Mathematical Software: An Introduction to Modern Mathematical Computing

Jonathan Borwein Matthew Skerritt

July 11, 2009

Preface

Thirty years ago mathematical, as opposed to applied numerical, computation was difficult to perform and so relatively little used. Three threads changed that:

- The emergence of the personal computer; identified with the iconic Macintosh but made ubiquitous by the IBM PC.
- The discovery of fibre-optics and the consequent development of the modern internet culminating with the foundation of the World Wide Web in 1989 made possible by the invention of hypertext earlier in the decade.
- The building of the *Three Ms*: *Maple, Mathematica* and MATLAB. Each of these is a complete mathematical computation workspace with a large and constantly expanding built-in "knowledge base". The first two are known as "computer algebra" or "symbolic computation" systems, sometimes written *CAS*. They aim to provide exact mathematical answers to mathematical questions like what is

  $$\int_{-\infty}^{\infty} e^{-x^2}\,dx,$$

  what is the real root of $x^3 + x = 1$, or what is the next prime number after $1,000,000,000$? The answers are

  $$\sqrt{\pi},\ \frac{\sqrt[3]{108 + 12\sqrt{93}}}{6} - \frac{2}{\sqrt[3]{108 + 12\sqrt{93}}},\ \text{and}\ 1000000007.$$

  The third 'M' is primarily numerically based. The distinction, however is not a simple one. Moreover, more-and-more modern mathematical computation requires a mixture of so-called *hybrid* numeric-symbolic computation and also relies on significant use of geometric, graphic and visualization tools. These we shall also introduce, both inside

Maple and later in the course using the "interactive geometry" package, *Cinderella*. We shall also make some use of MATLAB through a *Maple* interface, see also [5]. MATLAB is the preferred tool of many engineers and other scientist who need easy access to efficient numerical computation.

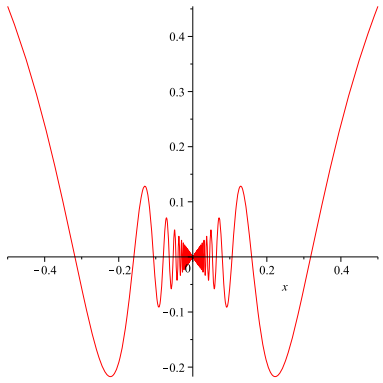
Of course each of theses threads relies on earlier related events and projects, and there are many other open-source and commercial software packages. For example, *Sage* is an opensource CAS, *GeoGebra* an opensource interactive geometry package and *Octave* is an opensource counterpart of MATLAB. But, here is not the place to discuss the merits and demerits of opensource alternatives. For many purposes *Mathematica* and *Maple* are interchangeable as adjuncts to mathematical learning. We propose to teach you the latter. After this course, you should find it easy to pick up the requisite skills to use *Mathematica* [13] or MATLAB.

Many introductions to computer packages aim to teach you the *syntax* (rules and structure) an *semantics* (meaning) of the system as efficiently as possible [6, 7, 8, 12]. They assume you know why you want to learn such things. By contrast, we intent to persuade you that *Maple* and other like tools are worth knowing assuming only that you wish to be a mathematician, a mathematics educator, a computer scientist, an engineer or scientist, or anyone else who wishes/needs to use mathematics better. We also hope to explain to you how to become an 'experimental mathematician' while learning to be better at proving things. To accomplish this our material is divided into four chapters on:

- **Elementary number theory**. Using only mathematics that should be familiar from high school, we shall introduce most of the basic computational ideas behind *Maple*. By the end of this chapter the hope is that the student can learn new features of *Maple* while also learning more mathematics.

- **Calculus of one and several variables.** In this chapter we shall revisit ideas met in first year calculus and introduce the basic ways to plot and explore functions graphically in *Maple*. Many have been taught not to trust pictures in Mathematics. This is bad advice. Rather, one has to learn how to draw trustworthy pictures.

$$> \; plot(x\sin(1/x), -\frac{1}{2}..\frac{1}{2})$$



- **Introductory linear algebra.** Similarly, in this chapter we will see how much of linear algebra can be animated (that is, brought to live) within a computer algebra system. While we suppose the underlying concepts will be familiar, this is not necessary. One of the powerful attractions of computer-assisted mathematics is that it allows for a lot of "learning while doing." especially by using the help files in the system and by learning how and when to consult internet mathematics resources such as *MathWorld, PlanetMath* or *Wikipedia*.

- **Visualisation and interactive geometric computation.** Finally, in the final chapter we shall explore more carefully how visual computing [9, 10] can help build mathematical intuition and knowledge. This is a theme we will emphasize throughout the course.

Each chapter has three main sections each of which forms the basis for a week's lectures. The fourth section of each chapter has exercises and additional examples. It is assumed the student will try many of them; indeed they will from the basis for the two mid-term exams. The final section of each chapter is entitled further explorations. It has two main intentions. One is to provide extra material for more mathematically advanced students. The other is as a set of possible topics for the final project which each student taking the course must produce.

Another subordinate goal of this course is to teach you how to produce mathematical documents such as the final project. To this end we will explore applications such as "MathType" and "TexPoint" which allow one to easily typeset mathematics within *Word* or *PowerPoint*. Moreover, Maple itself now provides a versatile presentation environment for mathematics. We will also touch upon the mathematical typesetting language TEX itself.

A more detailed discussion relating to many these brief remarks may be followed up in [4], in [2, 3] or [1], and in the references given therein.

Additional reading and references. We also supply a list of, largely recent, books at various levels that you may find useful or stimulating. Some are technical and some are more general. Some may be useful during a final project.

1. George Boros and Victor Moll, *Irresistible Integrals*, Cambridge University Press, New York, 2004.

2. Jonathan M. Borwein and Peter B. Borwein, *Pi and the AGM: A Study in Analytic Number Theory and Computational Complexity*, John Wiley & Sons, New York, 1987 (Paperback, 1998).

3. Christian S. Calude, *Randomness And Complexity, from Leibniz To Chaitin*, World Scientific Press, Singapore, 2007.

4. Gregory Chaitin and Paul Davies, *Thinking About Gödel and Turing: Essays on Complexity, 1970-2007,* World Scientific, 2007.

5. Richard Crandall and Carl Pomerance, P*rime Numbers: a Computational Perspective,* Springer, New York, 2001

6. Philip J. Davis, *Mathematics and Common Sense: A Case of Creative Tension,* A.K. Peters, Natick, MA, 2006.

7. Stephen R. Finch, *Mathematical Constants*, Cambridge University Press, Cambridge, 2003.

8. Marius Giaguinto, *Visual Thinking in Mathematics,* Oxford University, Oxford, 2007.

9. Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics*, Addison-Wesley, Boston, 1994.

10. Bonnie Gold and Roger Simons (Eds.), *Proof and Other Dilemmas: Mathematics and Philosophy*, Mathematical Association of America, Washington, DC, in press, 2008.

11. Richard K. Guy, *Unsolved Problems in Number Theory*, Springer-Verlag, Heidelberg, 1994.

12. Reuben Hersh, *What is Mathematics Really?* Oxford University Press, Oxford, 1999.

13. J. Havil, *Gamma: Exploring Euler's Constant*, Princeton University Press, Princeton, NJ, 2003.

14. Steven G. Krantz, *The Proof Is in the Pudding: A Look at the Changing Nature of Mathematical Proof*, Springer, 2010.

15. Marko Petkovsek, and Herbert Wilf, and Doron Zeilberger, *A=B,* A.K. Peters, Natick, MA, 1996.

16. Nathalie Sinclair, David Pimm, and William Higginson (Eds.), *Mathematics and the Aesthetic. New Approaches to an Ancient Affinity*, CMS Books in Math, Springer-Verlag, New York, 2007.

17. J. M. Steele, *The Cauchy-Schwarz Master Class*, Mathematical Association of America, Washington, DC, 2004.

18. Karl R. Stromberg, *An Introduction to Classical Real Analysis*, Wadsworth, Belmont, CA, 1981.

19. Richard P. Stanley, *Enumerative Combinatorics*, Volumes 1 and 2, Cambridge University Press, New York, 1999.

20. Terence Tao, *Solving Mathematical Problems*, Oxford University Press, New York, 2006.

21. Nico M. Temme, *Special Functions, an Introduction to the Classical Functions of Mathematical Physics*, John Wiley, New York, 1996.

22. Fernando R. Villegas, *Experimental Number Theory*, Oxford University Press, New York, 2007.

Finally many useful links are maintained by the authors of *Mathematics by Experiment* [2] at `www.experimentalmath.info`.

Jonathan Borwein and Matthew Skerritt, July 10th, 2009.

Contents

Conventions and Notation

Maple

Maple provides a wealth of different options for its interface, and mechanisms for entering commands. It is not within the scope of this document to deal with all of them. For the entirety of this document, we present *Maple* input as if it is entered in *Worksheet* mode with *2D Math* as both the input and output display settings. All three of these settings may be changed to be defaults (or not, of course) through *Maple*'s preferences.

This input scheme has the advantage that it looks a lot more like "regular" mathematics, making *Maple* worksheets easier to read than the historic text-only input scheme used in earlier versions. Using a worksheet mode instead of document mode also makes input, output and explanatory text generally much clearer to discern and is, in this author's opinion, superior when working on mathematical problems—although the document mode is probably preferred when trying to write results into a more human readable format.

Maple examples in this document are formatted to look like they would in a *Maple* worksheet using the above assumptions, and looks like the following:

```
> Input; Input
  Input
                                  Output
                                  Output
                                 Output
Warnings and Information
Errors
```

Each input—and its associated output, warnings and errors—are enclosed by a giant bracket (a "[", sometimes erroneously referred to as a "square bracket"), with a red > symbol acting as an input prompt. Multiple commands may be input together at the same prompt, and the input may even

Operation	Keys	Example	Example Input
Multiplication	`*`	$a \cdot b$	`a*b`
Powers	`^`	a^b	`a^b`
Subscripts	`_`	a_b	`a_b`
Fractions	`/`	$\frac{a}{b}$	`a/b`
Not Equal	`<>`	$a \neq b$	`a<>b`
Function	`->`	$a \rightarrow b$	`a->b`

Table 1: *Maple* Input Keystrokes

be spread over multiple lines. Input is coloured black and is left aligned. output is blue, and centred, warnings are blue in a monospaced "typewriter" font and left aligned, and finally errors are red in a monospaced "typewriter" font. This is almost identical to *Maple* (given the above assumptions), with the only difference that *Maple*'s errors are more of a purple-like colour. For the sakes of simplicity (and not too many colours) we have adopted red for this document.

It is, unfortunately, not immediately obvious which keystrokes produce which input effects. As such we include Table 1 which shows the non-obvious keystrokes. Note that *Maple* will automatically format the text as mathematics for you at the moment it is typed. *Maple* will also attempt to make sensible decisions, based around order of operations, as to which parts of the input will be effected. If *Maple* makes an incorrect decision, then parentheses (`(` and `)`) are needed to make the expression unambiguous.

Mathematics

Chapter 1

Number Theory

In this chapter you will learn the basics of the use of *Maple*, illustrated by fairly simple examples mostly involving integers. For this chapter you need to know what a sequence is, an infinite sum, summation notation, what a function is, and what a polynomial is. By the end of the chapter you should be comfortable using *Maple* for moderately complex tasks, and should be ready to learn new commands for doing specific mathematics—calculus or linear algebra, for example.

1.1 Introduction to *Maple*

Before we can set about exploring mathematics with *Maple* we need to know how to input basic commands into it. This section will introduce *Maple* and its most basic commands.

1.1.1 Inputting basic *Maple* expressions

At it's absolute most basic, *Maple* can be used as sort of overblown pocket calculator. We give it an expression to calculate it, and *Maple* performs the calculation.

```
> 1 + 2
```

$$3$$

```
> 2 · 3^5 + 12 − 2
```

$$496$$

or even more complicated statements involving factorials, trigonometric functions, and a lot more besides

```
> (sin(Pi/2) + 12! · sqrt(12)) / exp(4)
```

$$\frac{1 + 958003200\sqrt{3}}{e^4}$$

Notice that in this last example that *Maple* didn't provide a decimal number as the answer to the input. This rather nicely illustrates a key difference between *Maple* and your pocket calculator. *Maple* is a Computer Algebra System (CAS), and performs it's calculations as exactly as possible. When no exact number occurs, *Maple* provides an exact expression. So e^4 and $\sqrt{3}$ are exact values, whereas 54.59815003 and 1.732050808 (respectfully) are decimal approximations. *Maple* will give us the exact value unless we specifically ask it otherwise. To to this we either use the **evalf** command, or put a decimal point next to a constant.

```
> evalf((sin(Pi/2) + 12! · sqrt(12)) / exp(4))
```

$$3.039132676\ 10^7$$

```
> (sin(Pi/2) + 12! · sqrt(12.)) / exp(4)
```

$$\frac{1.659310217\ 10^9}{e^4}$$

Note in this second example the period after the 12 in the square root, which caused the numerator (but not the denominator) to be evaluated numerically. If in doubt, it's usually easier to just use **evalf**.

Since *Maple* is a CAS, we ought to be expect it can do some basic algebra. In point of fact it can, a good start to which is to work with basic polynomials.

```
> 3x^2 + 2x^3 + 3
```

$$3x^2 + 2x^3 + 3$$

```
> 4x^2 + 9x^2
```

$$13x^2$$

Notice that *Maple* automatically adds the like terms together.

```
> 3x^4 · 3y^2
```

$$9x^4y^2$$

In point of fact, *Maple* considers any word it does not otherwise know to just be an algebraic variable. We could use the strings "alice" and "bob" in place of x and y and *Maple* will treat them just like any other algebraic variable.

```
> alice + bob; % + 2; % + 5
```

$$alice + bob$$
$$alice + bob + 2$$
$$alice + bob + 7$$

Note for this last example there are a couple of things that we have not seen before. Firstly we have input 3 commands on the one line. Each command is ended by a semicolon (;), which tells *Maple* where a command ends. Older versions of *Maple* required a semicolon after each and every command - even if there was only a single command on a line - but version 12 only requires them between multiple commands. Secondly there is the special character %. This character is used to refer to the value of the most recent calculation. The % is a very useful tool, but be careful when using it, as the most recent calculation performed may not be on the same, or even the previous line of your input (see problem 3).

1.1.2 Variables

In addition to providing an "unkonwn" quantity for working with algebra, *Maple* variables (any string of characters that *Maple* doesn't know to be something else) can also have values assigned to them. There are two primary reasons to do this. The first is to give a name to an expression you want to use later, the other is to store a value that might change. Really, these are two sides to the same coin. To assign a value to a variable, we use the assignment operator (**:=**).

```
> A := 2
```

$$A := 2$$

Note that the assignment operator is often used in mathematics to denote a definition. This is no less true in *Maple*. We could happily view assigning a value to a variable name as defining the variable name to be that value.

Once we have assigned a value to a variable, when we use the variable name, *Maple* automatically uses its value for the computation.

```
> A; A + 2; A^10
```

$$2$$
$$4$$
$$1024$$

```
> Poly := 3*x^3 + 2*x^2 + 3*x; Poly + 3*x^2 + x
```

$$Poly := 3x^3 + 2x^2 + 3x$$
$$3x^3 + 5x^2 + 4x$$

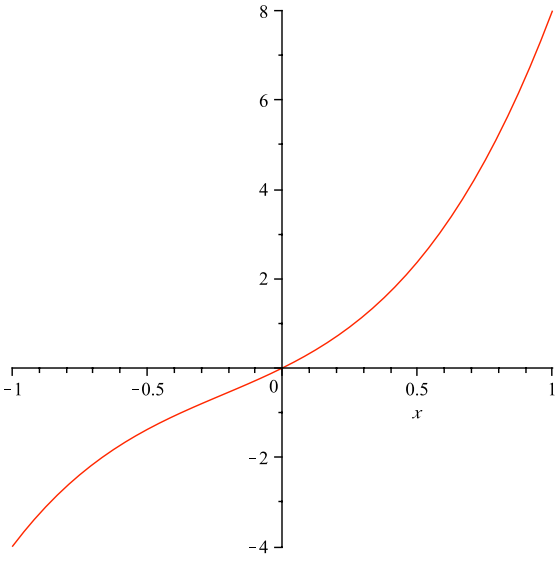
We can, if we wish, have *Maple* perform a calculation, and store the result of that calculation in the variable.

```
> value := 4 · 12 + 13^5
```

$$value := 371341$$

We can even use variable names as input to functions, and *Maple* still uses the value of the variable. We'll look at exactly what a function is in the next subsection, but for now notice that in the example below, *Maple* has plotted the cubic $3x^3 + 2x^2 + 3x$ to which we gave the name *Poly* in an earlier example above.

```
> plot(Poly, x = -1..1)
```



We may assign any valid *Maple* input to a variable, and *Maple* will always

use the value of the variable where we use its name. This can be very useful, if occasionally confusing.

So far we have used variables to give a name to something we want to do something with later on. However, as mentioned above, sometimes we want to use a variable as a storage box for values that can and will change. This is really no different to just reassigning a variable name to a different expression.

```
> a := 2; 2 · a
                                  a := 2
                                    4
> a := 4; 2 · a
                                  a := 4
                                    8
```

Such a technique, however, lets us use a variable to store intermediate results of a calculation in progress.

```
> total := 0
                                total := 0
> total := total + 12
                                total := 12
> total := total + 11
                                total := 23
```

Note here that not only are we reassigning the value of the variable named *total* but we are also using it's current value to calculate its new value. The line $total := total + 12$ means, in English, something along the line of "set the value of total to be its current value plus 12". What *Maple* actually does, is it evaluates the right hand side of the definition (which in this case is $total + 12$) and assigns the result of that calculation back to the variable name on the left hand side. These need not be the same variable.

```
> subtotal1 := 12; subtotal2 := 23
                              subtotal1 := 12
                              subtotal2 := 23
> total := subtotal1 + subtotal2
                                total := 35
```

Is perfectly valid, and is really just another case of the basic assignment of

a variable we saw at the beginning of the subsection.

1.1.3 Functions

Having dealt with basic input, and variables, we now move on to another key *Maple* concept, that of functions. A function, simplistically speaking, is just something that takes an input (perhaps several inputs) and produces output. We have already seen a couple of functions in our examples so far such as **sin** and **evalf**. Functions are written in the form of $name(input, input, \ldots)$ where $name$ is the name of the function, and $input, input, \ldots$ is a comma separates list of the inputs. To start with we will deal mostly with functions that take a single input, and should look very similar to the functions you will have seen in first year calculus.

```
> ifactor(1573)
```

$$(11)^2\ (13)$$

```
> factor(x^4 - 2*x^3 - 13*x^2 + 14*x + 24)
```

$$(x-2)(x-4)(x+3)(x+1)$$

```
> simplify((x^2 + x)/(x^3 + 2*x))
```

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

```
> is(1 < 2)
```

$$true$$

```
> lhs(A = B)
```

$$A$$

See Appendix A.2 for a list of commonly used basic functions. For more complicated functions, *Maple*'s own help files are always a good source of information. Every example so far has been a function that comes built-in to *Maple*. In the next section we will start creating our own functions, but for now we will look at an example of a function that takes multiple inputs.

```
> convert(0.1234, rational)
```

$$\frac{617}{5000}$$

The astute reader will also recognise that the plot function we saw earlier was also a function that took multiple inputs.

1.1.4 Sequences, Lists and Sets

In *Maple*, however, a sequence refers to a group of expressions separated by commas. For example

```
> 1, 2, 3; poly := x^3 + 3; poly, 5, bob
```

$$1, 2, 3$$
$$poly := x^3 + 3$$
$$x^3 + 3, 5, bob$$

are both *Maple* sequences. First year calculus students should have studied infinite sequences and series (which were just a more intricately constructed sequence), and the difference between a *Maple* sequence and a mathematical (infinite) sequence could potentially be confusing. Whilst *Maple* sequences could well be thought of as finite sequences, it might be easier to just recognise the fact that the two are different. *Maple* can handle the usual calculus sequences, but does so using the limit function, which will be dealt with in more depth in the Calculus chapter (Chapter 2).

```
> limit(1/k^2, k = infinity)
```

$$0$$

So with that clarification out of the way, let's get back to *Maple* sequences. For the remainder of this chapter a sequence will refer to a *Maple* sequence. Any valid *Maple* code may form an element of a sequence. The astute reader may have already noticed that the functions, above, that took multiple variables accepted their input as a sequence. For example the **plot** function took as it's input the sequence $poly, x = -1..1$ and the **convert** function had the sequence $0.1234, rational$ as it's input.

Since sequences are just another valid *Maple* expression, they may be stored in a variable. If we have two sequences and put them in a sequence, the result is one large sequence - not two nested sequences. This will become more clear when we see how lists and sets "nest" later in the subsection.

```
> A := a, b, c; B := 1, 2, 3; C := i, ii, iii; A, B, C
```

$$A := a, b, c$$
$$B := 1, 2, 3$$
$$C := i, ii, iii$$
$$a, b, c, 1, 2, 3, i, ii, iii$$

Sequences are can also be a very convenient way of assigning multiple vari-

ables in a single command. For instance, if we wished to assign $A := 1$, $B := 2$ and $C := 3$ then we could use

```
> A, B, C := 1, 2, 3; A; B; C;
```

$$A, B, C := 1, 2, 3$$
$$1$$
$$2$$
$$3$$

There is also a special sequence called the null sequence. This is a sequence with no elements, much like the empty set in set theory is the set with no elements. The null sequence in *Maple* is referenced by the name *NULL*. As such putting the null sequence in any sequence doesn't change the sequence at all.

```
> NULL, a, b, c; a, b, c, NULL; a, b, NULL, c
```

$$a, b, c$$
$$a, b, c$$
$$a, b, c$$

Using a null list allows a list to be built up by parts within a variable, a little like the *total* variable was in the previous subsection.

```
> S := NULL
```

$$S :=$$

```
> S := S, a, b, c
```

$$S := a, b, c$$

```
> S := S, d, e
```

$$S := a, b, c, d, e$$

If we want to produce a sequence that follows a fairly predictable pattern, we have a handy command, **seq**. To print out the first 10 squares we simply input

```
> seq(k^2, k = 1..10)
```

$$1, 4, 9, 16, 25, 36, 49, 64, 81, 100$$

or for something a little more complicated

```
> seq(3 · k^2 + k/2, k = 4..15)
```

$$50, \frac{155}{2}, 111, \frac{301}{2}, 196, \frac{495}{2}, 305, \frac{737}{2}, 438, \frac{1027}{2}, 595, \frac{1365}{2}$$

In general, to have *Maple* print out the sequence $\{x_n\}_{n=a}^{b}$ we use the command $seq(x_n, n = a..b)$. There is a shortcut that can be applied if we wish to repeat the same term multiple times - a thing which will be come desirable later on when we do some calculus. To do this we use the sequence operator (**$**).

> $x\$4$

$$x, x, x, x$$

This sequence operator can be used as a shortcut to the **seq** command, but be warned that it isn't quite as flexible as the **seq** command (see Examples 5 and 6).

If we have a sequence, we may wish to use only a subsequence of it, or perhaps only a single element. *Maple* allows this through indexing using the index operator [], or through subscripting in the graphical editor. Be warned, subscripting is a shorthand for the square bracket operator. If unsure, use square brackets.

> $S := seq(k^2, k = 1..10);$

$$S := 1, 4, 9, 16, 25, 36, 49, 64, 81, 100$$

> $S[3]; S_4;$

$$9$$
$$16$$

> $S[5..8]; S_{3..7}$

$$25, 36, 49, 64$$
$$9, 16, 25, 36, 49$$

> $S[..4]; S_{..3}; S[6..]; S_{7..}$

$$1, 4, 9, 16$$
$$1, 4, 9$$
$$36, 49, 64, 81, 100$$
$$49, 64, 81, 100$$

In the latter example, the "half ranges" $..n$ and $m..$ mean "the first n" and "the final m" elements respectively. Such notions make sense because we are indexing a sequence that contains only a finite number of elements. An alternate way of thinking of these ranges is to consider that *Maple* automatically inserts the beginning or the end index for the missing number as appropriate.

We now look at two related notions to sequences: lists and sets. Syntacti-

cally, a list or set is just a sequence enclosed in [] or {} respectively. Sets should be familiar to the reader, and are unsorted and ignore duplication as would be expected. Lists are ordered and allow duplications. Both lists and sets can be nested, which makes them distinct in behaviour from sequences. It is this distinction that makes lists and sets useful for removing ambiguity when trying to have a function recognise a sequence. See Example 14 for an example of this.

