executed, 51
rhs, 40
rsolve, 64
select, 17, 45
selectremove, 45
seq, 18
series, 39
set, 15
sign, 14
simplify, 42
symbolic, 42
sin, 23
solve, 57
sort, 43
sqrt, 23
the square root function, 13
starting Maple, 4
string, 21
student[completesquare], 40
subs, 46
subset, 7
Sum
evaluating Sum, 55
inert form of sum, 25
sum, 23
symbol, see name
tan, 23
taylor, 39
textplot, 68
Transpose, 69
type, 21
unassign, 10
union, 7
value, 25, 55
Vector, 32, 34
< >, 36
column, 35
dot product, 36
row, 35
whattype, 22
worksheet
classic, 4
visually enhanced, 4
writedata, 49
writeline, 49
writestat, 49
writeto, 49
xor, 7
zip, 7

Table of contents : Index