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

$$L := [1,1,2,2,3,3,4,4]$$
$$S := \{1,2,3,4\}$$

```
> [L,S]; {L,S}
```

$$[[1,1,2,2,3,3,4,4],\{1,2,3,4\}]$$
$$\{[1,1,2,2,3,3,4,4],\{1,2,3,4\}\}$$

Lists and sets may be indexed exactly as with a sequence, with the difference that if a range is indexed, then a list or set respectively is produced. Indexing a single element produces the element as with a sequence

```
> L[1], L[2], S[3], S[4]
```

$$1,1,2,2$$

```
> L[3..5], L[..3], S[3..], S[2..3]
```

$$[2,2,3],[1,1,2],\{3,4\},\{2,3\}$$

Observe that in the above two examples we have used a sequence to display the results of the calculations on a single line.

If we wish to know whether a list or a set contains a particular element, we may use the **in** command. This may be thought of as being the $\in$ operator, and is even printed as such in *Maple*.

```
> 1 in S
```

$$1 \in S$$

This command on it's own does nothing but write itself out again with the element symbol. In order to have *Maple* actually tell us whether or not $1 \in S$ (in the above case) then we need to use the **is** or **evalb** commands. We saw **is** in the functions subsection above. Both functions return a value of either *true* or *false*. In fact the function **evalb** is short for "evaluate as boolean" where a boolean value is either true or false.

is(? in ?) is more correct than **has** - so I'm avoiding the latter

```
> is(1 in S); evalb(9 in L)
                                  true
                                  false
```

And finally, the usual set operations of **union** and **intersect** work as expected with sets, and not at all with lists.

```
> {1, 2} union {2, 3, 4}; {1, 2} intersect {2, 3, 4}
                              {1, 2, 3, 4}
                                 {2}
```

1.2 Putting it together

In the previous section, we looked at inputting single commands into *Maple*. These may be thought of as building blocks. In this section we begin to put these building blocks together to produce more complicated calculations

1.2.1 Sums and Products

Having dealt with the basics of *Maple*, we can now move onto some mathematics a little more like we might see at the beginning of a first year maths course. *Maple* can handle sequences, and sums and products, both of the finite (naturally) and the infinite variety. We start with sequences.

Now suppose we want to add together the elements of some sequence. We can use an **add** or **sum** command. There are a couple of small differences between these two commands, but the most glaring difference is that **sum** is designed for infinite sums (sometimes known as series) and tries to be clever about the addition, whereas **add** simply adds together a bunch of terms, and so is generally a lot quicker than sum, but cannot handle series. Both the sum and add commands look very much like the seq command.

```
> add(k^2, k = 1..10)
                                  385
> sum(k^2, k = 1..10)
                                  385
```

The sum command comes in two forms. If the command is written with a lower case s, then it performs the summation as we have previously seen.

This form is known as an active sum. If the command is written with a capital S, then *Maple* simply writes the sum in the sigma notation. This form is known as an inert sum. In order to be able to ask *Maple* to perform the calculation of an inert sum, we have the **value** function. *Maple* will display an inert sum with a grey sigma, and an active sum (on the unusual occasions that it displays an active sum in sigma notation) with a blue sigma.

```
> Sum(k^2, k = 1..N)
```

$$\sum_{k=1}^{N} k^2$$

```
> factor(value(%))
```

$$\frac{1}{6}N(N+1)(2N+1)$$

As well as sums, *Maple* can also handle products. Much like add and sum, *Maple* has two functions to do products, one which simply performs a multiplication, and the other which is smarter but slower and designed for evaluating infinite products. These commands are **mul** and **product** respectively. Also like sum, the product command comes in an inert and active variant which is indicated by a capital P for the inert, and a lower case p for the active

```
> mul(k^2, k = 1..10)
```

$$13168189440000$$

```
> product(k^2, k = 1..10)
```

$$13168189440000$$

```
> Product(k^2, k = 1..N); value(%)
```

$$\prod_{k=1}^{N} k^2$$

$$\Gamma(N+1)^2$$

The result of the latter of these computations should make more sense if you know that $\Gamma(n) = (n-1)!$

1.2.2 Creating Functions

We have seen what a function is in the previous section. In addition to the built in functions, we may create our own. A function takes on the form *input* → *expression*, and is a perfectly valid *Maple* expression on it's own. The arrow is produced by typing the text `->`

```
> x -> 3*x^2 + 4*x - 2;
```

$$x \to 3x^2 + 4x - 2$$

The above function takes a single expression as input (which it calls x), and then performs the calculation $3x^2 + 4x - 2$. The name of the input variable is entirely arbitrary and could be any valid *Maple* variable name. In order to be able to actually use this function, we need to assign it to a variable name.

```
> f := x -> 3*x^2 + 4*x - 2;
```

$$f := x \to 3x^2 + 4x - 2$$

```
> f(2); f(4); f(A); f(Γ(N)); f([1,2])
```

$$18$$
$$62$$
$$3A^2 + 4A - 2$$
$$3\,\Gamma(N)^2 + 4\,\Gamma(N) - 2$$
$$3[1,2]^2 + 4[1,2] - 2$$

It is interesting to see in the last example above that even though it makes apparently no sense mathematically, *Maple* will, nonetheless, accept $[1, 2]$ as input to the function f and produce the output string as if $[1, 2]$ were a variable.

How about something that might be a bit more familiar. Recall from first year calculus that

$$\sum_{k=1}^{\infty} \frac{1}{k}$$

diverges, whereas

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

converges. This is a good reminder that the divergence test is aptly named and does not guarantee convergence, since the sequence has limit $\lim_{k\to\infty} \frac{1}{k} = 0$ but the series diverges. To verify the convergence of the $\frac{1}{k^2}$ series, we can have *Maple* calculate the ratio test.

$$> \ a := N \to \frac{1}{k^2}; limit(a(N), N = infinity); \text{abs}\left(\frac{a(N+1)}{a(N)}\right)$$

$$a := N \to \frac{1}{k^2}$$

$$0$$

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

It is clear that the convergence test passes (ie that $\left|\frac{a(N+1)}{a(N)}\right| < 1$), however we can ask *Maple* to verify this. Observe that the fraction is undefined for $N = -1$, but that this isn't a concern for us since we're only interested in natural values of N. However, we may need to be a little specific in what we ask *Maple*

$$> \ is(\% < 1)$$

$$FAIL$$

$$> \ is(\%\% < 1) \ \text{assuming} \ N :: posint$$

$$true$$

You should read $N :: posint$ as "N is a positive integer". We could, in this particular case (but not always), have used $N \geq 0$ in place of $N :: posint$ and would have obtained the same answer.

Now that we have verified that the series, indeed, converges we can see what it converges to. Whilst we could just ask *Maple*, it is probably better that we raise our confidence in the answer by doing some maths ourselves as well - plus we can recall some of our first year mathematics at the same time. Recall, then, that an infinite sum is a limit of parital sums. That is

$$\sum_{k=1}^{\infty} a(k) = \lim_{N\to\infty} \sum_{k=1}^{N} a(k)$$

We will create the partial sum of our series $\frac{1}{k^2}$ as a function in *Maple*.

$$> \ f := N \to sum\left(\frac{1}{k^2}, k = 1..N\right)$$

$$f := N \to \sum_{k=1}^{N} \frac{1}{N^2}$$

So we have f as a function of N where N is the number of terms to sum.

Let us now see how this sum behaves with increasing values

```
> f(1), f(2), f(10), f(100), f(10000)
```

$$1, \frac{5}{4}, \frac{1968329}{1270080}, \frac{1589508694133037873112297928517553859702383498543709859889432834803818131090369901}{9721861444343810305896579766726231441619755839957462417827203547055179861652480000}, -\Psi(1, 10$$

$$\frac{1}{6}\pi^2$$

Well that wasn't particularly helpful, although the term for $f(10000)$ looks promising. Let's try asking for decimal answers instead.

```
> seq(evalf(f(10^i)), i = 1..6)
```

$$1.549767731, 1.634983900, 1.643934568, 1.644834073, 1.644924068, 1.644933068$$

We needed to be careful here, as our function f consisted of a sum which used the dummy variable k, and so if we had also used k for the dummy variable in the **seq** command they would have conflicted. See exercise 13. We can also see here the clear convergence of the series to 1.644. . . - remembering that the sequence we asked for was $f(1), f(10), f(100), f(1000), f(10000), f(1000000)$. We may as well ask *Maple* if it can give us an answer for the limit.

```
> limit(f(n), n = infinity); evalf(%)
```

$$\frac{1}{6}\pi^2$$

$$1.644934068$$

And there we have it.

1.2.3 Loops and Decisions

Until now if we wanted to perform something several times, we either typed it in multiple times at the command prompt, or we constructed a sequence. Sometimes these options aren't satisfactory. Let us revisit our example of the series $\sum \frac{1}{k^2}$. Earlier we used a sequence to print out the sequence

$$\left\{ \sum_{k=1}^{10^N} \frac{1}{k^2} \right\}_{N=1}^{6}$$

which quite conveniently demonstrated the convergence of the series. The sequence was quite easy to read, however suppose we wanted to see more

values of the sequence. Let's look at the values of the partial sums for values of N as the first 20 powers of 2. That is $N = 2, 4, 8, 16, \ldots, 1048576$.

```
> seq(evalf(f(2^i)), i = 1..20)
    1.250000000, 1.423611111, 1.527422052, 1.584346533,
    1.614167263, 1.629430501, 1.637152005, 1.641035436,
    1.642982848, 1.643957982, 1.644445906, 1.644689957,
    1.644812005, 1.644873035, 1.644903551, 1.644918809,
    1.644926439, 1.644930253, 1.644932161, 1.644933114
```

That's a bit of a mess, but not completely unreadable. Now, we would like to see which values of N produce which of those outputs. We can work it out by counting from the left and working out the power of 2, but it would be nicer to see it. We'll tell *Maple* to print $f(N) =$ before each answer. In order to do this without having *Maple* evaluate $f(N)$ as a function, we enclose the f in single quotes, which tells *Maple* to treat it as symbol only, and not to evaluate it.

```
> seq('f'(2^i) = evalf(f(2^i)), i = 1..20)
f(2) = 1.250000000, f(4) = 1.423611111, f(8) = 1.527422052,
     f(16) = 1.584346533, f(32) = 1.614167263, f(64) =
  1.629430501, f(128) = 1.637152005, f(256) = 1.641035436,
   f(512) = 1.642982848, f(1024) = 1.643957982, f(2048) =
 1.644445906, f(4096) = 1.644689957, f(8192) = 1.644812005,
 f(16384) = 1.644873035, f(32768) = 1.644903551, f(65536) =
     1.644918809, f(131072) = 1.644926439, f(262144) =
     1.644930253, f(524288) = 1.644932161, f(1048576) =
                          1.644933114
```

Ugh, now that's really a mess. What would be better would be if we could get each equality on it's own line, instead of having it as one big sequence. Now, instead of just typing out all 20 expressions one after the other, *Maple* provides a mechanism for repeating calculations, called a loop.

> **for** i **from** 1 **to** 20 **do**' $f'(2^i) = evalf(f(2^i))$ **od**

$$f(2) = 1.250000000$$
$$f(4) = 1.423611111$$
$$f(8) = 1.527422052$$
$$f(16) = 1.584346533$$
$$f(32) = 1.614167263$$
$$f(64) = 1.629430501$$
$$f(128) = 1.637152005$$
$$f(256) = 1.641035436$$
$$f(512) = 1.642982848$$
$$f(1024) = 1.643957982$$
$$f(2048) = 1.644445906$$
$$f(4096) = 1.644689957$$
$$f(8192) = 1.644812005$$
$$f(16384) = 1.644873035$$
$$f(32768) = 1.644903551$$
$$f(65536) = 1.644918809$$
$$f(131072) = 1.644926439$$
$$f(262144) = 1.644930253$$
$$f(524288) = 1.644932161$$
$$f(1048576) = 1.644933114$$

Now this is much easier to read. *Maple* has calculated the expression $f(2^i) = evalf(f(2^i))$ for us, and has done so 20 times, each time with the value of i one bigger than the previous time. After each calculation it has output the result of the calculation just as it would have if we had entered it manually at the command prompt.

The **for** command is also useful if we have a small group of calculations which we wish repeated multiple times. For example, we could calculate the first twelve Fibonacci numbers using a **for** loop as follows:

```
> a := 1 : b := 1 : L := a, b :
  for i from 1 to 10 do
    c := a + b;
    L := L, c;
    a := b; b := c;
  od
```

$$c := 2$$
$$L := 1, 1, 2$$
$$a := 1$$
$$b := 2$$
$$c := 3$$
$$L := 1, 1, 2, 3$$
$$a := 2$$
$$b := 3$$

⟨Remaining output omitted for brevity⟩

```
> 'F' = F
```

$$F = (1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144)$$

Notice how we have broken the **for** command up over multiple lines. We could have kept it all on one line, but it would have been a little tougher to read. We also indented the commands that were to be performed within the loop to make it a little more obvious that they were inside the loop (which was not indented). Also notice that we put a semicolon at the end of each of the 3 commands we wished to repeat within the loop. Doing this is no different to the times earlier when we put semicolons between multiple commands on the one line.

There are a lot of intricacies to loops, and the reader should refer to *Maple*'s help files and exercises 15.

Suppose now that we want *Maple* to perform a calculation, but only on a particular condition. We might wish to calculate the square of a number, but only if that number is smaller than something else. We would tell *Maple* to do this as follows

This example feels completely asinine—I'd like a better one

```
> if a < b then a^2 fi
  Error, cannot determine if this expression is true or
                        false:  a < b
```

The above code should be read as "If a is smaller than b then calculate a^2". However, because *Maple* doesn't know what a and b are, then it cannot work out if $a < b$, and has told us so. If we give values to a and b then we should get some better results.

```
> a, b := 2, 3; if a < b then a^2 fi
```

$$a := 2$$
$$b := 3$$
$$4$$

```
> a, b := 3, 2; if a < b then a^2 fi
```

$$a := 3$$
$$b := 2$$

In the first example, it was certainly the case that $a < b$ and so *Maple* went and calculated $a^2 = 4$, but in the second case it was not the case that $a < b$ and so *Maple* did not do anything. It is usually quite rare that we use a decision manually when using *Maple*, where we can make these decisions for ourselves. It is much more common that a decision would be part of a function or a loop where we do not have the luxury of being certain what values our variables contain.

In the previous example, our decision did not do anything if our criterion was not true. Perhaps, however, we want to perform one calculation if the criterion is true, and another if it is not true. A good example of this is the absolute value function. Recall that

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{if } x < 0 \end{cases}$$

There is no convenient algebraic expression to perform this calculation, we must use a decision. In order to do this in *Maple* we use an **if** statement as we did above, with an extra parameter **else**. We will write a quick loop to print out the absolute value of some numbers for us.

```
> for x from -3 to 3 do
    if x >= 0 then print(x) else print(-x) fi
  od
```

$$3$$
$$2$$
$$1$$
$$0$$
$$1$$
$$2$$
$$3$$

We were forced to use the **print** function here, which just tells *Maple* to output an expression just as it would if we typed that expression in ourselves manually. This was required here because we had an **if** statement buried inside of a **for** statement. We will look more at this sort of thing later on, but for the time being be aware that once these things become buried inside each other (often referred to as "nesting") *Maple* will not produce output unless we specifically ask for it.

1.2.4 Procedures

When we created our own functions earlier, we used the very handy arrow notation ($\rightarrow$). Using this method is very convenient, and allows us a lot of power when using *Maple*, but it doesn't take long to find that it does have some limitations. How do we create a function that requires more than one *Maple* expression to calculate it's output? In fact, we cannot even use the **if** statement to make decisions in a function using this method

```
> abval := x -> if x >= 0 then print(x) else print(-x) fi
                    Error, invalid arrow procedure
```

As it happens, the functions we have created up till now with the $\rightarrow$ operator have been special cases of a more general *Maple* construction known as a procedure. A procedure is, as it's name suggests, a series of steps to be followed and behaves like the functions we have used and created so far in that it takes input and produces output. A procedure, however, allows multiple calculations to be performed as part of its processing, much like our loops and decisions above did.

We can see this if we use the **lprint** command with our already familiar arrow notation functions. We'll use a simple function that calculates the square of the input.

```
> x -> x^2; lprint(%)
                                  x -> x^2
        proc (x) options operator, arrow; x^2 end proc
```

A procedure, at its most basic, takes the form **proc**(*input*) *commands* **end proc**. Although we may use **end** as a shorthand version of **end proc**. The commands, like any *Maple* commands, must have semicolons between them if there is more than one. If we ignore the options in the example above, we see precisely that form. The last performed calculation in a procedure (when it executes) will be that which *Maple* outputs after the function call.

In order to create the absolute value function which we could not do earlier, we can use a procedure as follows.

```
> abval := proc(x)
    if x >= 0 then x else -x fi
  end :
```

```
> abval(-2), abval(5); abval(-0.22), abval(3/2); abval(x)
```

$$2, 5, 0.22, \frac{3}{2}$$

```
Error, (in abval) cannot determine if this expression is
                  true or false:  0 <= x
```

Our function works fairly well, although it doesn't cope well with unassigned variables. If we experiment a little we will probably find other things it doesn't handle very well. Nonetheless it does calculate the absolute value for whole numbers, and nicely demonstrates a simple procedure. We will point out here that *Maple* does actually provide a built in function for this named **abs** which is much more robust than the above procedure and also extends as far as calculating the modulus of a complex number.

```
> abs(-Pi), abs(1+2*I), abs(x), abs(z)
```

$$\pi, \sqrt{5}, |x|, |z|$$

So far all of the procedures we have seen or written so far have been procedures that could be more easily written using the arrow notation. So now we will look at something more complicated that cannot so easily be implemented using arrow notation. We will return to the Fibonacci numbers. We have seen in Section 1.2.3 a **for** loop which calculated the first twelve Fibonacci numbers. We will now create a procedure to calculate the n^{th} Fibonacci numbers.

```
> fib := proc(n)
    local a, b, c, i;
    description "Calculate the nth Fibonacci number"
    if n = 1 or n = 2 then
      1;
    else
      a, b := 1, 1;
      for i from 2 to n do
        c := a + b; a := b; b := c
      od;
      c;
    fi;
  end
```

```
> seq(fib(k), k = 1..10)
                    1, 1, 2, 3, 5, 8, 13, 21, 34, 55
```

The procedure looks complicated, but is really fairly simple. It takes a single input variable n, which is the position of the Fibonacci number we wish to calculate. If we ask for the 1^{st} or 2^{nd} Fibonacci number ($n = 1$ or $n = 2$) then the procedure returns 1. Note that in this case the final calculation to be performed is simply the calculation 1;. If, however, we asked for a larger numbered Fibonacci number, then the procedure executes a for loop like the one we created in Section 1.2.3. Here the very final calculation performed will be c;—which must be the n^{th} Fibonacci number because of the **for** loop that will be performed beforehand.

We have used two additional pieces of a procedure in this example. The **local** declaration tells *Maple* that the variables a, b, c and i are variables that are local to the procedure. In fact, if any of those variables are declared elsewhere, *Maple* will ignore that declaration inside of the procedure, and will not overwrite the variable. See Exercise **??** for an exploration and clarification of this. We also used an optional part of a procedure which is the **description** which is, as its name should suggest, just a description of what the procedure does. We may see this in maple with the **Describe** command.

```
> Describe(fib)
          # Calculate the nth Fibonacci number
                      fib( n )
```

Finally, we will make two observations and refine our Fibonacci number generating procedure one last time. The curious reader might wonder what

happens if we put a silly value into our *fib* procedure, like -1 or π.

```
> fib(-1); fib(Pi)
                                   c
Error, (in fib) final value in for loop must be numeric or
                          character
```

Neither of these are very satisfactory. Also, the reader may recall that the usual definition of the Fibonacci numbers is recursive. If $f(n)$ is the n^{th} Fibonacci number, then $f(n) = f(n-1) + f(n-2)$. One may wonder whether we may write our procedure in this more natural way. We will address both of these observations with a new procedure *fib2*.

```
> fib2 := proc(n :: posint)
    description "Calculate the nth Fibonacci number"
    if n = 1 or n = 2 then
      1;
    else
      fib2(n - 1) + fib2(n - 2)
    fi
  end

> seq(fib2(n), n = 1..10); fib2(-1); fib2(Pi)
                    1, 1, 2, 3, 5, 8, 13, 21, 34, 55
Error, invalid input:  fib2 expects its 1st argument, n, to
            be of type posint, but received -1
Error, invalid input:  fib2 expects its 1st argument, n, to
           be of type posint, but received Pi
```

Firstly we have added the text *::posint* to the variable name. This is our way of telling *Maple* that we want the variable n in the procedure to be a positive integer. Recall that we used this same notation with the **assuming** keyword in Section 1.2.2. See the help file (**?types**) for more information on the different types that a variable might be. If we give *fib2* anything other than a positive integer for the input parameter, we are given an error message as seen in the example, above. This error is much more clear than the one that the *fib* procedure gave - when it gave one at all.

Secondly, if the *fib2* procedure is given a value of n other than 1 or 2, then it calculates $fib2(n-1)+fib2(n-2)$. This means that the procedure performs itself again, but with different input parameters, as part of its calculation. This is called *resursion*, and a procedure that uses this technique is called a *recursive* procedure. Recall that the Fibonacci numbers are an example

of a recurrence relation. See Section 1.3.3 for more on recurrence relations. Recursive procedures like this one have limits on the number times they may refer back to themselves in a single execution, but are much clearer to read and understand.

We should point out now that, as we might have expected, *Maple* does provide an inbuilt function for calculating the Fibonacci numbers. The function is the **fibonacci** function from the *combinat* package, and can be called directly using the command **combinat[fibonacci](n)** (where n is a positive integer, of course). The reader is encouraged to look at other functions in the *combinat* package.

There are many other complexities to procedures, however what has been described here—as well as the relevant exercises—should be sufficient for most needs. Be sure to examine *Maple*'s help files on procedures for more information.

1.2.5 Nesting

We have looked at loops and decisions in the previous. Inside a loop, or a decision, we may have any valid *Maple* code, and even multiple valid *Maple* commands. One may ask if a loop may be placed inside of a loop, or if a decision may be placed inside of a decision, and the answer is yes they can. Doing such a thing is called *nesting*.

We will start with loops. Observe the following loop.

```
> for i from 1 to 3 do
    for j from 1 to 3 do
      print(i, j)
    od
  od
```

$$1, 1$$
$$1, 2$$
$$1, 3$$
$$2, 1$$
$$2, 2$$
$$2, 3$$
$$3, 1$$
$$3, 2$$
$$3, 3$$

Just as with a regular loop, the counting variable, i in this case, takes the

values $1, 2, 3$ and each time the contents of the loop are executed. However, in this case the contents of the loop is another for loop, and so for each of the values i takes on, the second loop executes and the j counting variable takes on the values $1, 2, 3$ and the second loop prints the pair i, j of the counting variables at that point in time. The result is seen above. This is very much like a double sum.

We need not limit ourselves to two nested loops either. We may do three, or–indeed–any number. For example

```
> for i from 1 to 2 do
    for j from 1 to 2 do
      for k from 1 to 2 do
        print(i, j, k)
      od
    od
  od
```

$1, 1, 1$
$1, 1, 2$
$1, 2, 1$
$1, 2, 2$
$2, 1, 1$
$2, 1, 2$
$2, 2, 1$
$2, 2, 2$

Just as loops may be nested, so may decisions. Suppose we want to perform different actions depending on whether a number, n say, is even or odd as well as if it is positive or negative. We may handle this with a nested decision. For example, we might have the following *Maple* code.

```
> if n ≥ 0 then
    if n mod 2 = 0 then
      f(n)
    else
      g(n)
    fi
  else
    if n mod 2 = 0 then
      h(n)
    else
      i(n)
    fi
  fi
```

If n is positive (or zero) the code then checks to see if n is even or odd and acts accordingly. Similarly if n is negative, the code then checks to see if n is even or odd and acts accordingly, but differently to if n was positive.

1.3 Enough code, already. Show me some maths!

We have spent the previous two sections learning *Maple* code for its own sake. By now we hope to be, at the very least, passably familiar with instructing *Maple* to perform calculations. The point of using *Maple* however is to allow us to perform and explore mathematics. In this section we shall do just that, and put the techniques and code learned in the previous sections to use with actual mathematics.

1.3.1 Induction

In general, the work we do in *Maple* will not constitute a mathematical proof. *Maple* is more a tool for exploring mathematics that will often lead to greater understanding and perhaps the production of a proof by the usual (non computer-related) means. However we may use it to perform some basic induction for us.

Recall the formula for the sum of the first n squares is

$$\sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}$$

Suppose we only want to add the first n even squares. It's not hard to manipulate the sum to an expression involving the above sum.

$$\sum_{k=1}^{n}(2k)^2 = \sum_{k=1}^{n} 4k^2 = 4\sum_{k=1}^{n} k^2 = \frac{2n(n+1)(2n+1)}{3}$$

and we can certainly check *Maple* to see if it provides the same answer.

```
> sum((2k)^2, k = 1..n)
```

$$\frac{4}{3}(n+1)^3 - 2(n+1)^2 + \frac{2}{3}n + \frac{2}{3}$$

```
> factor(%)
```

$$\frac{2}{3}n(n+1)(2n+1)$$

That is all well and good, but it's not induction. How about we try the first n odd squares.

$$\sum_{k=1}^{n}(2k-1)^2$$

We could follow the same approach we did above, and expand the summand into a quadratic, and apply the formulae we already know for $\sum_{k=1}^{n} k^2$, $\sum_{k=1}^{n} k$ and $\sum_{k=1}^{n} c$ respectively, and indeed a good number of first year students would probably prefer this to induction. We will not do that this time, however. Instead we will ask *Maple* what it thinks the answer is, and verify it using induction (also within *Maple*).

To begin with, we set up a function, to more easily reuse the calculations

```
> f := N -> sum((2k-1)^2, k = 1..N);
```

$$N \to \sum_{k=1}^{N}(2k-1)^2$$

Now we see what *Maple* thinks the function should look like for arbitrary N.

```
> g := factor(f(N));
```

$$g := \frac{1}{3}N(2N-1)(2N+1)$$

We have factorised the answer here to keep it neat, and we have assigned it to the variable g for future reference. So now we have a candidate for a formula. We can be pretty confident that it is correct since *Maple* provided

it, but it never hurts to check - especially since we don't know yet exactly how *Maple*'s sum function handles an indefinite sum like that. This we now do. First we begin with a basis case.

```
> f(1) = subs(N = 1, g)
```

$$1 = 1$$

Note here the use of the substitution command **subs**. This command substitutes the value 1 for N in the expression g. Note here, we could have simply asked *Maple is* $(f(1) = subs(N = 1, g))$, but the answer of true or false is sometimes unreliable, and it is best to see the statement written out in its entirety before asking for an **is** or **evalb**. It is clear from the output of *Maple* that the formula is correct for the basis case of $N = 1$.

Now we may complete the induction. Assuming that

$$\sum_{k=1}^{N}(2k-1)^2 = \frac{1}{3}N(2N-1)(2N+1)$$

we want to show that

$$\sum_{k=1}^{N+1}(2k-1)^2 = \frac{1}{3}(N+1)(2N+1)(2N+3)$$

which we will do by showing that

$$\left(\sum_{k=1}^{N}(2k-1)^2\right) + 2N + 1 - \frac{1}{3}(N+1)(2N+1)(2N+3) = 0$$

```
> f(N) + (2(N + 1) − 1)^2 − subs(N = N + 1, g)
```

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

```
> simplify(%)
```

$$0$$

And we're done. We may not be sure how *Maple* handles an indefinite sum, but we can be extremely confident with it's ability to do basic algebra. If we want to completely remove the question of the behaviour of *Maple*'s **sum** function out of the equation - as it is technically in question here - we can do the same thing with g alone.

However, the substitution command is a little long, and we probably don't really want to constantly be typing it in whenever we want to assign a value

to N. It would be much easier if g was a function itself. *Maple* provides a handy method for taking an arbitrary expression and turning it into a function. It is called **unapply**. We will use it now to make g into a function of N, rather than just a fixed expression as it is now.

```
> g := unapply(g, N)
```

$$g := N \to \frac{1}{3}N(2N-1)(2N+1)$$

And now we can perform the inductive step using the functional notation for g

```
> g(N) + (2(N+1) - 1)^2 - g(N+1); simplify(%)
```

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

$$0$$

Be careful here. It might look awfully tempting to try something like the following as the inductive step.

```
> f(N+1) - subs(N = N+1, g)
```

$$\frac{11}{3}N + \frac{19}{3} - 4(N+2)^2 + \frac{4}{3}(N+2)^3 - \frac{1}{3}(N+1)(2N+1)(2N+3)$$

```
> simplify(%)
```

$$0$$

However, this is *not* induction, because we have not used the assumption that $f(N) = g$ in order to show that $f(N+1)$ has the required form. All we have done in this case is verify that *Maple*'s **sum** command produces the desired formula for inputs of N and $N+1$, but we have not proved the relation using induction.

We will do one more induction with *Maple*. This time we will verify a well known one, and because we won't be explaining every step of the way, the process will be much shorter. We will verify the formula

$$\sum_{k=1}^{N} k^3 = \frac{N^2(N+1)^2}{4}$$

We can verify in our heads that the relation true when $N = 1$ so we may eschew the induction step in *Maple*, leaving us just the inductive step itself

$$> \; g := N \to \frac{N^2(N+1)^2}{4}; g(N) + (N+1)^3 - g(N+1); simplify(\%)$$

$$g := N \to \frac{1}{4}N^2(N+1)^2$$

$$\frac{1}{4}N^2(N+1)^2 + (N+1)^3 - \frac{1}{4}(N+1)^2(N+2)^2$$

$$0$$

1.3.2 Continued Fractions

Real numbers may, as we should already be aware, be expressed by decimals that either terminate or continue (countably) infinitely. The latter category may further partitioned into recurring and non-recurring infinite decimal representations. Rational numbers may be written as terminating or recurring decimals, and irrational numbers have an infinite non recurring decimal representation.

Another way to represent real numbers is with so called *continued fractions*. A continued fraction is a, potentially infinite, fraction of the form

$$a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cfrac{b_2}{\ddots}}} \text{ where } a_i, b_i \in \mathbb{Z}$$

However, for the purposes of this section, we will be concentrating on simple continued fractions, where the b_i are all 1

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\ddots}}}$$

often abbreviated to just $[a_0; a_1, a_2, \ldots]$

The procedure for calculating the continued fraction of a number is quite straightforward. Let $x \in \mathbb{R}$. We let $x_0 = x$ and separate the integer part from the fractional part. That is, we take $a_0 = \lfloor x_0 \rfloor$,

$$x = a_0 + (x_0 - a_0) = a_0 + \frac{1}{\left(\dfrac{1}{x_0 - a_0}\right)}$$

We now invert the fractional part for $x_1 = (x_0 - a_0)^{-1}$ and set $a_1 = \lfloor x_1 \rfloor$, yeilding

$$x = a_0 + \frac{1}{a_1 + (x_1 - a_1)} = a_0 + \frac{1}{a_1 + \frac{1}{\left(\frac{1}{(x_1 - a_1)}\right)}}$$

Inverting the fractional part again we end up with $x_2 = (x_1 - a_1)^{-1}$ and repeat the process.

We will explore this idea in *Maple* with a terminating decimal $x = 1.23456789$. We will use the **floor** command to extract the integer part. The reader is encouraged to read the help documentation regarding this function. We will be careful to make sure we use the rational representation of our number, rather than the decimal representation, in order to allow *Maple* to keep the calculations exact. We also need to tell the **convert** function to perform an exact conversion to a rational number, otherwise it will try to make a rational approximation of the decimal number it is given. See the *Maple*'s help files on *convert/rational* for more details.

```
> x := convert(1.23456789, rational, exact);
```

$$x := \frac{123456789}{100000000}$$

```
> x0 := x; L := floor(x0);
```

$$x_0 := \frac{123456789}{100000000}$$

$$L := 1$$

```
> x1 := 1/(x0 - floor(x0)); L := L, floor(x1)
```

$$x_1 := \frac{100000000}{23456789}$$

$$L := 1, 4$$

So we currently have $a_0 = 1$ and $a_1 = 4$ for our continued fraction. Next we will need to calculate $x_2 = x_1 - 4$ and add $floor(x_2)$ to the end of the list, and repeat the process. Since we started off with a terminating decimal, we should expect the continued fraction to also terminate. So instead of tiring our fingers typing the same two commands over and over again, we will write a loop to finish the process for us.

> **for** i **from** 2 **while** $x_{i-1} - \text{floor}(x_{i-1}) \neq 0$ **do**
> $x_i := \dfrac{1}{x_{i-1} - \text{floor}(x_{i-1})}; L := L, \text{floor}(x_1);$
> **od**

$$x_2 := \frac{23456789}{6172844}$$
$$L := 1, 4, 3$$
$$x_3 := \frac{6172844}{4938257}$$
$$L := 1, 4, 5, 1$$
$$x_4 := \frac{4938257}{1234587}$$
$$L := 1, 4, 3, 1, 3$$
$$x_5 := \frac{1234587}{1234496}$$
$$L := 1, 4, 3, 1, 3, 1$$
$$x_6 := \frac{1234496}{91}$$
$$L := 1, 4, 3, 1, 3, 1, 13565$$
$$x_7 := \frac{91}{81}$$
$$L := 1, 4, 3, 1, 3, 1, 13565, 1$$
$$x_8 := \frac{81}{10}$$
$$L := 1, 4, 3, 1, 3, 1, 13565, 1, 8$$
$$x_9 := 10$$
$$L := 1, 4, 3, 1, 3, 1, 13565, 1, 8, 10$$

And so we have the continued fraction representation of 1.23456789 as

$$1 + \cfrac{1}{4 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{13565 + \cfrac{1}{1 + \cfrac{1}{8 + \frac{1}{10}}}}}}}}}$$

or, more compactly, $[1; 4, 3, 1, 3, 1, 13565, 1, 8, 10]$. For the remainder of this document we will only use this more compact notation when referring to continued fractions.

As the reader might very well have come to expect by now, *Maple* has functions inbuilt to allow for the conversion of a real or rational number to a continued fraction. There are, in fact, two methods. The first method is to use the highly flexible **convert** function, which is explored somewhat in Exercise **??**.

```
> convert(123456789/100000000, confrac)
```

$$[1, 4, 3, 1, 3, 1, 13565, 1, 8, 10]$$

However, you should be aware that using this method may run afoul of *Maple*'s internal digit precision. For instance, with the default precision of 10 digits, *Maple* produces the following output.

```
> convert(1.23456789, confrac)
```

$$[1, 4, 3, 1, 3, 1, 13565]$$

Which is missing the last 3 quotients. However, if we raise the internal precision to 20 digits then we obtain the correct result. For best results, one should avoid using decimals with the continued fraction conversion when using the **convert** funtion, if at all possible.

The alternative, and probably preferred, method is to use the **cfrac** function which is located within the *numtheory* package. This function seems to behave better with decimal numbers but note, however, that it will write the continued fraction out in the long form, unless we ask it otherwise.

```
> x := convert(1.23456789, rational, exact) :
  numtheory[cfrac](x); numtheory[cfrac](1.23456789, quotients)
```

$$1 + \cfrac{1}{4 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{3 + \cfrac{1}{1 + \cfrac{1}{13565 + \cfrac{1}{1 + \cfrac{1}{8 + \frac{1}{10}}}}}}}}}$$

$$[1, 4, 3, 1, 3, 1, 13565, 1, 8, 10]$$

The **cfrac** function also allows for converting a converted fraction in list form back to a rational number (see Exercise 23).

Let us explore continued fractions with some irrational numbers now. We'll start with the golden ratio $\phi = \frac{1+\sqrt{5}}{2}$. Since the name *phi* is reserved in the *numtheory* package, we will avoid using it, and use GR (short for golden ratio) instead.

```
> GR = (1 + sqrt(5))/2; convert(GR, confrac)
```

$$[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

Well that's interesting. Let's what the *numtheory* package makes of it

```
> numtheory[cfrac](GR, quotients)
```

$$[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, \ldots]$$

```
> numtheory[cfrac](GR, quotients, 20)
```

$$[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, \ldots]$$

Note here that the ellipses above do not mean "continuing on in the same fashion" like they usually would in printed mathematics, but instead means something closer to "and some more stuff that wasn't calculated". So what we know is that the first 21 quotients in the continued fraction of ϕ are all 1, and that there are some more quotients that we don't know about.

As it happens, being an irrational number, the continued fraction representation of ϕ is infinite, just as its decimal representation. However, unlike its decimal representation—which exhibits no particularly discernable pattern—the continued fraction representation does indeed exhibit a clear pattern. The pattern we have seen above continues infinitely for a continued fraction representation that is all 1.

So why is this? Well, first notice that $2 < \sqrt{5} < 3$ since $4 < 5 < 9$, and so $1 < \phi < 2$. This tells us straight away that the integer part of ϕ is 1. Subtracting this yields

$$\phi - 1 = \frac{1+\sqrt{5}}{2} - \frac{2}{2} = \frac{\sqrt{5}-1}{2}$$

Inverting this we get

$$\frac{2}{\sqrt{5}-1} = \frac{2}{\sqrt{5}-1} \cdot \frac{\sqrt{5}+1}{\sqrt{5}+1} = \frac{2(\sqrt{5}+1)}{4} = \frac{1+\sqrt{5}}{2} = \phi$$

Which is back where we started from. It should be clear from this then that both $\phi - 1 = \frac{1}{\phi}$ and that we can continue the continued fraction process begun above indefinitely providing the continued fraction $[1; 1, 1, \ldots]$

Finally, let us look at the continued fractions of some other irrationals

```
> with(numtheory):
  cfrac(exp(1), quotients, 20);
  cfrac(sqrt(2), quotients, 20);
  cfrac(sqrt(5), quotients, 20)

        [2, 1, 2, 1, 1, 4, 1, 1, 6, 1, 1, 8, 1, 1, 10, 1, 1, 12, 1, 1, 14, ...]
          [1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ...]
          [2, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ...]
```

We can see, then, that continued fractions may give more information regarding the patterns behind a real number that its decimal expansion might. When exploring an unknown number, we should look at both a decimal approximation, as well as its continued fraction. However, not all numbers have a nice continued fraction representation, as we explore in Exercise 24.

1.3.3 Recurrence Relations

We should recall from Section 1.2.4 the Fibonacci numbers, and their formulation as $f(n) = f(n-1) + f(n-2)$ where $n \in \mathbb{N}$ and $f(1) = f(2) = 1$.

This is an example of a recurrence relation - a sequence where the value of each element is dependent on one or more previous elements.

The requirement that n be natural is important here, since the recurrence relation describes a *sequence* and not a function - even though we have used functional notation so far. Recurrence relations are often written using the subscript notation more usually associated with a sequence, in which case the Fibonacci numbers are defined as a sequence $\{f_n\}_{n\in\mathbb{N}}$ where $f_n = f_{n-1}+f_{n-2}$ and $f_1 = f_2 = 1$. However, due to the way *Maple* handles recurrence relations, we will continue to use the functional notation instead of the subscript notation.

The *order* of a recurrence relation is how far back one must look in order to calculate a term. The Fibonacci numbers are therefore an order 2 because one needs to know the two previous terms in order to calculate any term. A recurrence relation may have other properties, but for simplicities sakes, let us say that a *constant coefficient, linear, homogeneous* recurrence relation of order k is a recurrence relation that has the form

$$a(n) = c_1 \cdot a(n-1) + \cdots + c_k \cdot a(n-k)$$

where the c_i are constants.

We will look at some 1^{st} order recurrence relations to start with. In fact, we will look in full generality at first order linear recurrence relations with constant coefficients. Let $a(n) = c_1 \cdot a(n-1)$. This is the general form of a 1^{st} order linear homogenous recurrence relation. If we wish to know what the 100^{th} term in the sequence was—provided, of course that we knew both the 1^{st} term $a(0)$, and the coefficient c_1)—then we would have to calculate the 2^{nd} term before we could calculate the 3^{rd} term and so on. All in all we would have to perform 99 calculations. In fact, this is exactly what he have had to do when we wrote loops and procedures to calculate the Fibonacci numbers in Section 1.2.3 and Section 1.2.4.

What we would ideally like is some formula or function of n that would always calculate the n^{th} term in the sequence. The act of finding such a formula is called *solvign* the recurrence relation. In the case of 1^{st} order relations—and especially 1^{st}order linear homogeneous recurrence relations with constant coefficients—doing so is quite straight forward. Observe that

$$\begin{aligned}
a(0) &= 1 \cdot a(0) && = (c_1)^0 \cdot a(0)\\
a(1) &= c_1 \cdot a(0) && = (c_1)^1 \cdot a(0)\\
a(2) &= c_1 \cdot a(1) = c_1 \cdot (c_1 \cdot a(0)) && = (c_1)^2 \cdot a(0)\\
a(3) &= c_1 \cdot a(2) = c_1 \cdot \big((c_1)^2 \cdot a(0)\big) && = (c_1)^3 \cdot a(0)
\end{aligned}$$

and so on. The pattern here is quite clear.

$$a(n) = (c_1)^n \cdot a(0)$$

We may quite easily now, if we wished, calculate the 100^{th} element of the sequence as $(c_1)^{99} \cdot a(0)$.

A simple example of such a recurrence relation is a bank account which earns, say, 5% interest both calculated and paid annually. If we let $a(n)$ be the amount of money at the end of n years, then $a(0)$ is the initial deposit, and $a(n) = 1.05 \cdot a(0)$. If we start with \$5000 then we know, thanks to the analysis performed above, that after 5 years we will have $(1.05)^5 \cdot 5000 = \$6381.41$.

We can explore recurrence relations inside of *Maple*, of course. *Maple* provides the **rsolve** command for solving recurrence relations. Happily, this function is not limited to only linear, homogeneous, constant coefficient recurrence relations. Unfortunately, not all recurrence relations are solvable. Let us see what *Maple* says about our earlier analysis.

```
> rsolve(a(n) = c1 · a(n − 1), a(n))
```

$$a(0)c_1^n$$

Here we asked *Maple* to solve the recurrence relation $a(n) = c_1 \cdot a(n-1)$ and asked it to solve for $a(n)$. This means that the answer returned by the **rsolve** function should be the formula for $a(n)$. As we should hope, *Maple* gave us the answer we already knew. Let us now, quickly, drop the constant coefficient condition, and see what happens. For maximum generality, we will assume that a function $f(n)$ multiplies the previous term in the recurrence.

```
> rsolve(a(n) = f(n) · a(n − 1), a(n))
```

$$\prod_{n0=0}^{n-1} f(n0+1)\ a(0)$$

It is not immediately clear which terms are part of the product. However, if *Maple* has multiple terms within a sum or product, it will enclose them in parentheses. With this in mind, the $a(0)$ term isn't inside the product, and the entire expression might less ambiguously be written as

$$\left(\prod_{n0=0}^{n-1} f(n0+1)\right) a(0) \quad \text{or} \quad a(0) \prod_{n0=0}^{n-1} f(n0+1)$$

We may confirm the result using a very similar analysis to that performed

for the constant coefficient cast

$$\begin{aligned}
a(1) = f(1)\,a(0) &= \left(\prod_{k=1}^{1} f(k)\right) a(0) = a(0)\prod_{k=0}^{1-1} f(k+1)\\
a(2) = f(2)\,a(1) = f(2)f(1)a(0) &= \left(\prod_{k=1}^{2} f(k)\right) a(0) = a(0)\prod_{k=0}^{2-1} f(k+1)\\
a(3) = f(3)\,a(2) = f(3)f(2)f(1)a(0) &= \left(\prod_{k=1}^{3} f(k)\right) a(0) = a(0)\prod_{k=0}^{3-1} f(k+1)\\
&\vdots\\
a(n) = f(n)f(n-1)\cdots f(1)\,a(0) &= \left(\prod_{k=1}^{n} f(k)\right) a(0) = a(0)\prod_{k=0}^{n-1} f(k+1)
\end{aligned}$$

Let us now look at 2nd order linear recurrence relations with constant coefficients. Performing an analysis like the ones above that we performed for 1st order recurrence relations is of limited to no use with 2nd order relations. In short—there's too much going on to see a clear pattern. Fortunately there are some nice theorems that allow for easy solution.

Given a 2nd order linear, homogeneous recurrence relation with constant coefficients, $a(n) = a_1 \cdot a(n-1) + a_2 \cdot a(n-2)$ we construct the *characteristic polynomial* $x^2 - a_1 x - a_2$ and find the roots. Note that the recurrence may be re-written as $a(n) - a_1 \cdot a(n-1) + a_2 \cdot a(n-2) = 0$ and so the transformation to the characteristic polynomial should be more clear now. Once we have the roots r_1, r_2 of the characteristic polynomial, then the recurrence relation has a formula depending on whether the roots are distinct or not. The general form of the solution is

$$a(n) = \begin{cases} A \cdot (r_1)^n + B \cdot (r_2)^n & \text{if } r_1 \neq r_2 \\ A \cdot (r_1)^n + n \cdot B \cdot (r_1)^n & \text{if } r_1 = r_2 \end{cases}$$

where A and B are constants. If we know some initial conditions, then we may substitute them into the general form of $a(n)$ in order to calculate exactly what the constants A and B are. What is interesting here is that the general form will *always* satisfy the recurrence relation no matter the choice of constants.

Let us explore this with the Fibonacci numbers. The characteristic polynomial we need to find the roots of is is $x^2 - x - 1$. Using the quadratic formula $x = \frac{1}{2a}(-b \pm \sqrt{b^2 - 4ac})$ we have $x = \frac{1}{2}(1 \pm \sqrt{5})$. So $r_1 = \frac{1}{2}(1 + \sqrt{5})$—which is the golden ratio that we looked at in Section 1.3.2—and $r_2 = \frac{1}{2}(1 - \sqrt{5})$. Turning to *Maple* now.

```
> f := n → A·((1+sqrt(5))/2)^n + B·((1-sqrt(5))/2)^n;
```

$$f := n \to A\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n + B\left(-\frac{1}{2}\sqrt{5}+\frac{1}{2}\right)^n$$

```
> f(n-1)+f(n-2); simplify(%)
```

$$A\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^{n-1} + B\left(-\frac{1}{2}\sqrt{5}+\frac{1}{2}\right)^{n-1} + A\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^{n-2} + B\left(-\frac{1}{2}\sqrt{5}+\frac{1}{2}\right)^{n-2}$$

$$\frac{16\left(A\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n + B\left(-\frac{1}{2}\sqrt{5}+\frac{1}{2}\right)^n\right)}{\left(\sqrt{5}+1\right)^2\left(\sqrt{5}-1\right)^2}$$

Unfortunately, for some reason, *Maple* doesn't simplify the denominator of the expression. It should take no time at all to calculate in our head that

$$\left(\sqrt{5}+1\right)^2\left(\sqrt{5}-1\right)^2 = \left(\left(\sqrt{5}+1\right)\left(\sqrt{5}-1\right)\right)^2 = 4^2 = 16$$

and so the denominator and the multiple of 16 will cancel. We could, also, have asked *Maple* to specifically simplify only the denominator.

```
> denom(%) = simplify(denom(%))
```

$$\left(\sqrt{5}+1\right)^2\left(\sqrt{5}-1\right)^2 = 16$$

In either case, we end up with the formula for $f(n)$ when we simplify $f(n-1)+f(n-2)$, showing that the choice of constants A and B does not matter.

Now, in order to find the constants for the specific case of the Fibonacci numbers, we could plug $f(1) = 1$ and $f(2) = 1$ into our equation, above - and solve for A and B. Instead of this, however, we'll just ask *Maple* to solve the recursion for us. However, because we have already defined the the variable f, above - we will be sure to use a different variable name this time.

```
> rsolve({F(n) = F(n-1)+F(n-2), F(1) = 1, F(2) = 1}, F(n))
```

$$\frac{1}{5}\sqrt{5}\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n - \frac{1}{5}\sqrt{5}\left(\frac{1}{2}-\frac{1}{2}\sqrt{5}\right)^n$$

and so it would seem that $A = \frac{1}{5}\sqrt{5}$ and $B = -\frac{1}{5}\sqrt{5}$.

1.3.4 The Sieve of Eratosthenes

It would be hard to justify this chapter as being about number theory if we didn't mention prime numbers at some stage. Here we will look at the problem of listing numbers which are prime, and will use a technique known to the ancient Greeks. In particular, it was a man named Eratosthenes who is responsible for this technique.

Suppose we want to find all the prime numbers less than some number, 100 say. First we write the numbers in a square (or rectangle) thusly

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Since we know that 1 is not prime, we have crossed it out to start with. We now find the first uncrossed number, 2 in this case. This number will be prime (and we know it is, anyway), and so no multiple of that number can be prime. So we go and cross out all the multiples of this number.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Having done that, we find the next number along (after 2, which was our previous number) which is not crossed out. In this case it's 3. This number must be prime because it is not a multiple of any prime number less than it - of which there was only one in this case. We now cross out all multiples

of 3.

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	25	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	35	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	49	~~50~~
~~51~~	~~52~~	53	~~54~~	55	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	65	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	77	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	85	~~86~~	~~87~~	~~88~~	89	~~90~~
91	~~92~~	~~93~~	~~94~~	95	~~96~~	97	~~98~~	~~99~~	~~100~~

The next number that is not crossed out is 5, which must be prime because it is not a multiple of any prime smaller than it. We repeat this process until such times as our next uncrossed number is greater than 10. "Why 10?" you ask. Well because $10 = \sqrt{100}$, and 100 was the bound on the primes we wanted to find. Allow us to explain

When looking for divisors of a number, n say, we need only by checking whether the numbers up to (and including) $\sqrt{n}$ are divisors of n. This is true because any divisor $a < \sqrt{n}$ will have a codivisor b such that $a \cdot b = n$. If $b \leq \sqrt{n}$ then $a \cdot b < \sqrt{n} \cdot \sqrt{n} = n$, so it must be that $b > \sqrt{n}$. Furthermore, if we are only interested in the prime factors then we only need check the primes less than $\sqrt{n}$ as being factors of n.

With Eratosthenes' sieve, when we find a prime, we then go and cross off any number for which that prime is a divisor. Once we get past 10, then we have crossed off every multiple of every prime less than or equal to $\sqrt{100}$ and so we must have found at least one divisor for every number less than or equal to 100, so long as there was a divisor to find. Everything that is left uncrossed must be a prime number.

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Which gives us our list of prime numbers less than 100 as 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89 and 97.

Now we will implement this technique in *Maple*. We will make a procedure which we will name **Eratosthenes** that will return a sequence of all the primes less than some given number N which must be a positive integer.

In order to represent in *Maple* the "crossing out" of numbers as we performed in the sieve (above), we are going to create a list, L which contains N elements. If L_i is true, then i is prime, and if L_i is $false$ then i is not prime. The array will begin with all it's values as $true$ and we will "cross out" by setting the appropriate value in the list to $false$. We will also create a sequence named *primeslist* which will start out empth, and into which we will add the primes as we find them.

```
> Eratosthenes := proc(N :: posint)
    local L, primeslist, n, k;
    L := Array(2..N, i → true);
    for n from 2 to trunc(sqrt(N)) do
      if L_n = true then
        primeslist := primeslist, n;
        for k from n by 1 while k · n ≤ N do
          L_n := false
        od
      fi
    od
    primeslist
  end
```

```
> Eratosthenes(1000)
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
  59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109,
113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179,
181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241,
251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313,
317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389,
397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461,
463, 467, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547,
557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617,
619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691,
701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773,
787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859,
863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947,
            953, 967, 971, 977, 983, 991, 997
```

Instead of a list, we declared an array, An Array is a quite powerful, and very general class in *Maple*, but in this example it behaves almost exactly like a list. Using the array gives us two advantages over a list. Firstly we were able to specify an exact range of indices for indexing ($2..N$ in this case). Secondly we were easily able to specify a value for the array elements to have when the array was created. This initialisation was in the form of a function mapping an index variable (which we called i) to the value we wanted L_i to have. Without the array we would have needed to use something like $primeslist := [seq(true, k = 1..N)]$ to achieve a similar result. In our case above we wanted the value at all indices to be *true* but this need not have been the case, and we could have had a more complicated initialisation had we wanted.

Not sure - is seq(true, k=1..N) so bad? OTOH introducing a function assignment now might not be such a bad thing, as we'll see it later with matrices

Generating prime numbers is actually a fairly computationally intensive task. The Sieve of Eratosthenes is quite interesting in it's own right, and is reasonably efficient for such a simple algorithm. However, for primes lager than 10000 it starts to get noticeably slow. Of course as we should have well and truly come to expect by now, *Maple* provides inbuilt functions for finding primes, as well as testing primality. The big advantage to the inbuilt functions is that they perform all sorts of tricks to keep the execution times remarkably quick.

These functions are **ithprime** which calculates the i^{th} prime number, **nextprime** which calculates the next prime number *after* a given number, **prevprime** which calculates the previous prime number *before* a given number, and finally **isprime** which calculates whether a given number is prime or not.

```
> ithprime(5), nextprime(5), nextprime(6), prevprime(5), prevprime(4)
                                  11, 7, 7, 3, 3
```

1.4 Problems and Exercises

1. Enter the following expressions into *Maple*

   (a) $\frac{1}{2}$

   (b) e^2

   (c) $x^2 + x - 1$

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

   Perform the following calculations in *Maple*. Obtain a decimal approximations as well as simplified exact values.

   (e) $\sin\left(\frac{3\pi}{5}\right)$

   (f) $\frac{15!}{e^4}$

   (g) $\frac{12! + 2^{\frac{1}{3}}}{\sin(2)^3}$

   (h) $2^{2^{2^2}}$

   What do you notice about the floating point approximation of the final calculation, compared to the exact value?

2. We will explore *Maple*'s digit precision for decimal approximations. Enter the following commands into *Maple* and explain how the precision has been modified. Try some entries of your own if you are still unsure, or to test your explanation.

   (a) > *evalf*(Pi); *evalf*(exp(1))

   (b) > *evalf*[20](Pi); *evalf*[25](exp(1))

   (c) > *evalf*(Pi, 35); *evalf*(exp(1), 30)

   (d) > *Digits* := 50; *evalf*(Pi); *evalf*(exp(1))

   Obtain numeric approximations of the following expressions, to the indicated digit precision

   (e) π^2 to 10 decimal places

   (f) $\sin(1)$ to 10 decimal places

   (g) e^{π} to 15 decimal places

   (h) $\log(\pi)$ to 22 decimal places

   You can reset the digit precision to 10 decimal places using either *Digits* := 10 or the **restart** command. Beware that the **restart** command will completely undo anything you have done previously.

3. This exercise shows two cautionary scenarios.

   (a) Some variables in *Maple* are protected—which means their values cannot be changed. Enter the following into *Maple*

i. > $NULL := 7$

ii. > $\sin := \exp(2)$

iii. > $\log := x^2 + 3x - 2$

iv. > $sum := \dfrac{x+1}{2x-2}$

Why might these names be protected?

(b) We will look at why caution should be taken when using the % operator. Follow the instructions below

i. Enter the following *Maple* commands. Make sure to keep each entry in its own input bracket

> $(x+y)^3$

> $expand(\%)$

> $subs(y = 1, \%)$

ii. Edit the 1st command to instead read $(x+y)^4$. Be sure to remember to hit *enter* to perform the new calculation

iii. Go back to the 3rd calculation and press enter. What happened?

4. The **convert** function is a very useful fucntion with very many uses. The following are just a tiny handful of examples of what the convert function is capable of. Enter the following into *Maple* and also have a read of the help files regarding the **convert** function.

(a) > $convert\left(\dfrac{3 \cdot Pi}{2}, degrees\right)$

(b) > $convert(25\ degrees, radians)$

(c) > $convert\left(10, units, \dfrac{metres}{second}, \dfrac{kilometres}{hour}\right)$

(d) > $convert\left(60, units, \dfrac{km}{h}, \dfrac{m}{s}\right)$

(e) > $convert(\sinh(x), \exp)$

(f) > $convert\left(\dfrac{\exp(x) + \exp(-x)}{2}, trig\right)$

(g) > $convert([x, 2x, 3x, 4x], `+`)$

(h) > $convert([x, 2x, 3x, 4x], `*`)$

Now perform the following conversions.

(i) How many furlongs per fortnight is 60 kilometres per hour?

(j) What is the trigonometric identity for $\dfrac{e^x - e^{-x}}{e^x + e^{-x}}$

(k) What is the partial fraction representation of $\dfrac{x^2 + 3x + 2}{(x-1)(x^2+2)^2}$

5. We have seen the **$** operator used to create repeated sequences. It may also be used as a shortcut to the **seq** command. Enter the following commands, and explain how the **$** operator works. Try some entries of your own if you are still unsure, or to test your explanation.

   (a) > $k^2\$k = 4..10$
   (b) > $a_i\$i = 3..6$
   (c) > $\frac{1}{k}\$k = 1..7$
   (d) > $\$3..5$

   Confirm your guess with the *Maple*'s help: (**?$**)

6. *Maple*'s **seq** command is capable of creating sequences where the index variable increases by more than 1 at each step, or even where the index variable takes on arbitrary values entirely. Enter the following into *Maple* and explain what is happening. Note that the index variable can be any valid variable.

   (a) > $seq(i^2, i = 1..10, 3)$
   (b) > $seq(a^2 - a - 1, a = 1..10, 3)$
   (c) > $seq\left(\frac{2}{n}, n \textbf{ in } [2, 3, 5, 7, 11, 12]\right)$
   (d) > $seq\left(fib^2, fib \textbf{ in } [1, 1, 2, 3, 5, 8]\right)$

   Now produce the following sequences using the **seq** command, and the ideas above.

   (e) $9, 25, 49, 81$
   (f) $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{7}, \frac{1}{11}, \frac{1}{16}$

   *Note that neither of these techniques can be used with the **$** operator—see Example 5*

7. For lists and sequences—and many other *Maple* constructs that use natural indices—indexing may be performed by using a negative index which begins counting from the end, instead of the beginning. To illustrate this, create in *Maple* a list $L := [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$. Issue the following commands.

   (a) > L_{-1}
   (b) > $L[-3]$
   (c) > $L_{-4..-2}$
   (d) > $L[-7..-5]$
   (e) > $L_{-4..}$
   (f) > $L[..-6]$

   Which of the following commands do you expect to fail? Enter them into *Maple* and see if you were correct. Explain why the failures occurred.

(g) $L_{-1..-3}$
(h) $L[3..-3]$
(i) $L[4..-7]$
(j) $L_{-7..2}$

Hint: Try and change the negative index into the usual positive index.

8. Enter the following commands into *Maple*. For the purposes of these commands $L := [x, y, z, x, y]$

(a) > $op([1,2,3,4,5])$
(b) > $nops([1,2,3,4,5])$
(c) > $numboccur([1,2,3,4,5],3)$
(d) > $op(L)$
(e) > $nops(L)$
(f) > $numboccur(L,x)$

What does it look like the **op**, **nops** and **numboccur** functions are doing? Now change L to be $L := [[1,2],[2,3],[3,4]]$ and consider the following.

(g) > $op(L)$
(h) > $nops(L)$
(i) > $numboccur(L,3)$
(j) > $op(x^2+2x-3)$
(k) > $nops(x^2+2x-3)$
(l) > $numboccur(x^2+2x-3,x)$

What do you now think **op**, **nops** and **numboccur** are doing? Read *Maple*'s help files regarding these functions and see if they agree with your observations and guesses.

Finally, have a look at the *ListTools* package, and in particular the **Occurrences** function (which may be directly reached with **ListTools[Occurences]**). Change L to be $L := [[1,2],[3,4],5,6,[1,2]]$ and use this function to have *Maple* calculate the following. You should be able to verify the answers by eye.

(m) The number of occurrences of the element $[1,2]$ in the list L
(n) The number of occurrences of the element 5 in the list L
(o) The number of occurrences of the element x in the list L
(p) The number of occurrences of the element $[y,z]$ in the list L

9. Create a list containing the first 1000 digits of π, in the correct order. In order to do this you will need to manipulate the first 1000 digits of π into an integer and use the command **convert(n, base, 10)** which will convert an *integer* n into a list of digits in base 10.

Hint: You will need to use the **trunc** command at some stage.

10. Calculate the following sums and products

(a) $\displaystyle\sum_{i=1}^{100} \frac{1}{2^i}$ (c) $\displaystyle\sum_{k=0}^{\infty} \frac{1}{n!}$

(b) $\displaystyle\prod_{n=1}^{6} (1 + x^{2^n})$ (d) $\displaystyle\prod_{k=0}^{\infty} \frac{4k^2}{4k^2 - 1}$

The following should look familiar from first year calculus.

(e) $\displaystyle\sum_{k=0}^{\infty} \frac{x^{2n}}{(2n)!}$ (f) $\displaystyle\sum_{k=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$

Convert the following sums to sigma notation, then use that to implement them using the **sum** command.

(g) $1 + 8 + 27 + \cdots + n^3$

(h) $1 + \dfrac{1}{2} + \dfrac{1}{4} + \cdots + \dfrac{1}{2^i}$

(i) $\dfrac{1}{1 \cdot 3} + \dfrac{1}{3 \cdot 5} + \dfrac{1}{5 \cdot 7} + \cdots + \dfrac{1}{(2k-1)(2k+1)}$

11. Use *Maple*'s **sum** command to explore the binomial formula. Recall that the binomial theorem states that

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k \text{ where } \binom{n}{k} = \frac{n!}{(n-k)!k!}$$

You should also use *Maple*'s help to find a built in function that will calculate the binomial coefficients $\binom{n}{k}$.

(a) Create a function, f say, using the arrow notation ($\rightarrow$) that expands $(x+y)^n$.

(b) Create another function, g say, using the arrow notation that uses *Maple*'s **sum** command to evaluate the sum in the binomial formula.

(c) Choose some values for n and check that $f(n) = g(n)$.

(d) Ask *Maple* to evaluate $f(N)$. What does it evaluate to?

Perform these tasks twice—once using *Maple*'s built-in function for binomial coefficients and once using the formula for the coefficients.

12. An arithmetic sequence is a sequence where each term differs from the previous term by a fixed amount. This "fixed amount" is called the *common difference* of the sequence. For example $a = \{1, 2, 3, 4, \ldots\}$, $b = \{1, 3, 5, 7, \ldots\}$ and $c = \{1, 4, 7, 10, \ldots\}$ are arithmetic sequences with a common difference of 1, 2 and 3 respectively.

Arithmetic sequences beginning at 1 generate polygonal numbers. The sequence a generates the triangular numbers—the n^{th} triangular number is the sum of the first n terms in the arithmetic progression. Similarly the n^{th} square number is the sum of the first n terms in the arithmetic progression b and the n^{th} pentagonal number is the sum of the first n terms in the arithmetic sequence c. *Note:* the square numbers are exactly the numbers that are perfect squares.

(a) Write a function using the arrow notation that will output the first n terms in the arithmetic sequence c (above) that will generate the pentagonal numbers.

(b) Write another function using the arrow notation that will calculate the n^{th} pentagonal number.

(c) Repeat (a) and (b) for the hexagonal numbers. (*You will need to extrapolate which arithmetic sequence generates the hexagonal numbers*).

Hint: Can you find a formula for the n^{th} element of the sequence?

13. Enter the following into *Maple*. What is causing the errors? Can you modify the commands so that they work?

(a) > $k := 1; sum\left(\frac{1}{k}, k = 1..10\right)$

(b) > $n := 3; Sum(n^3, n = 1..5); value(\%)$

(c) > $f := N \rightarrow sum(k, k = 1..N); seq(f(k), k = 1..10)$

Note that the **seq** command does not exhibit this problem.

(d) > $k := 1; seq\left(\frac{1}{k}, k = 1..10\right) :' k' = k;$

(e) > $n := 3; seq(n^3, n = 1..5) :' n' := n;$

14. If we wish to apply a function to every item in a list, *Maple* provides a function named **map**. Enter the following commands, and read *Maple*'s help on the **map** function. What is the **map** function doing?

(a) > $map(\exp, [1, 2, 3, 4, 5])$

(b) > $L := [0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi]; map(\sin, L)$

(c) > $f := x \rightarrow 2 \cdot x^2; map(f, [0, 2, 4, 6, 8])$

(d) > $f := x \rightarrow \frac{x}{2}; L := [0, \frac{1}{2}, 1, \frac{3}{2}, 2]; map(f, L)$

(e) > $map(N \rightarrow \frac{1}{N}, [2, Pi, \exp(1), \log(Pi)])$

(f) > $C := [seq(k + 1 + k \cdot I)]; map(z \rightarrow \text{abs}(z), C)$

Create a list P which contains the first 5 prime numbers $[2, 3, 5, 7, 11]$. Use **map** and the list P to create the following lists

(g) $[3, 4, 6, 8, 12]$

(h) $[5, 10, 26, 50, 122]$

(i) $[\ln(2\pi), \ln(3\pi), \ln(5\pi), ln(7\pi), ln(11\pi)]$

15. Just like sequences (see Exercise 6) *Maple* **for** loops are capable of having their counter variable increase by more than 1 at each step, or even have the counter variable take on arbitrary values entirely (from a list or a set). Enter the following into *Maple* and explain what is happening.

(a) > **for** i **from** 1 **to** 10 **by** 3 **do** i^2 **od**

(b) > **for** a **from** 1 **to** 10 **by** 3 **do** $a^2 - a - 1$ **od**

(c) > **for** n **in** $[2, 3, 5, 7, 11, 12]$ **do** $\frac{2}{n}$ **od**

(d) > **for** fib **in** $[1, 1, 2, 3, 5, 8]$ **do** fib^2 **od**

Loops in *Maple* may also continue indefinitely until some criteria is met. Enter the following, and explain what the loop is doing.

(e) > **for** i **from** 1 **while** $i^2 < 1000$ **do** i **od**

(f) > **for** N **from** 1 **by** 2 **while** $N^3 < 10000$ **do** $N, N^3, N^3 - N^2$ **od**

Create loops to calculate the following

(g) Decimal approximations of e, e^6, e^{11} and so on where the power of e is less than 100.

(h) Decimal approximations of $\frac{\pi}{2}, \frac{\pi}{3}, \frac{\pi}{5}, \frac{\pi}{7}, \frac{\pi}{11}$.

(i) The Fibonacci numbers less than 1200.

Note: the n^{th} Fibonacci number can be calculated with the inbuilt command **combinat[fibonacci](n)**

16. Implement the following procedure. What does it appear to do?

```
> f := proc()
    local a, b;
    global G, H;
    a, b := 1, 2;
    G, H := 3, 4;
  end
```

Now perform the following in *Maple*.

(a) > $a := \text{Pi}; a; f(); a;$

(b) > $b := \exp(2); b; f(); b;$

(c) > $G := \log(Pi); G; f(); G;$

(d) > $H := \exp(Pi); H; f(); H;$

What is happening to the **local** and **global** variables. What do you think is the difference between a local and a global variable for a procedure. Perform some tests in *Maple* and refer to the help files on procedures (**?procedure**) to confirm your guess.

17. Write procedures to perform the following. Your procedures may have 2 or more input variables by having a sequence of variable names inside the parentheses after the **proc** declaration. For example **proc**(x, y) or **proc**(a, b, c).

    (a)

    (b)

    (c)

18. We may measure the amount of execution time taken for a command, or set of commands. The command **time()** will return the current execution time. Note that this is different to system time. Enter the following in order to measure the amount of time taken by the **sin** command.

    (a) > $start := time(); \sin(2); time() - start$

    Use this technique to measure the execution time of the various Fibonacci procedures from Section 1.2.4 as indicated below. If you are feeling clever, try and create a procedure that will perform the timing.

    (b) fib(100)

    (c) fib2(10)

    (d) combinat[fibonacci](100)

    (e) fib(10000)

    (f) fib2(30)

    (g) combinat[fibonacci](10000)

    Why might the *fib2* procedure be showing the speeds it does? Have a read of *Maple*'s help files for *option* and *remember*.

    (h) Re-implement the *fib2* function with the *remember* option, and measure its execution times. Do the execution times change?

19. Use nesting to perform the following.

    (a) Print out the first 5 rows of Pascal's triangle.

    (b) Create a function or procedure that will print out the first N rows of Pascal's triangle

Hint: Try asking *Maple* what it knows about Pascal's triangle. You might need to not use punctuation. Also recall that Exercise 11 dealt with the binomial formula, which is related to Pascal's triangle.

20. One should be careful when using the % operator for creating functions. Enter the following into maple

    (a)
    ```
    > sum(k, k = 1..n);
      f := n -> %;
      'f'(2) = f(2);
    ```

    (b)
    ```
    > sum(k, k = 1..n);
      f := unapply(%, n);
      'f'(2) = f(2);
    ```

    What appears to have happened here?

21. Use the **sum** command to find a formula for an arbitrary arithmetic sequence beginning at 1 with common difference d. Use induction to prove this formula. (See Exercise 12 for the definition of an arithmetic sequence)

    Repeat this for an arbitrary arithmetic sequence beginning at a with common difference d.

22. Perform the manual calculation of the continued fraction of 1.23456789 (the one involving the **for** loop) from Section 1.3.2 without using a rational number to begin with. What happens? Why do you think this is happening?

    WARNING: Save your worksheet before attempting this. You will probably want to use the "stop sign" icon to cancel computations, if things seem to be going on for too long.

23. Use the **cfrac** function to change the following continued fractions into rational numbers.

    (a) $[1, 2]$
    (b) $[1, 1, 2]$
    (c) $[1, 1, 1, 2]$
    (d) $[1, 1, 1, 1, 2]$

    Do you notice an interesting pattern with these examples? Formulate a conjecture regarding this pattern of continued fraction and the rational number it is equal to, and test your conjecture. **Hint:** This behaviour is related to the Fibonacci numbers and their relation to the golden ratio.

24. Have *Maple* calculate the continued fractions for the following numbers.

(a) $\sqrt[3]{2}$ (b) $(\sqrt[5]{2})^3$ (c) π

Can you see any pattern? Be sure to try several computations, computing different amounts of quotients each time.

25. What is the solution to the general 1^{st} order recurrence relation

$$a(n) = f(n)\, a(n-1)^p$$

where $p > 1$ is a positive power?

Test this answer by picking some simple functions $f(n)$ (polynomials are recommended) and powers of p and seeing if the first however-many terms in the sequences agree when you calculate the terms both using the recursion, and using the formula given by **rsolve**.

26. Find general solutions to the following recurrence relations. Verify the solutions.

(a) $a(n) = 5a(n-1) - 6a(n-2)$

(b) $s(n) = 2s(n-2)$

Now find the solutions to the following recurrence relations with initial values. Test the solutions both by testing that that the formula satisfies the recurrence, and by having maple calculate the first 10 or 20 terms of the sequence from both the recurrence and the solution.

(c) $f(n) = f(n-1) + 2f(n-2)$ where $f(0) = 1, f(1) = -2$

(d) $a(n) = a(n-1) \cdot a(n-2)^2$ where $a(0) = 2, a(1) = 3$

27. Find a formula for the number of ways to climb a flight of steps of height n if 1 or 2 steps may be taken at a time.
Hint: formulate the problem as a recurrence relation.

can we get a better example than this? I nicked it from Math1510

28. Using the techniques from Exercise 18 to measure the time taken for the Sieve of Eratosthenes from Section 1.3.4 to calculate the first 100, $1,000$ and $10,000$ primes. Using this information, estimate how long you think it will take to calculate the first $100,000$ primes.

Produce the same lists of primes using *Maple*'s **seq** command and the inbuilt prime number functions, and measure how much time these take. Were these quicker or slower than the Eratosthenes sieve procedure?

1.5 Further Explorations

1. The field of rational numbers may be extended to incorporate irrational numbers in a way similar to the way in which the real numbers are extended to the complex numbers. If R is the solution of the equation $x^2 = 2$, then it must be that $R^2 = 2$, and we know that R must be an irrational number.

   If we include our new number R into the rationals (that is consider $Q' = \mathbb{Q} \cup \{R\}$) then our new set Q' will no longer be a field. In particular, what is $1+R$? In order to have a field, we need to add every rational multiple of R as well as addition of every rational number with every multiple of R.

   We end up with what is knows as a *field extention* $\mathbb{Q}(R) = a + bR | a, b \in \mathbb{Q}$ with the operations

   $$a + bR + c + dR = (a + c) + (b + d)R$$
   $$(a + bR)(c + dR) = ac + (ad + bc)R + bdR^2 = (ac + 2bd)(ad + bc)R$$

   where $a, b, c, d \in \mathbb{Q}$.

   Maple allows exploration of these ideas with the **evala** and **RootOf** functions. For instance

   ```
   > R := RootOf(x^2 = 2)
                          R := RootOf(_Z^2 - 2)
   ```

   ```
   > R^2; evala(R^2)
                          RootOf(_Z^2 - 2)^2
                                   2
   ```

2. 3x+1 Problem

3. Normality of Numbers

Chapter 2

Calculus

#insert ¡ introduction ¿

⋮

Emphasise human/computer collaboration.

2.1 Revision and Introduction

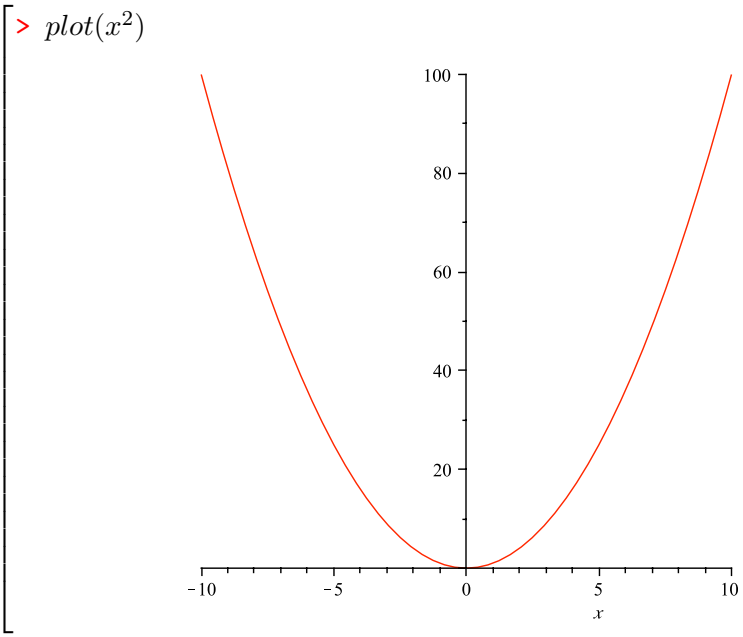
In this section we will introduce the *Maple*commands best suited for studying and performing calculus. In addition we will recall key concepts from typical first year calculus courses. It is, however, expected that the reader is familiar with the concepts behind and is able to perform such first year calculus - including (but not limited to) differentiation and integration of single variable functions, evaluation of limits, curve sketching, etc. The reader is encouraged to review their favourite (or most readily available) calculus text.

2.1.1 Plotting

In much first year calculus the ability to be able to visualise the functions and concepts being studied is quite valuable, but usually not readily available. Indeed in almost anything involving calculus visualisation is a powerful tool. So we will begin this Calculus chapter by looking at how to have *Maple*plot functions.

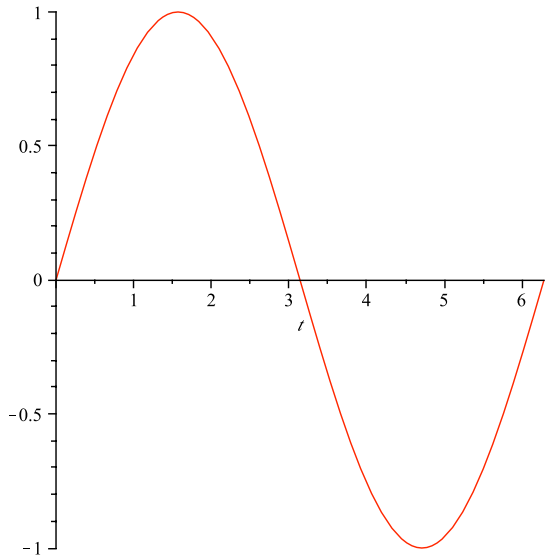
Briefly in the Section 1.1.2 we saw a plot of a cubic. The *Maple*command

to plot a function is, unsurprisingly, **plot**. The most basic use of the **plot** function is to simply give it an expression involving a single variable, usually x but any valid *Maple*variable name will work just as well.



*Maple*automatically assigns the horizontal axis to be the axis for the independent variable (x in the previous example) and labels it as such. The vertical axis is unnamed, but depicts - as it always does - the values of the function (or the dependent variable). Also, unless we tell it otherwise, the **plot** command will plot over the horizontal interval $(-10, 10)$. If we wish to plot over a different interval, we must provide a second argument.

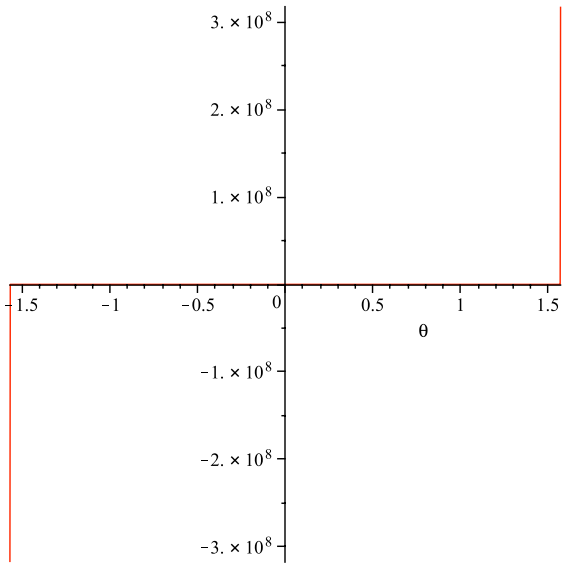
> $plot(\sin(t), t = 0..2 \cdot Pi)$



Notice here that our horizontal axis here is now t, where it was x for the parabola example. Notice the vertical axis in these two examples. The values of the vertical axis automatically adjust to suit the function we are trying to plot. For the parabola the vertical axis was between 0 and 100 - corresponding to the values the parabola would have for $-10 \leq x \leq 10$ - and the sine curve used vertical range $(-1, 1)$ just like we would have hoped it did. It should be interesting to note that the actual screen space taken up by the two plots is the same in both cases, showing that the vertical scale is different in both cases. In fact, the vertical scale and horizontal scale may be different even in the same plot, as is the case in both of these examples.

If we have a function to plot - instead of an expression - there is another way we may ask for a plot. Note above that whilst sin is definitely a function, $\sin(t)$ is actually a *Maple*expression. The former will take an input and produce an output, but the latter will not take any input at all. If we have a function or procedure we wish to plot, we may omit the input from the function/procedure, and omit the variable name from the second input parameter. For example to plot the tan function between its asymptotes we can use the following.

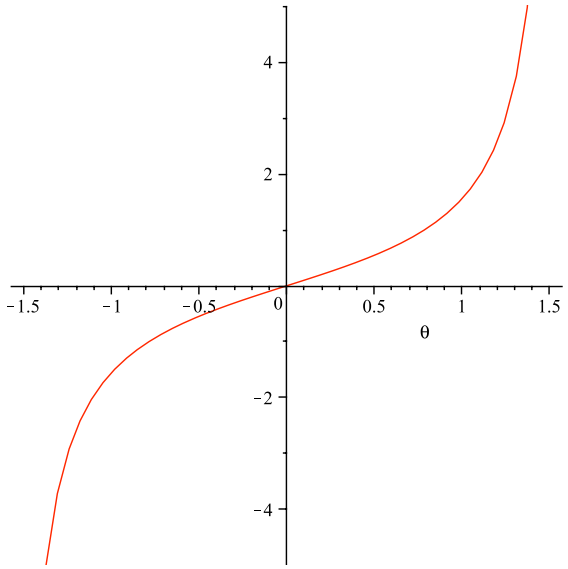
$$> \ plot\left(\tan, -\frac{Pi}{2}..\frac{Pi}{2}\right)$$



This is equivalent to the command $plot\left(\tan(t), t = -\frac{Pi}{2}..\frac{Pi}{2}\right)$. Unfortunately the plot we see certainly doesn't look like the tan function that we all know and love. If we have a look at the first plot we produced, we should notice the scale of the vertical axis is exceptionally large, and that we have two seemingly vertical lines at the end of the graph. If we were to somehow limit the vertical interval (and thus the vertical scale), perhaps we'd get a better picture.

As it happens, we may do just this with a third argument to the plot function. This third argument must be a range. It may be in the form of an interval (a..b) or may be assigned to a variable (var = a..b). In the latter case, the vertical axis will be labelled by the variable name. In either case, however, the vertical axis range must be present after the horizontal axis range. For example, if we attempt to plot the tan function.

```
> plot(tan, -Pi/2..Pi/2, -5..5)
```

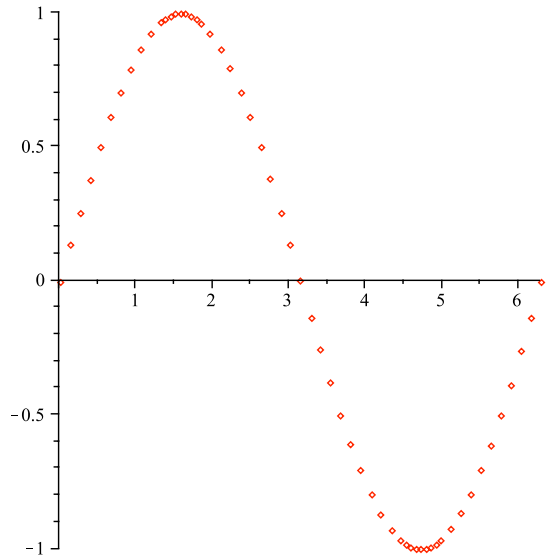


we now have a much more familiar plot. To better understand why this helped, we need to know how *Maple*plots a function.

When *Maple*plots a function it evaluates that function at various points along horizontal interval and fits a curve to the sampled points. By default *Maple*uses 50 sample points, but the number of points may be specified by an input parameter to the **plot** function. Usually the sample points are evenly spaced along the horizontal range, however *Maple*is rather clever and if it detects that the values of the function are changing too quickly between sample points, it will sample the function at an extra point between them to try and obtain more and better information to plot the function with. This extra sampling - known as subdivision - can, by default, happen up to 6 times between any two sample points and so one should be aware that *Maple*could potentially end up evaluating a function at as many as 6 times the number of sampling points requested. This behaviour is, of course, all able to be controlled and modified using input parameters see **?plot/options** and Exercise **??**.

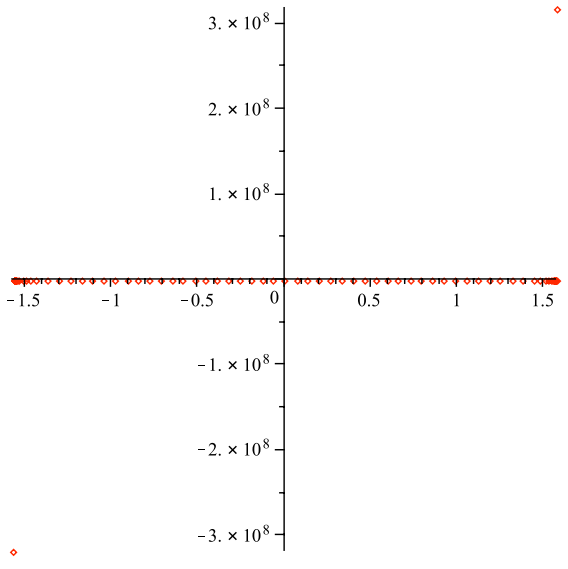
We may see this behaviour by asking *Maple*to plot points, instead of a line, as follows

> $plot(sin, 0..2 \cdot Pi, style = point)$



Observe the subdivision happening at the peaks of the curve. Now, if we have a look at the tan plot in point mode, we will see two extreme points far to the top and bottom of the graph, with the remainder of the sampled points on or very near the x-axis.

> $plot(tan, -\frac{Pi}{2}..\frac{Pi}{2}, style = point)$



Of course, with the vertical range so large, the scale is such that the points

seemingly on the x-axis could have values varying anywhere between ± 1000000 or more and we wouldn't be able to tell the difference.

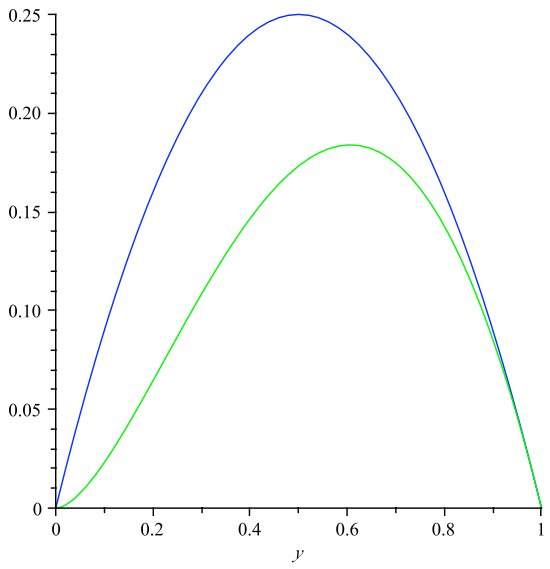
Without explicitly specifying the vertical range, *Maple*adjusts it accordingly between the largest and smallest sampled values. It should be clear then that by specifying the vertical range we see only the plot generated from the sampled points inside that range. It is worth noting that even with a specified vertical range *Maple*still samples all the same points as it would have without the specified range.

2.1.2 Multiple Plots

An interesting example comes to us from Borwein and Devlin [4]. Suppose we have two expressions, $y - y^2$ or $-y^2 \log(y)$, and wish to know - and eventually prove - which (if either) is always larger when $y \in (0, 1)$. Our instinct should be to plot the two expressions over the unit interval. However, up until now we have only plotted single expressions. Two separate plots will be of limited (if any) use to us. We need a way to plot the two expressions on the same pair of axes. This is made possible in one of two ways.

The first method is far and away the simplest. *Maple*'s **plot** command will happily plot a list of expressions (or functions). In place of a single expression we simply provide a list, and any parameters which modify the plots much also be provided in a list. For example, to plot the above two functions, wiht the first one being coloured blue, and the second coloured green (so we may identify which is which) we would enter the following.

> $plot([y - y^2, -y^2 * log(y)], y = 0..1, colour = [blue, green])$



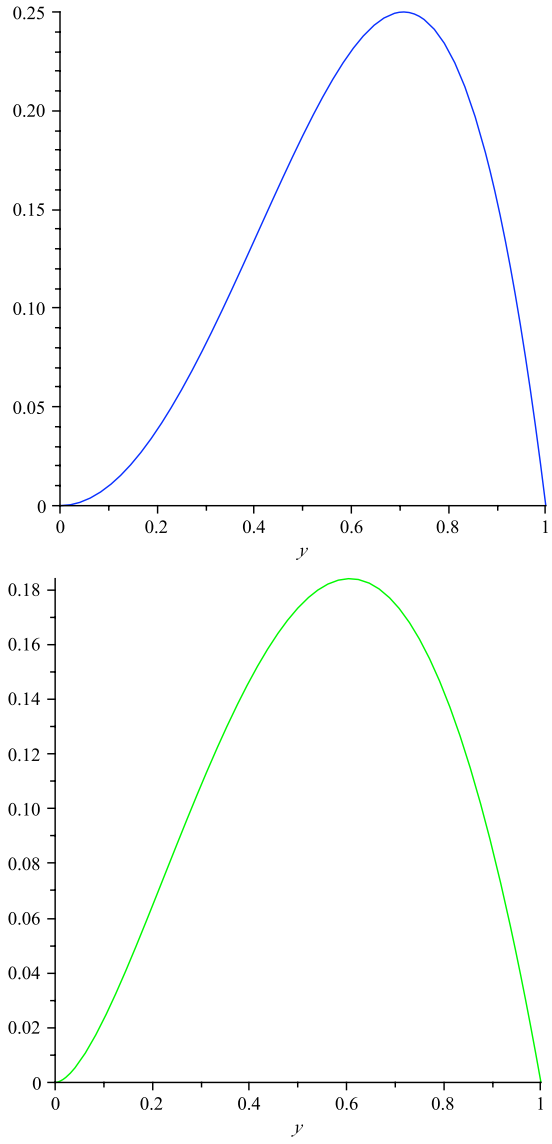
For the second method, let us now consider a modified version of our example. This time we will compare $y^2 - y^4$ and $-y^2 \log(y)$ (also from [4]). First we will make a simple observation: up until now, any valid maple expression could be assigned to a variable name. It should, then, be a natural question to ask whether the same can be done with the plot function. The answer is that yes, it can - although the logistics of such an assignment are probably not obvious. Let's try this.

> $plot1 := plot(y^2 - y^4, y = 0..1, colour = blue);$
> $plot2 := plot(-y^2 \cdot \log(y), y = 0..1, colour = green)$

$$plot1 := PLOT(\ldots)$$
$$plot2 := PLOT(\ldots)$$

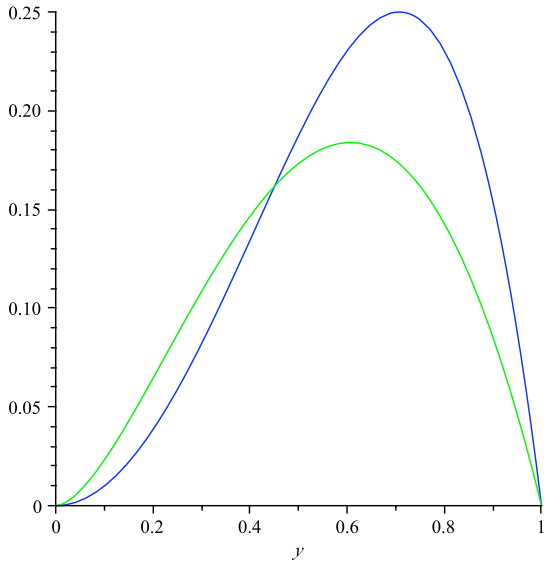
Maple stores what is known as a plot structure in the variable. If we wish to see the actual plot, then we may either just ask maple for the contents of the variable in the usual way, or we may use the **display** function. It is important to know that the **display** function is part of a *Maple*package named **plots** that is not loaded by default. The entire package can be loaded by issuing the command $with(plots)$, however we will simply ask *Maple*for the specific **display** function from that package by using what is known as the *long form* of the function name - $plots[display]$. The command should be read as "*the* **display** *function from the* **plots** *package*".

```
> plot1; plots[display](plot2)
```



This is all well and good, but it doesn't really help us show multiple plots on the same axes - at least not in any way differently from the method we have already used. The key to this lies in the **display** function, whose entire purpose is not so much the display of single stored plots (which, as we have seen can be displayed easily without the use of this function), but multiple plots. Just as the plot function will accept a list of expressions and plot them on the same axes, so will **display** accept a list (or set) of plot structures, and display them all on the same axes.

```
> display([plot1, plot2])
```

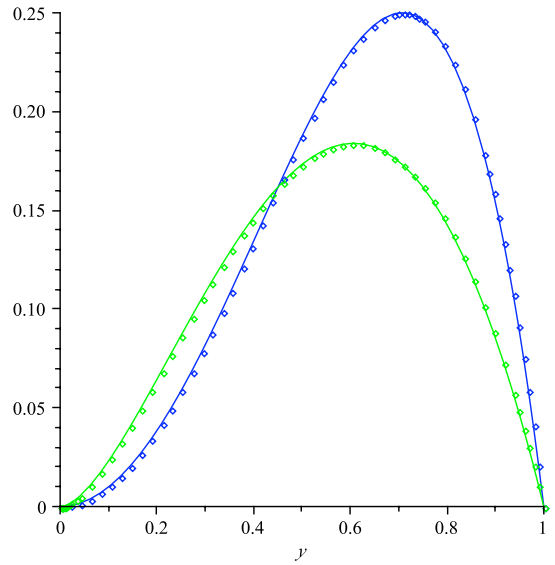


The use of **display** for the example above may seem unnecessarily long and complicated, when we can more easily use just a single plot function as we did in the first example. Such an observation is quite well founded, however there are plenty of situations where the easier method is unavailable to us, and it is for these cases where **display** really shines.

To illustrate this power, we will show the previous example and a plot of its sample points at the same time.

Not the best example, as *plot([f,f,g,g], y=0..1, style= [line, point, line, point], colour= [blue, blue, green, green])* also works happily

```
> f := y^2 - y^4; g := -y^2 * log(y);
  curves := plot([f, g], y = 0..1, colour = [blue, green]) :
  points := plot([f, g], y = 0..1, colour = [blue, green], style = point) :
  plots[display]([curves, points])
```



2.1.3 Limits

Calculus, ultimately, all comes down to limits - so it is with limits that we will begin our exploration of calculus proper. We have already seen and used, very briefly in the previous chapter, the *Maple***limit** command. We will look at it in more detail here, and recall quickly the maths behind limits.

We may take a limit of a sequence (of the infinite variety) or of a function. The intuitive (and mathematically imprecise) notion of a limit is a value that we may approach as close as we could ever wish, just by travelling sufficiently far along the sequence or function.

should we start with functions, *then* do sequences?

We'll start with sequences. Recall that the limit of a sequence $\{x_n\}_{n=1}^{\infty}$ written $\lim_{n\to\infty} x_n$ is a number k such that for every $\epsilon > 0$ we can find a natural number N and when $n \geq N$ is is the case that $|x_n - k| \leq \epsilon$.

The sequence $\left\{\frac{1}{k}\right\}_{k=1}^{\infty}$ should be familiar and has, of course, limit 0. We will ask *Maple*to verify this. The *Maple***limit** command, just like **sum** and **prod** has both an inert and active form, with the inert form being the one with the capital 'L'.

really here Maple is performing the function limit evaluation. Need to use assume to get it to do integer.

> $f := k \to \frac{1}{k}; Limit(f(k), k = infinity) = limit(\frac{1}{k}, k = infinity)$

$$f := k \to \frac{1}{k}$$

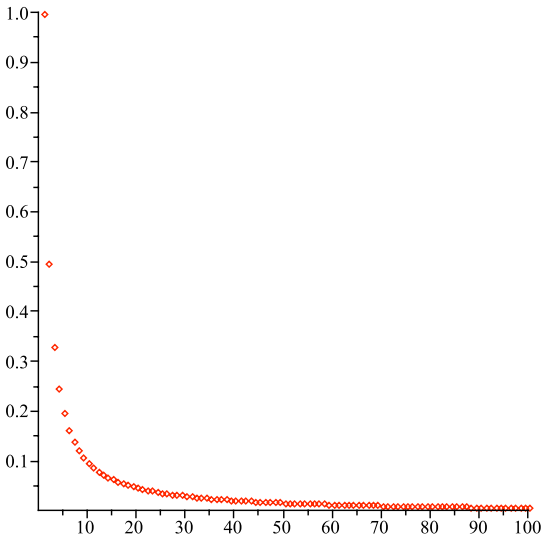
$$\lim_{k \to \infty} f(k) = 0$$

As usual, the **value** command will perform the active calculation on the inter form.

We may also, if we wish a more visual clarification of the convergence, plot this. We could simply just plot the continuous function $\frac{1}{k}$ to see the convergence (using the fact that if the function converges, then the sequence evaluated only at integer points also converges). Instead, however, we will use *Maple*'s point plot option and see only the points of the sequence in our visualisation.

To do this we will construct a sequence (*Maple*sequence, that is) of 2-element lists $[x, y]$ each of which will represent a point in the Cartesian plane. Because we are plotting a sequence, we choose the x-axis to be our index, and the y axis will be the sequence element. As such the points will be $[k, \frac{1}{k}]$. We will look at the first 100 points. This sequence of lists will then be put into a containing list, so the **plot** function does not get confused.

> $plot\left(\left[seq\left(\left[k, \frac{1}{k}\right], k = 1..100\right)\right], style = point\right)$



The convergence is visually pretty clear. It is worth stressing at this point, however, that these plots give an *indication* of convergence, not a proof of convergence. There is always the possibility that the sequence does something odd after the interval we have plotted. So we must still perform regular mathematics to verify the limits, or *at the very least* ask *Maple*to evaluate the limit.

Recall now that an an infinite sum is defined to be a limit of it's partial sums. Mathematically, that is

$$\sum_{k=1}^{\infty} f(k) = \lim_{N\to\infty} \sum_{k=1}^{N} f(k)$$

We have already used *Maple*'s **sum** command to calculate infinite sums for us in Section **??**, but we will do this again now, and demonstrate the limit property. Let us use the series $\frac{1}{k^2}$ again - which we know from the previous section converges to $\frac{\pi^2}{6}$.

```
> sum(1/k^2, k = 1..N);
  Limit(%, N = infinity) = limit(%, N = infinity)
```

$$-\Psi(1, N+1) + \frac{1}{6}\pi^2$$

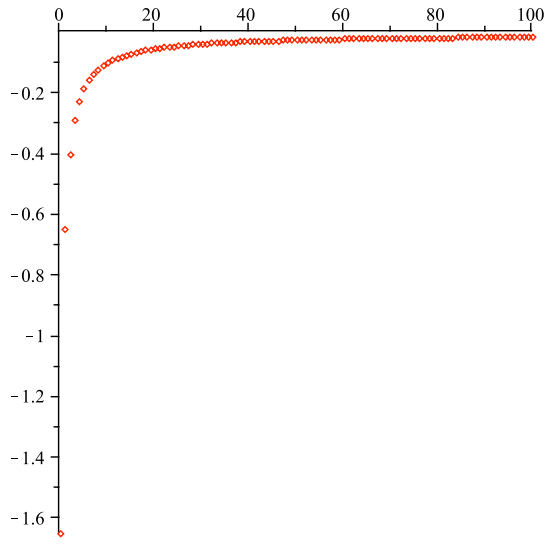
$$\lim_{N\to\infty} \left(-\Psi(1, N+1) + \frac{1}{6}\pi^2\right) = \frac{1}{6}\pi^2$$

We have not seen the Ψ function before, although we may ask *Maple*about it by using the command **?Psi**. We should expect, due to the algebra of limits, that $\lim_{N\to\infty} -\Psi(1, N+1) = 0$. Since it is always good to cross check answers we are unfamiliar with, we shall do so.

```
> limit(-Psi(1, N + 1))
```

$$0$$

> $plot(-\Psi(1, N+1), N = 1..100)$

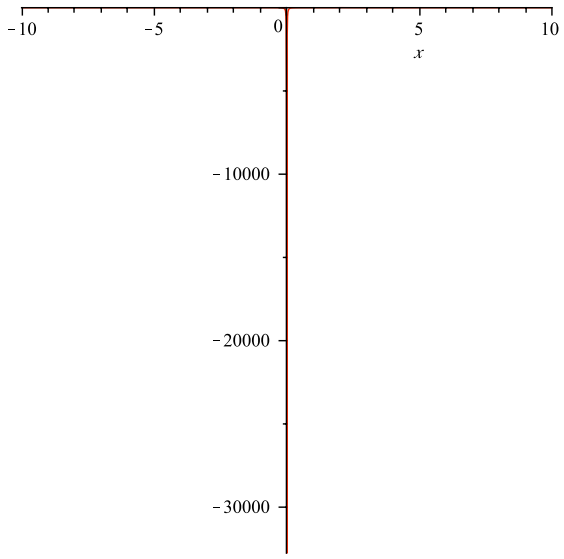


The convergence in the picture seems pretty clear, and combined with the results *Maple*gave us for the infinite sum as well as the limit of the partial sums, and the limit of the Ψ function - not to mention the numeric approximations we saw in the previous section - we may be quite confident of the validity of the answer.

Hmmm, should we introduce a summary of concepts at the end of a chapter? With all this starting out broad, then simplifying, key concepts could get lost. Or should exercises sort that out?

Now let us look at continuous function limits. For this purpose we shall consider the function $f(x) = \frac{1}{x^3-2x^2}$. Our first impulse should be to plot the function to see what it looks like, but doing so produces something not very useful.

> $plot\left(\frac{1}{x^3-2x^2}\right)$



We could attempt to use trial and error to find a good interval to plot over, but instead we shall perform some calculus. Firstly note that the denominator is equal to $x^2(x-2)$, which tells us that the function is not defined at $x = 0$ or $x = 2$ and that, we should probably expect vertical asymptotes at those points. It should also be clear, using the algebra of limits, that

$$\lim_{x\to a}\frac{1}{x^3-2x^2} = \lim_{x\to a}\frac{\frac{1}{x^3}}{1-\frac{2}{x}} = \frac{\lim_{x\to a}\frac{1}{x^3}}{1-\lim_{x\to a}\frac{2}{x}}$$

and so for $a = \pm\infty$ the limit will be 0.

The limits at $a = 0$ and $a = 2$ are only a little bit trickier to work out. We again look at the factored denominator $x^2(x-2)$ and observe that

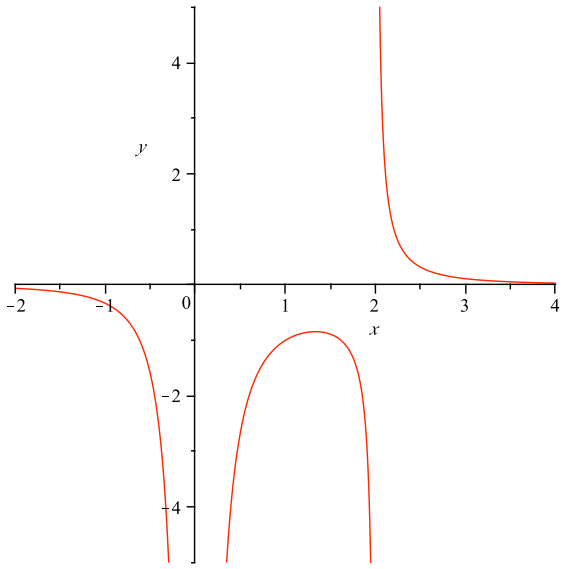
$$\lim_{x\to 0} x^2(x-2) = 0 \text{ and } \lim_{x\to 2} x^2(x-2) = 0$$

Furthermore can see that x^2 will always be ≥ 0, and that for all points near $a = 0$ it is the case that $x - 2 < 0$ so the denominator near $a = 0$ must always be negative. We conclude therefore that the limit at $a = 0$ is $-\infty$. Finally observe that for $x > 2$ we have $x - 2 > 0$ and that for $x < 2$ we have $x - 2 < 0$ which tells us that $f(x) \to -\infty$ as $x \to 2^-$ and so $f(x) \to \infty$ as $x \to 2^+$

We now have plenty of information for us to choose appropriate plotting ranges. In order to put the undefined points (with the vertical asymptotes) evenly spread across the x-axis we will plot across the interval $x \in (-2, 4)$.

However, because we know that the function has vertical asymptotes, we should limit the y axis range somewhat. A quick substitution of $x = 1$ into the function shows that the midpoint between the vertical asymptotes attains the value -1, and so we will take a reasonable guess that $y \in (-5, 5)$ will suffice.

```
> plot(1/(x^3-2*x^2), x=-2..4, y=-5..5, discont=true);
```



The **discont=true** argument to the plot command simply tells **plot** that we are plotting a discontinuous function allowing for a better plotting. Without that argument, **plot** assumes it is plotting a continuous function and ends up putting vertical lines between discontinuous points, or at asymptotes.

We will now verify the limits with *Maple*

```
> f := 1/(x^3-2*x^2);
  limit(f, x = -infinity), limit(f, x = 0), limit(f, x = 2), limit(f, x = infinity)
```

$$f := \frac{1}{x^3 - 2x^2}$$

$$0, -\infty, undefined, 0$$

The undefined limit at $a = 2$ should be completely unsurprising thanks to the plot we performed above. The limit from below and the limit from above are not the same. Recall that if $\lim_{x\to a^-} = \lim_{x\to a^+} = L$ if and only if $\lim_{x\to a} = L$. Fortunately for us *Maple*can handle single directional limits by allowing the **limit** command to take a third input variable for this purpose, which may be one of **left** or **right**. As might be expected, the left limit at a is when $x \to a^-$ and the right limit at a is when $x \to a^+$.

Exercise Ideas: Limits returning ranges. Left and right limits of discontinuous functions that are different, and not $\pm\infty$

```
> Limit(f, x = 2, left) = limit(f, x = 2, left);
  Limit(f, x = 2, right) = limit(f, x = 2, right);
```

$$\lim_{x\to 2^-} \frac{1}{x^3 - 2x^2} = -\infty$$

$$\lim_{x\to 2^+} \frac{1}{x^3 - 2x^2} = \infty$$

We will now return to sequence limits, and a cautionary example. *Maple*'s **limit** function calculates real or complex valued limits. When we used it to calculate sequence limits above, what we were really doing was evaluating the real-valued limit at infinity of the functions in question, and using the Theorem that if $f(x) \to L$ as $x \to \infty$ exists for a real valued function f, then the sequence $\{f(n)\}_{n\in\mathbb{N}}$ converges to the same limit as $n \to \infty$.

However, caution is advised. This relationship between limits of functions and limits of sequences does not work the other way around. Consider the function

$$f(x) = x\sin(Pi \cdot x) + 2\tanh(x - 5)$$

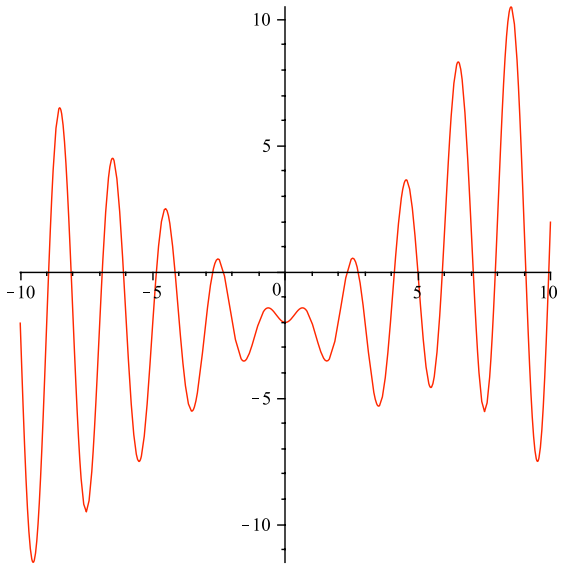
. If we evaluate limits or plot the function we might very well be tempted to conclude that the sequence does not converge.

```
> f := x → x · sin(Pi · x) + 2 · tanh(x − 5);
  limit(f(x), x = infinity)
```

$$f := x \to x \cdot \sin(\pi \cdot x) + 2 \cdot \tanh(x - 5)$$

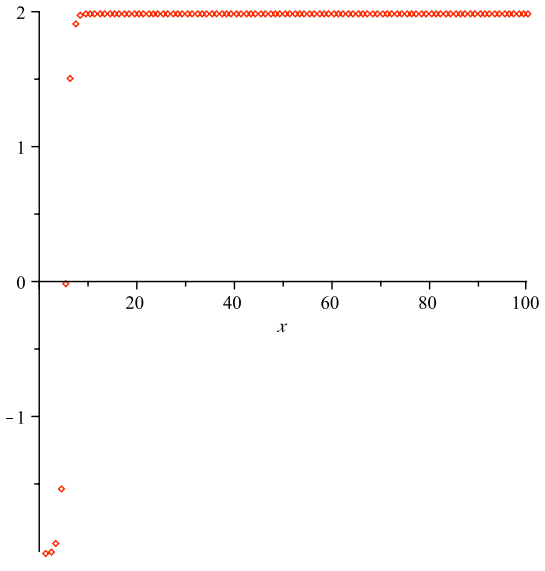
$$undefined$$

```
> plot(f)
```



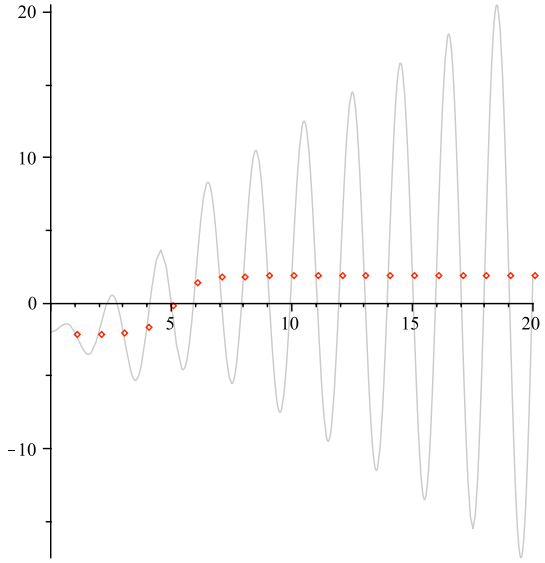
However, our calculus knowledge should tell us that $n\sin(\pi n) = 0$ for $n \in \mathbb{N}$, and since $\tanh(x) \to 1$ as $x \to \infty$ it must be the case that $2\tanh(x-5) \to 2$ as $x \to \infty$. In short, the sequence $\{f(n)\}_{n\in\mathbb{N}}$ ought to converge to the value 2. Plotting only the sequence points confirms this.

```
> plot([seq([n, f(n)], n = 1..100)], style = point)
```



If we plot the real-valued function and the sequence on the same axes, we can see the convergence a little better.

```
> p1 := plot(f, 0..20, colour = grey) :
  p2 := plot([seq([n, f(n)], n = 1..100)], style = point) :
  plots[display]([p1, p2])
```



So all that remains is to see if we may convince *Maple*to provide us with the correct answer for the sequence limit. Since we are evaluating a limit at infinity we are already evaluating a left limit and cannot possibly try to change to a right limit or a bi-directional limit. A good attempt would be to use the **assuming** modifier.

```
> limit(f(n), n = infinity) assuming n :: posint
```

$$undefined$$

Unfortunately, as it turns out, due to n being a dummy variable inside the **limit** function, the assumptions we requested for n do not quite seem to be being applied. See **?assuming** for more details. The solution is to use the **assume** command, which works in a similar way to **assuming** except that the assumptions become globally accepted for the entire worksheet (as opposed to just the current command). So we will reset the n variable after performing our calculation.

```
> assume(n :: posint); limit(f(n), n = infinity); n :=' n' :
```

$$2$$

To reiterate the key points here, the lack of a limit of a real valued function does not imply the lack of a limit of a sequence of evaluations of that function, and - more importantly - one must always keep their wits about them

when using a CAS.

2.1.4 Differentiation

Differentiation is, fundamentally, all about calculating rates of change. Recall that the derivative of a function, $f(x)$ say, at any point a is the slope of the tangent line to f at the point a. Recall, also, that the tangent at a is defined to be the line through a with slope equal to the limit of the slopes of lines drawn between a and points near a. As depicted in Figure 2.1. So

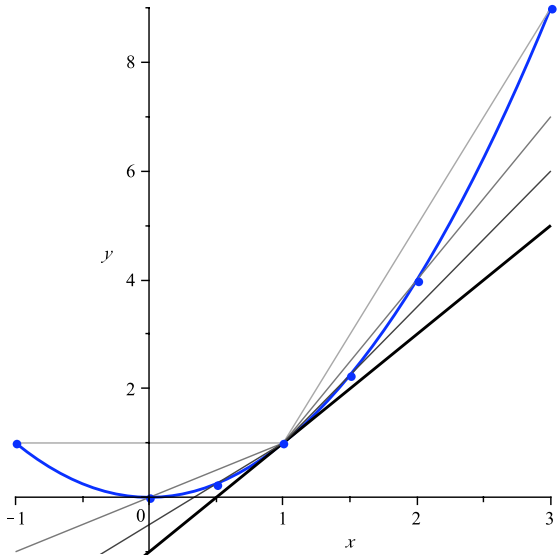


Figure 2.1: Depiction of the convergence of lines to the tangent

it is that we come to the limit definition of the derivative at a point a as

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a} = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

.

Let us explore this a little before we introduce *Maple*'s native differentiation code. We'll start with a parabola $f(x) = x^2$ and so, of course, we know that $f'(x) = 2x$ meaning that the tangent to the parabola at any point a has slope $2a$. We shall write a small procedure to output the limit and its value.

```
> d := proc(f :: procedure, a)
  Limit((f(a+h)-f(a))/h, h=0) =
  limit((f(a+h)-f(a))/h, h=0)
  end:
```

```
> d(x -> x^2, 1); d(x -> x^2, 2); d(x -> x^2, x)
```

$$\lim_{h\to 0} \frac{(1+h)^2-1}{h} = 2$$

$$\lim_{h\to 2} \frac{(2+h)^2-4}{h} = 4$$

$$\lim_{h\to 0} \frac{(x+h)^2-x^2}{h} = 2x$$

If we look a little closer at the final limit, we should see that $(x+h)^2 - x^2 = x^2 - 2hx - h^2 - x^2 = h(2x-h)$ and so the limit then becomes

$$\lim_{h\to 0} \frac{h(2x-h)}{h} = \lim_{h\to 0}(2x-h) = 2x$$

thus verifying both *Maple*'s limit calculation and our regular differentiation technique (for the parabola, at least).

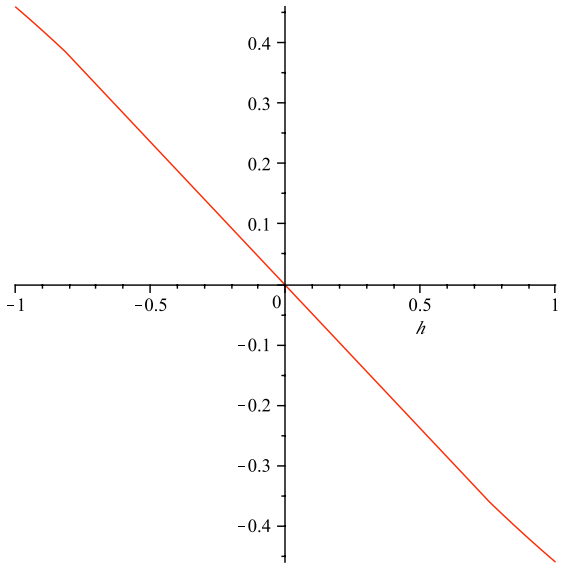
Looking quickly at a trigonometric function, sin at point $\frac{\pi}{2}$ we see a limit that is rather harder to work out on paper.

```
> d(sin, Pi/2);
```

$$\lim_{h\to 0} \frac{\cos(h)-1}{h} = 0$$

As ever, our instinct should be to plot the function. However, to be on the safe side, we will ask for a lot of sample points, to try and get a good idea of the behaviour near the $h = 0$ point which we know is undefined.

```
> plot((cos(h) - 1)/h, h = -1..1, numpoints = 1000)
```



Of course, in order to perform differentiation in *Maple*we need not perform limit calculations each and every time. There are two functions for obtaining the derivative of a function named **diff** and **D**. The difference between them is that **diff** takes an expression as input and produces an expression as output, and **D** takes a function as an input and produces a function as an output. It should also be mentioned that **diff** also has an inert form **Diff**.

```
> diff(x^2 + x + 1, x); diff(exp(2x), x); Diff(sin(x), x) = diff(sin(x), x)
```

$$2x + 1$$
$$2e^{2x}$$
$$\frac{d}{dx}\sin(x) = \cos(x)$$

Notice that we need to tell **diff** which variable we are taking the derivative with respect to. For similar differentiation using the **D** function, we have the following.

```
> D(x -> x^2 + x + 1); D(exp); D(sin)(Pi)
```

$$x \to 2x + 1$$
$$\exp$$
$$-1$$

Note that since **D** produces a function, we may immediately pass input to it

for evaluation, as we did in the last calculation, above (which quite correctly calculated $\cos(\pi) = -1$).

If we wish to calculate a second derivative, we do the following

> $p := x \to x^2 + x + 1; diff(p(x), x, x); D^{(2)}(p);$

$$p := x \to x^2 + x + 1$$
$$2$$
$$2$$

Note that the parentheses around the 2 in $D^{(}2)$ are very important, and their omission will cause the command to fail. In general, for the k^{th} derivative we use $diff(f(x), x, x, \ldots, x)$ where the sequences of x's is of length k (which we may abbreviate $x\$k$), or we use $D^{(k)}(f)$.

> $p := 4x^5 - 3x^2 + x - 2:$
> **for** k **from** 1 **while** $diff(p(x), x\$(k-1)) \neq 0$ **do**
> $Diff(p(x), x\$k) = D^{(k)}(p)(x)$
> **od**

$$\frac{d}{dx}(4x^5 - 3x^2 + x - 2) = 20x^4 - 6x + 1$$
$$\frac{d^2}{dx^2}(4x^5 - 3x^2 + x - 2) = 80x^3 - 6$$
$$\frac{d^3}{dx^3}(4x^5 - 3x^2 + x - 2) = 240x^2$$
$$\frac{d^4}{dx^4}(4x^5 - 3x^2 + x - 2) = 480^x$$
$$\frac{d^5}{dx^5}(4x^5 - 3x^2 + x - 2) = 480$$
$$\frac{d^6}{dx^6}(4x^5 - 3x^2 + x - 2) = 0$$

2.1.5 Integration

Integration grows out of the problem of calculating area underneath a curve - although its applications are far more wide and varied than that. Recall that a definite integral of a function f between two points a and b may be approximated by a limit of rectangles.

$$\int_a^b f(x)\,dx = \lim_{n\to\infty}\left(\Delta x \sum_{k=1}^{n} f(a + k\Delta x)\right) \text{ where } \Delta x := \frac{b-a}{n}$$

In fact, the approximation can be made from rectangles coming from an arbitrary partitioning of the interval (a, b) with the heights of the rectangles

being taken from arbitrary points within each of the partition elements. See [11] or any elementary calculus text for more details.

```
> integral := proc(f :: procedure, r :: range) local Delta, a, b;
  a, b := lhs(r), rhs(r); Delta := (b-a)/n
  Limit(Delta * Sum(f(a + k * Delta), k = 1..n), n = infinity) =
  limit(Delta * sum(f(a + k * Delta), k = 1..n), n = infinity)
  end :
```

```
> integral(x -> x^2, 0..2); integral(sin, 0..Pi); integral(sin, 0..2 * Pi)
```

$$\lim_{n\to\infty}\left(\frac{2\left(\sum_{k=1}^{n}\frac{4k^2}{n^2}\right)}{n}\right)=\frac{8}{3}$$

$$\lim_{n\to\infty}\left(\frac{\pi\left(\sum_{k=1}^{n}\sin\left(\frac{k\pi}{n}\right)\right)}{n}\right)=2$$

$$\lim_{n\to\infty}\left(\frac{2\pi\left(\sum_{k=1}^{n}\sin\left(\frac{2k\pi}{n}\right)\right)}{n}\right)=0$$

Once this limit is established, then we are presented with the Fundamental Theorem of Calculus which states that

$$\int_a^b f(x)\,dx = F(b) - F(a) \text{ where } F' = f$$

and nicely links differentiation and definite integrals. We also, by convention, denote the indefinite integral

$$\int f(x)\,dx = F(x) \Leftrightarrow F' = f$$

We may use this to check the limits found above. Firstly we have $\frac{x^3}{3}$ as an antiderivative of x^2, and evaluating $\frac{2^3}{3}-\frac{0}{3}=\frac{8}{3}$ verifies the integral. Similarly we have $-cos$ as an antiderivative of sin leading us to $-cos(\pi)-(-cos(0)) = 1+1 = 2$ and $(-cos(2\pi)-(-cos(0))) = -1+1 = 0$. In fact, the final integral can be verified by the symmetric nature of the cosine graph between 0 and 2π.

Again, as with differentiation, we need not perform limit calculations within *Maple*if we wish to calculate an integral, as there is a handy function named

int (and its inert form **Int**). The **int** function can handle both definite and indefinite integrals and takes an expression as it's first argument, and a range as its second argument in the form $var = a..b$ where var is the variable to integrate over. In the case of an indefinite integral, the second parameter is just var with no range.

```
> Int(x^2, x = 1..3) = int(x^2, x = 1..3); Int(sin(x), x) = int(sin(x), x)
```

$$\int_1^3 x^2 \, dx = \frac{26}{3}$$

$$\int sin(x) \, dx = -cos(x)$$

The **int** command will also accept function (as opposed to expression) input for its first parameter in precisely the same way as the **plot** function does - by omitting the $var =$ from the second parameter. However, it can only do this for definite integrals, and the inert form does not behave very well this way.

```
> Int(sin, 0..Pi) = int(sin, 0..Pi)
```

$$Int(sin, 0..Pi) = 2$$

```
> value(lhs(%))
```

$$2$$

2.2 Univariable Calculus

2.2.1 Optimisation

Suppose we wish to calculate the longest ladder that we may carry around a corner with one corridor 2m in width, and the other 1m in width. This is an example of an optimisation problem. We may use calculus (specifically, differentiation) to solve problems along these lines.

Figure **??** depicts the problem. Given any angle θ we we know that

$$x = \frac{1}{\cos(\theta)} \text{ and } y = \frac{2}{\sin(\theta)}$$

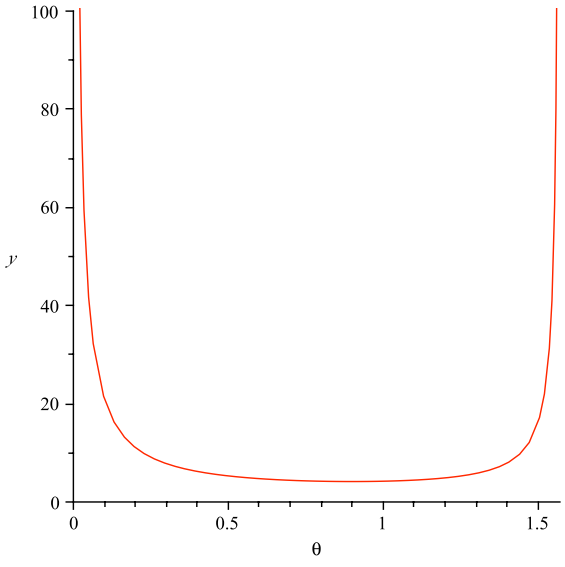
Therefore, the length, L, of a ladder that touches the corner, and the opposite walls of each corridor at an angle of θ to the corner is given by the formula

$$L := \frac{1}{\cos(\theta)} + \frac{2}{\sin(\theta)}$$

We can see straight away that $0 \leq \theta \leq \frac{\pi}{2}$ and that $(\cos(\theta))^{-1} \to \infty$ as $\theta \to 1^-$ as well as $(\sin(\theta))^{-1} \to \infty$ as $\theta \to 0^+$ which tells us that our L function will tend toward infinity at its endpoints. We also know that $L > 0$ for all values of θ—just as we would expect for the length of a ladder. Let us plot the function, being sure to limit the y-axis.

```
> L := theta -> 1/cos(theta) + 2/sin(theta);
  plot(L(theta), theta = 0..Pi/2, y = 0..100)
```

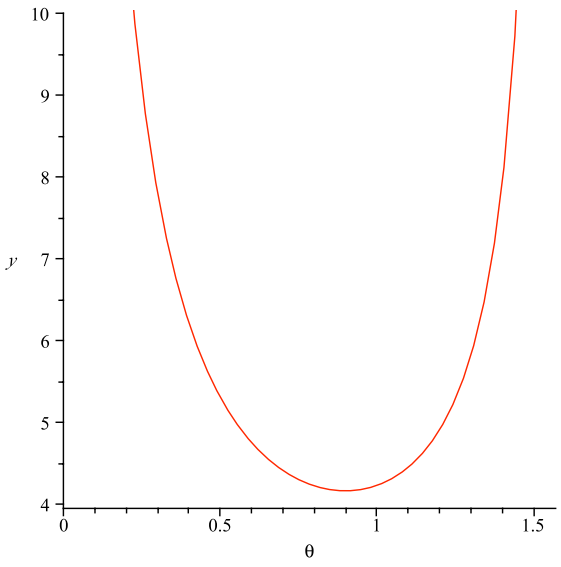
$$L := \theta \to \frac{1}{\cos(\theta)} + \frac{2}{\sin(\theta)}$$



What is interesting here is that the function is concave up and has no maximum values, although it does have a minimum value. What's going on here? Well our function L, as we know, calculates the length of a ladder at an angle of θ that touches both the far walls of the two corridors as well as the corner. Such a ladder is the longest possible ladder that can rest at that particular angle. However, if a ladder is to be carried around the corner, then it must go through all angles from $\theta = 0$ to $\theta = \frac{\pi}{2}$ and not become stuck. What we are, in essence, looking for is the shortest of all such longest ladders, and hence a minimum of the function L.

With a little trial and error, we may find the following plot and see that the minimum value lies somewhere in the range $\frac{\pi}{4} \leq \theta \leq 1$.

```
> plot(L(theta), theta = 0..2*Pi, y = 4..10)
```




```
> D(L)(theta); solve(% = 0)
```

$$\frac{\sin(\theta)}{\cos(\theta)^2} - \frac{2\cos(\theta)}{\sin(\theta)^2}$$

$$\arctan(2^{1/3}), -\arctan\left(\frac{1}{2}\,2^{1/3} - \frac{1}{2}I\sqrt{3}\,2^{1/3}\right), -\arctan\left(\frac{1}{2}\,2^{1/3} + \frac{1}{2}I\sqrt{3}\,2^{1/3}\right)$$

Well, that's a bit of a mess. *Maple*seems to have given us three solutions, two of which appear to be complex. Nonetheless, there is clearly a real solution there, and evaluating it numerically shows it to be in the range we were expecting.

```
> evalf(arctan(2^(1/3))); L = L(arctan(2^(1/3))); evalf(%)
```

$$0.8999083481$$

$$L = \sqrt{2^{2/3}+1} + 2^{2/3}\sqrt{2^{2/3}+1}$$

$$L = 4.161938184$$

And so there we have it. A ladder of approximately 4.16 metres in length is the longest we may carry around the corner. We can see quite well from the plots above that this is clearly a local minimum, however a quick 2nd derivative test won't hurt.

```
> D^(2)(L)(arctan(2^(1/3))); is(% > 0); evalf(%%)
```

$$3\,2^{2/3}\sqrt{2^{2/3}+1}+3\sqrt{2^{2/3}+1}$$

$$\mathit{true}$$

$$12.48581455$$

2.2.2 Integral Evaluation

Integral evaluation can, at times, be quite tricky. A tool like *Maple*can indeed be an asset, however even it may be unable to symbolically perform certain integrals. For example, suppose we ask *Maple*to integrate xe^{x^3} between 0 and 1.

```
> int(x · exp(x^3), x = 0..1)
```

$$\int_0^1 xe^{x^3}\,dx$$

Notice here that we definitely used the active form of the integral command, and yet *Maple* still returned only the integral back again. It seems that *Maple* cannot provide a better symbolic answer than the integral itself. The use of hand-methods should prove equally frustrating. (Note here that the integral of xe^{x^2} should be trifling easy). At times like this, all one can really do is ask for a numerical answer

```
> evalf[50](int(x · exp(x^3), x = 0..1)); identify(%);
```

$$0.78119703110865591510743281434829950577669739096218$$

$$0.78119703110865591510743281434829950577669739096218$$

The use of the **identify** is a request to *Maple* to take a good guess at a likely symbolic representation of the given decimal number. In this case **identify** simply gives us back the decimal number, meaning that it could not find any likely symbolic representation. Similarly, Sloane's encyclopedia of integer sequences, and the Inverse Symbolic Calculator also turn up nothing. It looks like we're stuck with just a numerical approximation of the integral.

Suppose we wish to evaluate the integral

$$\int_0^{\pi} \frac{x\sin(x)}{1+\cos^2(x)}\,dx$$

Well, our first attempt should be to see what *Maple*thinks.

$$\begin{bmatrix} > \ simplify\left(int\left(\frac{x\cdot\sin(x)}{1+\cos(x)^2}, x=0..\,\mathrm{Pi}\right)\right) \\ \frac{1}{4}\pi^2 - dilog\left(\frac{\sqrt{2}-1-I}{\sqrt{2}-1}\right) + dilog\left(\frac{1-I+\sqrt{2}}{1+\sqrt{2}}\right) \\ - dilog\left(\frac{1+I+\sqrt{2}}{1+\sqrt{2}}\right) + dilog\left(\frac{\sqrt{2}-1+I}{\sqrt{2}-1}\right) + I\pi\ln(1+\sqrt{2}) \end{bmatrix}$$

That's not really very useful. Let's try and evaluate that mess as a decimal number.

$$\begin{bmatrix} > \ evalf[50](\%) \\ 2.4674011002723396547086227499690377838284248518102 - 3.\ 10^{-49} I \end{bmatrix}$$

Hmmm, that's a complex number even if the complex part is infinitesimal. Let's try again, and see if asking *Maple* to evaluate the integral numerically works any better. We'll even throw in an **identify** for good measure.

$$\begin{bmatrix} > \ int\left(\frac{x\cdot\sin(x)}{1+\cos(x)^2}, x=(0.)..\,\mathrm{Pi}\right); identify(\%) \\ 2.4674011002723396547086227499690377838284248518102 \\ \\ \frac{1}{4}\pi^2 \end{bmatrix}$$

So we have it that numerically evaluating the integral, and asking *Maple* to guess a symbolic value gives us that the integral is very probably equal to $\frac{1}{4}\pi^2$. At this point we would start looking for a more formal proof. Fortunately, in this case, it can be shown that

$$\int_0^\pi x f(\sin(x))\,dx = \frac{\pi}{2}\int_0^\pi f(\sin(x))\,dx$$

and it should be pretty clear that

$$\frac{\sin(x)}{1+\cos^2(x)} = \frac{\sin(x)}{2-\sin^2(x)} = f(sin(x)) \text{ where } f = \frac{x}{2-x^2}$$

which leaves us with the rather simpler integral

$$\int_0^\pi \frac{x\sin(x)}{1+\cos^2(x)}\,dx = \frac{\pi}{2}\int_0^\pi \frac{\sin(x)}{1+\cos^2(x)}\,dx$$

As it happens, this new integral is easier for *Maple* to handle.

```
> Pi/2 * int(sin(x)/(1+cos(x)^2), x=0..Pi)
```

$$\frac{1}{4}\pi^2$$

Let us perform another integral. This time we will evaluate

$$\int_0^\infty \frac{x^2}{\sqrt{e^x-1}}\,dx$$

As a first attempt, we will see what *Maple* makes of the integral.

```
> int(x^2/sqrt(exp(x)-1), x=0..infinity)
```

$$\int_0^\infty \frac{x^2}{\sqrt{e^x-1}}\,dx$$

unfortunately *Maple* apparently cannot evaluate this integral. Our next course of action will be to attempt numeric evaluation.

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

$$16.37297620$$

$$\frac{9}{2}\,2^{3/5}\,\sqrt{3}\,\zeta(5)^9$$

and we have a possible value for the integral. Remember, however, that the **itentify** function gives a *probable* symbolic expression but not a certain one. Some more work will need to be performed in order to prove the identity with certainty.

We will have a closer look at the integral now. Looking at the denominator, $\sqrt{e^x-1}$, we see that a substitution of $x = \log(u)$ will change the denominator to $\sqrt{u-1}$ thus eliminating the exponent term. Performing this substitution, we see that

$$\frac{dx}{du} = \frac{1}{u} \Rightarrow dx = \frac{du}{u}$$

It is also the case that $\log(0) = 1$ and $\log(x) \to \infty$ as $x \to \infty$ so that

$$\int_0^\infty \frac{x^2}{\sqrt{e^x-1}}\,dx = \int_1^\infty \frac{\log(u)^2}{u\sqrt{u-1}}\,du$$

Turning again to *Maple* with this new integral we find altogether better success than we did previously.

```
> int(log(u)^2/(u*sqrt(u-1)), x = 0..infinity)
```

$$\frac{1}{3}\pi^3 + 4\,\pi\,\ln(2)^2$$

interestingly, however, we have an altogether different symbolic answer to the one produced by **identify**, above. It is prudent then, to perform a sanity check

```
> evalf(%)
```

$$16.37297620$$

which is precisely the same 10-digit value we obtained from our first attempt. This is absolutely expected, of course, since substitutions do not alter the value of an integral.

As an epilogue to this example, it turns out that if we obtain even one extra digit of precision for this integral, then *Maple* is completely unable to identify the decimal approximation.

```
> evalf[11](int(x^2/sqrt(exp(x)-1), x = 0..infinity)); identify(%)
```

$$16.372976196$$
$$16.372976196$$

and, indeed, any precision greater than 11 digits, it seems that *Maple* is unable to identify the number. Exploring this a little more closely

```
> evalf(int(x^2/sqrt(exp(x)-1), x = 0..infinity) - 9/2*2^(3/5)*sqrt(3)*Zeta(5)^9)
```

$$-4.\ 10^{-8}$$

we see that the two differ at, approximately, 8 decimal places. It would seem that **identify** was confused by not having enough decimal digits of the number in question to work with.

At least two things should be readily apparent from these examples. Firstly and foremost is that *Maple* is not a substitution for poor calculus skills—or at least is a poor one. Secondly, answers—especially from the **identify** function—should be checked from a number of avenues before being accepted. Human/machine collaboration is the name of the game here. *Maple* can be wonderful for performing tedious calculations quickly, and is remarkably adept at performing integral calculus, but a correct substitution, or other mathematical insight on our part can mean the difference between successfully obtaining a symbolic answer, or not.

2.2.3 Differential Equations

Differential equations are equations that relate a function to its derivatives. A solution to a differential equation is a function who has the required relationship with its derivatives. The simplest differential equation is $y' = y$ which has the solution $y = e^x$. This should be nothing new anybody who has studied first year calculus.

An exact linear differential equation is a differential equation of the form

$$y' + P(x)\, y = Q(x)$$

Such differential equations can be solved by use of an *integrating factor*

$$I(x) := e^{\int P(x)\, dx}$$

Which has the property that

$$\int I(x)\, y' + I(x)\, P(x)\, y\, dx = I(x)\, y$$

and allows us to solve the equation by multiplying left hand side and right hand side by the integrating factor, integrating both sides, and solving for y.

It is fairly simple to verify the claimed property of the integrating factor by hand. We will check it in *Maple*.

```
> IF := exp(int(P(x), x));
```

$$IF := \mathrm{e}^{\int P(x)\, dx}$$

```
> int(IF * diff(y(x), x) + IF * P(x) * y(x))
```

$$\mathrm{e}^{\int P(x)\, dx}\, y(x)$$

```
> diff(IF * y(x))
```

$$\mathrm{e}^{\int P(x)\, dx} \left(\frac{d}{dx}\, y(x) \right) + \mathrm{e}^{\int P(x)\, dx}\, P(x)\, y(x)$$

Of course, we only needed to check one of these identities, the other one we get for free with the Fundamental Theorem of Calculus.

With this identity available to us, it is easy to see that the solution to the exact differential equation is

$$y' + P(x)\,y = Q(x) \implies y = \frac{\int I(x)\,Q(x)\,dx}{I(x)}$$

So if we consider the exact linear differential equation $xy' + y = 3x^3$, which may be rewritten as $y' + x^{-1}y = 3x^2$ then we may use *Maple*

> $P := x \to \frac{1}{x}; Q := x \to 3 \cdot x^2; IF := \exp(int(P(x), x))$

$$P := x \to \frac{1}{x}$$
$$Q := x \to 3\,x^2$$
$$IF := x$$

> $\frac{int(IF \cdot Q(x), x) + C}{IF}$

$$\frac{\frac{3}{4}\,x^4 + C}{x}$$

Note, however, that for a general solution we needed to manually add the constant of integration when calculating the answer, since *Maple*'s **int** function does not include this constant.

We may cross-check this using *Maple*'s inbuilt differential equation solving function **dsolve**.

> $dsolve(x \cdot diff(y(x), x) + y(x) = 3 \cdot x^3)$

$$y(x) = \frac{\frac{3}{4}\,x^4 + \_C1}{x}$$

Moving on now to second-order differential equations. A *second-order linear differential equation* is a differential equations of the form

$$P(x)\,y'' + Q(x)\,y' + R(x)\,y = G(x)$$

and is furthermore said to be *homogeneous* if $G(x) = 0$. Finally, if the functions $P(x)$, $Q(x)$ and $R(x)$ are constant functions then the differential equation is said to have *constant coefficients*, however if the recurrence relation is still to be second-order then $P(x) \neq 0$.

Homogeneous second-order linear differential equations with constant coefficients may be solved in a manner almost identical to that used to solve

homogeneous second-order linear recurrence relations with constant coefficients, which we looked at in Section 1.3.3.

Given the equation $ay''+by'+c=0$ we construct the characteristic equation $at^2+bt+c=0$ and solve for t. There are only three possibilities for the roots r_1, r_2 of the equation. The general formula is as follows

$$y(x) = \begin{cases} Ae^{r_1 x} + Be^{r_2 x} & r_1 \neq r_2 \\ Ae^{r_1 x} + Bxe^{r_2 x} & r_1 = r_2 \\ e^{\alpha x}\left(A\sin(\beta x) + B\cos(\beta x)\right) & r_1, r_2 = \alpha \pm i\beta \end{cases}$$

where A and B are arbitrary constants.

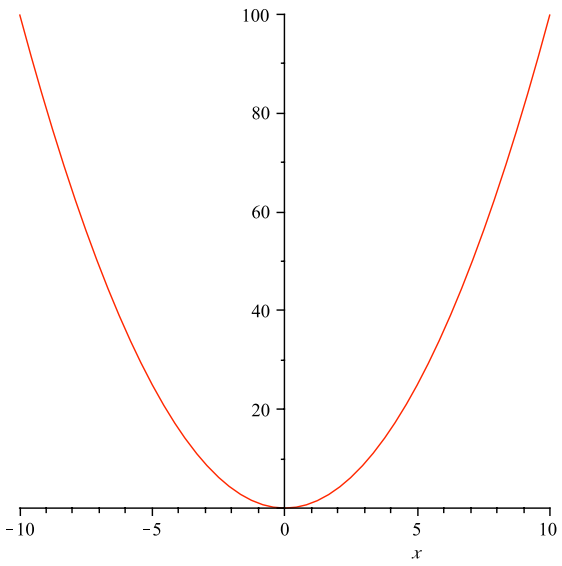
2.2.4 Parametric Equations, Alternative Coordinates, and other esoteric Plotting fun

Recall that when a graph is plotted, what we are seeing is a graphical representation of pairs of points that satisfy some relationship. The parabola, for example, is the collection of all points (x, y) in $\mathbb{R} \times \mathbb{R}$ where $y = x^2$.

We may also plot functions from parametric equations, where a parameter, t say, varies, and the points in the plot are of the form $(x, y) = (x(t), y(t))$. There is no necessarily a relationship between the x and y co-ordinates in a parametric equation—appart from the fact that they share the same t-value. Of course, any function $f(x)$ may be turned into a parametric equation $(x, y) = (t, f(t))$.

Maple's **plot** function will allow us to plot parametric equations, if we so wish. To do so, we much pass a 3-element list as the first argument to the **plot** function, instead of the usual expression. This list must be in the form of $[x(t), y(t), t = a..b]$ where $x(t)$ and $y(t)$ are arbitrary expressions in the variable t. Of course, like everything else in *Maple*, we are not confined to using t as our variable if we do not wish. For example, to plot our parabola using parametric equations, we would input the following.

> $plot([t, t^2, t = -10..10])$



Maple automatically scales the horizontal and vertical axes to fit the plot. However one should be careful, the x and y ranges *are independent* of the parameter range, and all three may be modified seperately. In the case of our parabola, there is a very direct relationship between the parameter t, and the horizontal and vertical ranges. This need not be so.

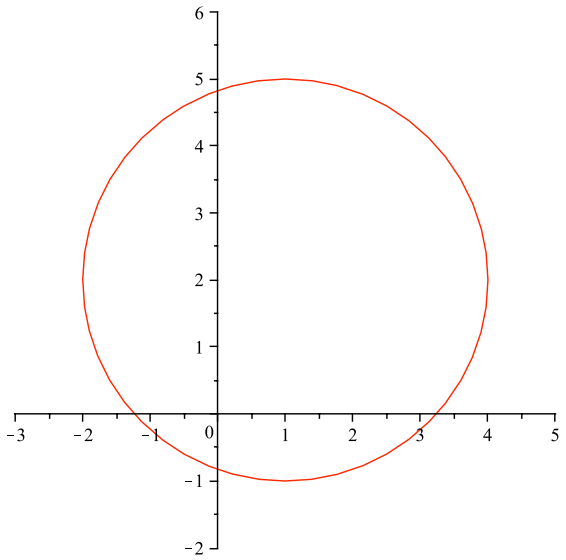
As an example, recall the parametric equations of a circle; $(x, y) = (r\cos\theta, r\sin\theta)$ where r is the radius, and our parameter $\theta \in [0, 2\pi]$. Actually, this is not completely general, but is the parametric equations for a circle centred at the origin. For a circle centred at an arbitrary point, (x_0, y_0) say, then our parametric equations are

$$(x, y) = (x_0, y_0) + (r\cos\theta, r\sin\theta) = (x_0 + r\cos\theta, y_0 + r\sin\theta)$$

This is easiest to see by treating these parametric equations as vector equations.

We will plot a circle, centred at $(1, 2)$, with radius 3. For such a plot, we should expect the horizontal range to be $-2..4$ and the vertical range to be $-1..5$. In order to demonstrate the independent nature of these ranges, we will specifically ask for the plot for a larger range.

> $plot([1 + 3 \cdot \cos(t), 2 + 3 \cdot \sin(t), t = 0..2 \cdot \mathrm{Pi}], -3..5, -2..6)$



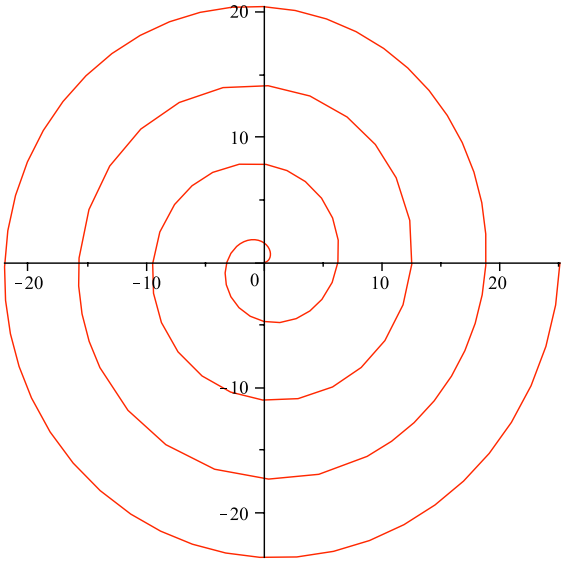
We can clearly see the circle neatly within the ranges we expected, and yet *Maple* has happily plotted with the extra range we requested.

In a lot of, if not most, cases separately modifying the horizontal, vertical, and parameter ranges will not be necessary. Nonetheless it is worth being aware of the fact that they may be independently specified.

The circle example demonstrates a very useful feature of parametric plots, which is that they may be used to plot curves that are not the result of functions. The circle is clearly not a function, as it violates the vertical line test. Recall that a function always associates a single value in the range for each single value in the domain. This is not the case with a circle, we cannot assign a function $y = f(x)$ which will produce all the points in a circle.

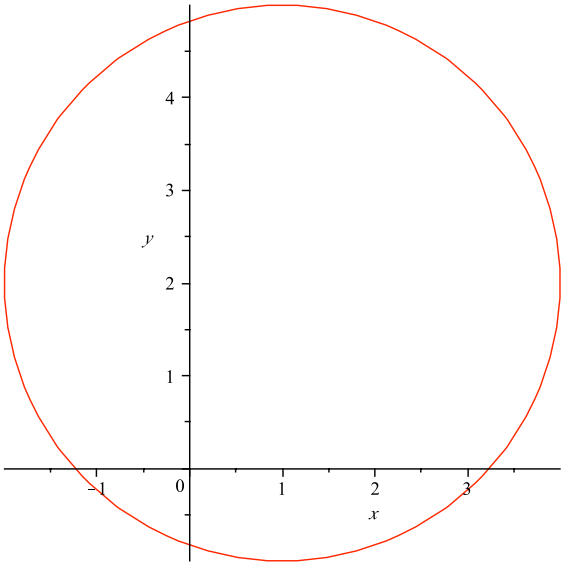
To further illustrate this advantage of parametric equations, we will plot a spiral using parametric equations. We will use the parametric equations $(x, y) = t\cos t, t\sin t$. This varies from the circle in that the radius is no longer fixed. If we think of the points (x, y) as vectors, then each point on the line is a vector of length t and angle t. As t increases, then the angle will cycle, but the length will continue increasing. We will plot four full revolutions of this spiral.

> $plot([t \cdot \cos(t), t \cdot \sin(t), t = 0..8\,\mathrm{Pi}])$



Returning to our circle plots, the reader may well recall that whilst there is no explicit function for a circle, there most certainly is an equation which gives an implicit function for the circle. That equation is, of course, $(x - x_0)^2 + (y - y_0)^2 = r^2$ where (x_0, y_0) is the centre of the circle and r is the radius. One may well wonder if *Maple* can plot such implicit equations. The answer is that yes it can, although we need to use a special function in the plots package, which goes by the name of **implicitplot**.

> $plots[implicitplot]\left((x-1)^2 + (y-2)^2 = 9, x = -3..5, y = -2..6\right)$



Notice that even though we asked for the plot to be in the extra range, just as we did with the parametric plot above, the **implicitplot** function plotted the circle to its extremities, and no more. It would seem that when evaluating implicit equations, *Maple* evaluates all the pairs (x, y) within the range that satisfy the equation, and then works out the required scale for the axes.

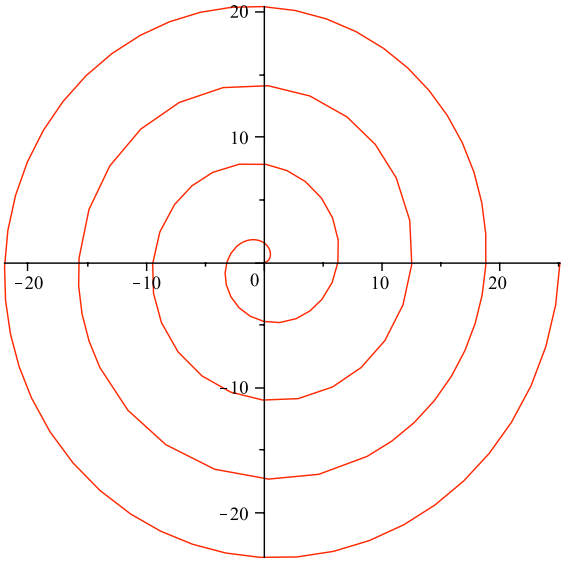
So far we have considered only Cartesian co-ordinates (of the form (x, y) in relation to two axes) when plotting. Another common co-ordinate system is the so-called *polar* co-ordinates, which are co-ordinates of the form (r, θ) where θ is the angle, and r is the distance travelled in that direction. The polar co-ordinates $(\sqrt{2}, \frac{1}{4}\pi)$, for example, correspond to the Cartesian co-ordinates $(1, 1)$. We may freely convert between polar and Cartesian co-ordinates with the identites $r = \sqrt{x^2 + y^2}$, $x = r\cos\theta$ and $y = r\sin\theta$. These identities may easily be confirmed by drawing up a trigonometric triangle.

Maple will happily plot polar equations (that is equations given in terms of polar co-ordinates) for us either on a regular Cartesian pair of axes, or on a special background more suited to the polar co-ordinates. The latter requires a special function in the plots package. Polar plots expect an expression for r to be a function of θ (just as regular plots expect an expresion for y as a function of x). This should be unsurprising, given that polar equations are usually written as $r = r(\theta)$. Let us start with a circle, as it is quit simple. A circle contains points that are equidistant from it's centre, hence r is the constant radius, and θ may take any value. So our polar equation is simply $r = C$ where C is a constant.

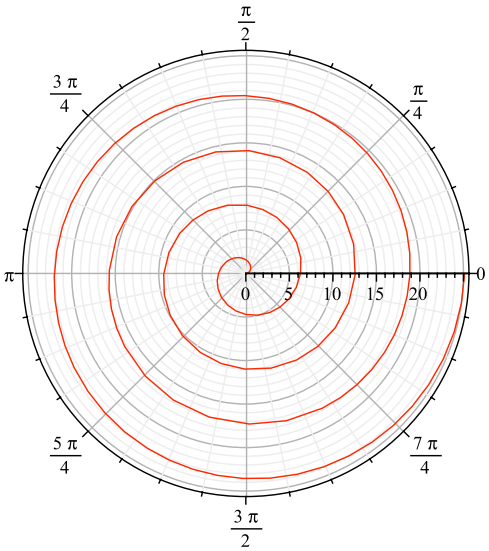
Such a circle, however, will always be centred at the origin. Instead, we'll try and plot our circle centred at the Cartesian co-ordinates $(1, 2)$ with radus 3 (which we have already plotted in two different ways). Unfortunately we cannot think of our polar co-ordinates as vectors quite so easily (although we could happily think of them as complex numbers, if we wished), so establishing the equation of this circle will be somewhat in-depth. Instead we'll look at the spiral again.

The spiral construction with parametric equations $(x, y) = (t\cos t, t\sin t)$ was constructed in a way that is very amenable to a polar equation. Recall that as our t parameter varied as both an angle and a distance. This sounds very much like a polar equation. It should come as no surprise then that the polar equation of the spiral is simply $r = \theta$. To plot this, we simply pass the parameter *coords=polar* to the **plot** function, or alternatively we may use the **polarplot** function from the plots package.

> $plot(\text{theta}, \text{theta} = 0..8 \cdot \text{Pi}, coords = \text{polar})$



> $plots[polarplot](\text{theta}, \text{theta} = 0..8 \cdot \text{Pi})$

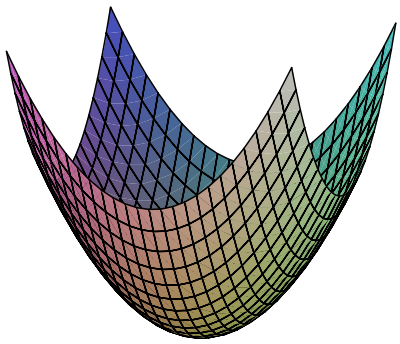


2.3 Multivariable Calculus

2.3.1 3-dimensional Plotting

The **plot** function, which is for 2-dimensional plots, has a counterpart named **plot3d** which is rather unsurprisingly for 3-dimensional plots. Ordinarily, **plot3d** will plot a function $z = f(x, y)$, for example to plot the paraboloid $z = x^2 + y^2$ we simply input

```
> plot3d(x^2 + y^2, x = -1..1, y = -1..1)
```



Be arware that unlike **plot**, the **3dplot** has no default values for the input range and so we must explicitly state ranges for both independent variables.

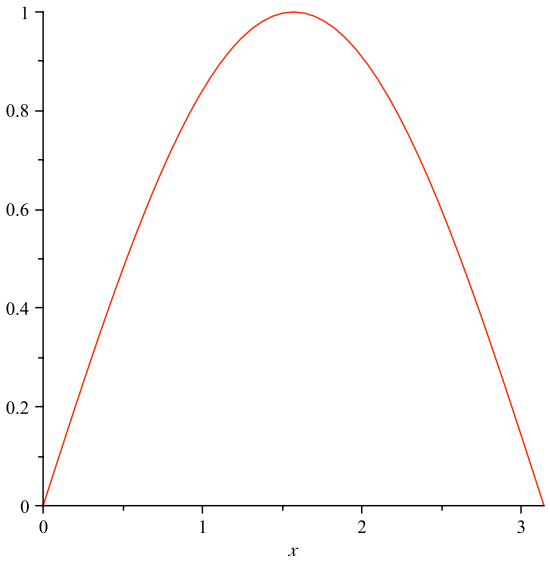
2.3.2 Surfaces and Solids of Rotation

In general, calculating volumes is usually a job for iterated integrals (see Section 2.3.4). However, volumes of solids of revolution may be calculated with a single integral.

Rotation about an arbitrary line in Linear Algebra chapter?

A solid of revolution is taken by rotating a graph in the plane about a line. Usually, but not always, one of the axes is chosen.. As an example, let us consider the sine curve between 0 and π.

> $plot(\sin(x), x = 0..\mathrm{Pi})$



If we rotate this curve about the x-axis we can imagine a sort-of symmetrical teardrop shape.

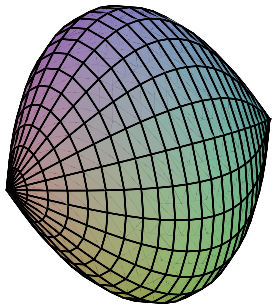
In fact, we can do better than imagine. We can have *Maple* draw us a picture. In order to plot our rotated sine curve, however, we will need to use a parametric equation. To this end, notice that at any point $x \in [0, \pi]$, our rotated surface will have a cross-section, parallel to the y-z plane which is a circle or radius $\sin(x)$. We may, therefore, parameterise the surface of the rotated surface by

$$[x, y, z] = [t, \sin(t)\sin(\theta), \sin(t)\cos(\theta)]$$

where $t \in [0, \pi]$ and $\theta \in [0, 2\pi)$.

This we may now plot, by asking **plot3d** to plot a list *of exactly three elements*, instead of an expression. The 3-element list is interpreted to be the parametric values for points $[x, y, z]$

> $plot3d([t, \sin(t) \cdot \sin(theta), \sin(t) \cdot \cos(theta)], t = 0..\mathrm{Pi}, theta = 0..2 \cdot \mathrm{Pi})$



The parameterisation of this surface gives us a hint as to how to use integration to calculate the volume. We think of the volume as an infinite number of infinitely small discs (otherwise known as "circles"), and add up the area of each circle over an interval—the interval being $[0..\pi]$ in this case. The accumulated area is the volume. This is identical to a Riemann sum where we add up the height of an infinite number of infinitely small boxes (otherwise known as "lines") and accumulate these heights over an interval to obtain an area. The discs we are calculating are perpendicular to the axis of rotation.

We know that the area of any particular disc is πr^2, and since our radius is $\sin(x)$, each disc in our particular example will have area $A(x) = \pi \sin(x)^2$. So the area, remembering we have only rotated the portion of the sine curve between 0 and π, will be

$$\int_0^\pi A(x)\,dx = \int_0^\pi \pi \sin(x)^2\,dx = \pi \int_0^\pi \sin(x)^2\,dx$$

This we may now calculate in *Maple*, or manually as we see fit.

> $\mathrm{Pi} \cdot int(\sin(x)^2, x = 0..Pi)$

$$\frac{1}{2}\,\pi^2$$

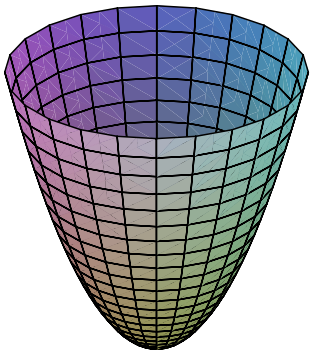
In general, the volume of a solid produced by rotating a function $f(x)$ inside

of an interval $[a, b]$ around the x-axis is given by the integral

$$\pi \int_a^b f(x)^2 \, dx$$

If we wish to rotate a function $f(x)$ around the y-axis, then we will need to rewrite the function as a function of y instead. Note that this is more complicated than just replacing every x in the function with a y. For example, let us rotate the parabola $y = x^2$ around the y-axis for $x \in [0, 2]$. We'll start by plotting what we want to see.

```
> plot3d([x·sin(theta), x·cos(theta), x^2], x = 0..2, theta = 0..2·Pi)
```



This parameterisation was come to first by realising that *Maple*'s **plot3d** always plots with the z-axis pointing up and the y-axis pointing out of the screen (until we rotate it using the mouse). So we renamed the y-axis to be the z-axis instead, giving us our "new" function of $z = x^2$. Given an arbitray x-value, as we rotate this point around the z-axis then we create a circle $[x \sin(\theta), x \cos(\theta)]$ for $\theta \in [0, 2\pi)$ on the $x - y$ plane—which is the floor of our 3d plot now, remember. The z value for any point on this circle will still be x^2. And so we have our parameterisation.

Now in order to use our integral, above, we need to be rotating around the same axis as the independent variable of our function. Here however, we are rotating a function of x around the z-axis (still using the z axis as the upward pointing one). We need to rewrite our function as $x = f(z)$. Rearranging $z = x^2$ in this manner produces $x = \sqrt{z}$. We also notice that

$x = 0 \implies z = 0$ and $x = 2 \implies z = 4$, so we may equivalently consider our rotation as rotating the function $x = \sqrt{z}$ for $z \in [0, 4]$ around the z-axis.

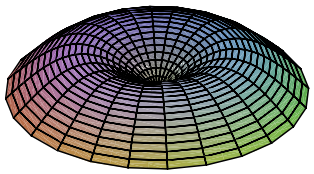
We may now perform the integration to calculate the volume.

```
> Pi · int(sqrt(z)^2, z = 0..4)
```

$$8\,\pi$$

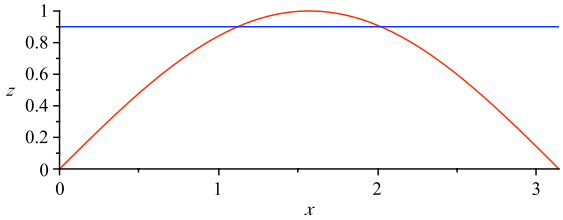
Let us return now to our sine function, plotted between $x = 0$ and $x = \pi$. Let us now rotate this around the z-axis (treating z as the axis pointing "up" again, as we did with the last example). We'll start like we have each other time, with plotting the surface to see what we're dealing with. We add a *scaling=constrained* parameter to make the plot more clear in this case.

```
> plot3d([x · sin(theta), x · cos(theta), sin(x)], x = 0.. Pi, theta = 0..2 ·
  Pi, scaling = constrained)
```



Now we have a problem here, since there's no particularly easy or, at least, obvious way to rewrite $z = \sin(x)$ as a function of z. While the z-interval is clearly $[0, 1]$, for any given value of z there are *two* values of x. This is easily seen with a plot.

```
> plot([sin(x), 0.9], x = 0..Pi, z, colour = [red, blue], scaling = constrained)
```



The line $z = 0.9$ clearly cuts the sine function in two places.

Now, we may try and express the solid of rotation as an area between two functions, $f(z), g(z)$ and calculate the integral accordingly. However finding these two functions will be tedious and time consuming. Instead, we will use a slightly different integral to calculate the area. Our previous method used discs that were perpendicular to the axis of rotation. This time we will use a method known as "shells". To do this, we will approximate the area as infinitely many cylinders, centred at the origin, and with radius x and height $\sin(x)$. The surface area of each cylinder will be the circumference multiplied the height. As such the area function will be $A(x) = 2\pi x \sin(x)$, and our volume may now be calculated as

$$\int_0^{\pi} A(x)\,dx = \int_0^{\pi} 2\pi x \sin(x)\,dx = 2\pi \int_0^{\pi} x \sin(x)\,dx$$

```
> 2 · Pi · int(x · sin(x), x = 0..Pi)
```

$$2\,\pi^2$$

And, in general, if we rotate a function $y = f(x)$ between $x = a$ and $x = b$ around the y-axis then the volume of the solid of revolution is given by the integral

$$2\pi \int_a^b x f(x)\,dx$$

And we always have the possibility of changing between these two integrals, by re-writing the function and interchanging the dependent and independent variables if the integration in one method proves too troublesome.

We may check our paraboloid volume calculation using this method.

```
> 2 · Pi · int(x · x^2, x = 0..2)
```

$$8\,\pi$$

2.3.3 Partial and Directional Derivatives

Recall that for a function of two or more variables, then derivatives are taken with respect to one of the variables at a time. These are known as *partial derivatives*. There are several ways to denote partial derivatives, but we shall use two. Let $f : \mathbb{R}^2 \to \mathbb{R}$. Then the first partial derivative of f with respect to x is denoted

$$f_x \quad \text{or} \quad \frac{\partial f}{\partial x}$$

and the first partial derivative of f with respect to y is denoted

$$f_y \quad \text{or} \quad \frac{\partial f}{\partial x}$$

Both of these are functions ($\mathbb{R}^2 \to \mathbb{R}$) in their own right.

Note that, if we do not have a name for our function, we may write the function in brackets after the ∂ notation. For example

$$\frac{\partial}{\partial x}\left(\frac{x^2+y^2}{x^2-y^2}\right)$$

would be the first partial derivative with respect to x of the rational polynomial $(x^2+y^2)/(x^2-y^2)$.

If we take the derivative of one of these derivatives, then we again must take a partial derivative. By doing so we obtain a second partial derivative. The variable the second derivative is taken with respect to may be different to that the first derivative was taken with respect to (since a first partial derivative is still a valid function in its own right). The four possible second partial derivatives are as follows.

$$\begin{aligned} f_{x,x} &\quad \text{or} \quad \frac{\partial^2 f}{\partial x^2} \\ f_{x,y} &\quad \text{or} \quad \frac{\partial^2 f}{\partial y \partial x} \\ f_{y,x} &\quad \text{or} \quad \frac{\partial^2 f}{\partial x \partial y} \\ f_{y,y} &\quad \text{or} \quad \frac{\partial^2 f}{\partial y^2} \end{aligned}$$

In general if we have a function which is a n^{th} partial derivative, which we then take yet another partial derivative of, then we have the following scenario

$$(f_{x_1,\ldots,x_n})_{x_{n+1}} = f_{x_1,\ldots,x_{n+1}} \quad \text{or} \quad \frac{\partial}{\partial x_{n+1}}\left(\frac{\partial^n}{\partial x_n \cdots \partial x_1}\right) = \frac{\partial^{n+1}}{\partial x_{n+1} \cdots \partial x_1}$$

Note that the above also demonstrates why the ∂ notation has the variables written backwards.

Maple may perform partial derivatives. The **diff** function will happily perform multivariable derivatives. For a first partial derivative, we use tell **diff** which variable we want to take the derivative with respect to

```
> p := sum(sum(x^i * y^j, j = 0..2), i = 0..2);
  diff(p, x); diff(p, y)
```

$$p := 1 + y + y^2 + x + xy + xy^2 + x^2 + x^2y + x^2y^2$$

$$1 + y + y^2 + 2\,x + 2\,xy + 2\,xy^2$$

$$1 + 2\,y + x + 2\,xy + x^2 + 2\,x^2y$$

And for 2^{nd} and later partial derivatives, we simply list the derivatives in the order they are to be taken. Note, that this is exactly what we did for single variable derivatives, only that there was only one variable in that case. So $\frac{d^2f}{dx^2}$ was calculated with $diff(f(x), x, x)$ which told maple to take the first derivative with respect to x and then the second derivative with respect to x. Multivariable derivatives are different only in that we may take the derivative with respect to a different variable at each step.

```
> diff(p, x, x); diff p, x, y);
  diff(p, y, x); diff(p, y, y)
```

$$2 + 2\,y + 2\,y^2$$

$$1 + 2\,y + 2\,x + 4\,xy$$

$$1 + 2\,y + 2\,x + 4\,xy$$

$$2 + 2\,x + 2\,x^2$$

Of course, the inert form of the function still works, as do—as suggested above—higher order derivatives.

```
> Diff(p, x, x, y) = diff(p, x, x, y)
```

$$\frac{\partial^3}{\partial y \partial x^2}\left(1 + y + y^2 + x + xy + xy^2 + x^2 + x^2y + x^2y^2\right) = 2 + 4\,y$$

The **D** function may also be used. For a function of multiple variables, to compute the first partial derivative with respect to the first variable, we use the subscript 1 with the D. To compute the first partial derivative with respect to the second variable, we use a subscript of 2, etc. For example

```
> f := (x, y) → x^2 - y^2 + x·y; D[1](f), D[2](f)
```

$$f := (x, y) \to x^2 - y^2 + xy$$
$$(x, y) \to 2\,x + y$$
$$(x, y) \to -2\,y + x$$

To compute second partial derivatives, we simply subscript **D** with a sequence of numbers describing the variable numbers, in the order that the derivatives are taken. Note that while this is different to the method we used to take derivatives of single variable functions with the **D** command, this method will also work for single variable functions. The following examples may be quickly checked by hand.

```
> Diff(f(x, y), x$2) = D[1,1](f)(x, y)
  Diff(f(x, y), y, x) = D[2,1](f)(x, y);
```

$$\frac{\partial^2}{\partial x^2}\left(x^2 - y^2 + xy\right) = 2$$
$$\frac{\partial^2}{\partial x \partial y}\left(x^2 - y^2 + xy\right) = 1$$

Recall that for a function with continuous second partial derivatives, then the second partial derivatives $f_{y,x} = f_{x,y}$. This is Clairaut's Theorem (see [11]). More precisely, if the function f is defined on a disc, D say, that is a subset of the x-y plane, and the partial second derivatives are continuous on that same disc, then $f_{y,x}(a, b) = f_{x,y}(a, b)$ for every $(a, b) \in D$. It follows, of course, if the function f is defined for all of $\mathbb{R}^2$ and its partial derivatives are continuous on all of $\mathbb{R}^2$ then it will certainly be the case that $f_{y,x} = f_{x,y}$.

We have seen an example of this already, above, with the second partial derivatives of our polynomial p. A quick check of our first partial derivatives of our function f, shows that the second partial derivatives $f_{x,y}$ and $f_{y,x}$ will clearly be equal. Let's look at another example

```
> f := sin(x^2 + y^2) ; diff(f, x, y) = diff(f, y, x);
```

$$f := \sin\left(x^2 + y^2\right)$$
$$-4\,\sin\left(x^2 + y^2\right) yx = -4\,\sin\left(x^2 + y^2\right) yx$$

And this even extends to higher parital derivatives

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

$$-8\,\cos\left(x^2 + y^2\right) yx^2 - 4\,\sin\left(x^2 + y^2\right) y$$
$$-8\,\cos\left(x^2 + y^2\right) yx^2 - 4\,\sin\left(x^2 + y^2\right) y$$
$$-8\,\cos\left(x^2 + y^2\right) yx^2 - 4\,\sin\left(x^2 + y^2\right) y$$

We may also see a failure of Clairaut's Theorem.

The directional derivatives allow us to calculate the *tangent plane* to a surface in 3d. Given a function, $f(x, y)$ say, with continuous derivatives the equation of the tangent plane to the surface of f at a point (x_0, y_0, z_0) is

$$z - z_0 = \frac{\partial f}{\partial x}(x_0, y_0) \cdot (x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0) \cdot (y - y_0)$$

which may be re-written as

$$\begin{aligned} z &= \frac{\partial f}{\partial x}(x_0, y_0) \cdot (x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0) \cdot (y - y_0) + z_0 \\ &= \frac{\partial f}{\partial x}(x_0, y_0) \cdot (x - x_0) + \frac{\partial f}{\partial y}(x_0, y_0) \cdot (y - y_0) + f(x_0, y_0) \end{aligned}$$

since if (x_0, y_0, z_0) is a point on the surface of $f(x, y)$, then it must be the case that $z_0 = f(x_0, y_0)$.

We may (and, indeed, will) explore this in *Maple*. First we will create a procedure which returns an arrow-notation function for the tangent plane equation.

```
> tp := proc(f :: procedure)
    (x0, y0) -> f(x0, y0) + D[1](f)(x0, y0) * (x - x0) + D[2](f)(x0, y0) *
  (y - y0)
  end :
```

Now we have a function *tp* which, when given a procedure as input, will return a new procedure which will calculate the tangent plane at a given point. Because the functions returns a function, we expect to input $tp(f)(a, b)$ to calculate the tangent plane to the function f at the point (a, b).

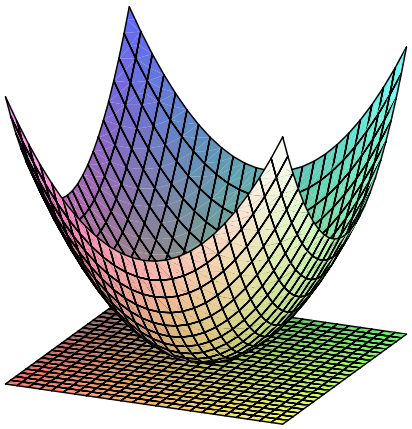
```
> f := (x, y) -> x^2 + y^2; tp(f)(a, b)
```

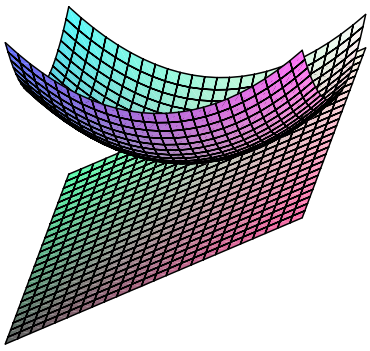
$$f := (x, y) \to x^2 + y^2$$

$$a^2 + b^2 + 2\,a\,(x - a) + 2\,b\,(y - b)$$

Well, that's all well and good, but it's about time we produced some plots.

```
> plot3d([f(x,y),tp(f)(0,0)],x=-2..2,y=-2..2);
  plot3d([f(x,y),tp(f)(1,1)],x=-2..2,y=-2..2);
```





2.3.4 Double Integrals

2.4 Exercises

The exercises for this, and subsequent, will be less numerous, and slightly longer in duration compared to the exercises from Chapter 1. An effort has been made to keep the amount of effort each question requires to be roughly the same for each question, and also for each question to be more or less self contained. However, no guarantees to this effect are made.

1. Plot the following functions. Make sure that, where possible, you plot all important information to the plot—turning points, zeroes, etc.

   (a) $x^5 - 7x^4 - 162x^3 + 878x^2 + 3937x - 15015$

   (b) $\dfrac{\sin x}{x}$

   (c) $\dfrac{\cos x - 1}{x}$

   (d) $x^5 - 3x^4 + x^2 - x - 5$

   The following plots produce slightly unexpected results. Plot the functions, and identify the unexpected behaviour. Modify the **plot** parameters to produce a more correct plot. Also, have *Maple* plot the functions with identical scale for the vertical and horizontal axes.

   (e) $2 + \sin x$ (f) $\sin(x)^2 + \cos(x)^2$

2. Using the substitution $u = \pi - x$ and *Maple*, check the identity

$$\int_0^\pi x f(\sin x)\,dx = \frac{\pi}{2}\int_0^\pi f(\sin x)\,dx$$

   The **Change** function, found in the IntegrationTools package will perform substution on integrals.
   Note: the **Change** function seems to work best on inert integrals.

3. Plot pac-man on a set of Cartesian axes. **Hint**: use multiple polar co-ordinate plots and the **display** function.

2.5 Further Explorations

Chapter 3

Linear Algebra

3.1

3.1.1 1st Year Review

3.1.2 Simultaneous Linear Equations

3.1.3 Coupled Differential Equations

3.1.4 Difference Equations

3.2

3.2.1 Bases

3.2.2 Linear Transformations

3.2.3 Matrices as Linear Transformations

3.2.4 Change of Basis

- Rotating points around arbitrary vectors
- Rotated hyperbola (use alternate basis to obtain cartesian coordinates)
- Nonstandard asymptotes

3.3

3.4 Exercises

3.5 Further Explorations

Chapter 4

Visualisation and Geometry

4.1 Geometry in Maple

4.2 Interractive Geometry

4.2.1 The Argand Diagram

4.3 Lions-Mercier iterations

4.4 Exercises

4.5 Further Explorations

Appendix A

Maple Input Reference

A.1 *Maple* math-mode Input Conventions

A.2 Common *Maple* functions

Bibliography

[1] David Bailey, Jonathan Borwein, Neil Calkin, Roland Girgensohn, Russell Luke, and Victor Moll. *Experimental Mathematics in Action*. AK Peters, 1st edition, 2007.

[2] Jonathan Borwein and David Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. AK Peters, 2nd edition, 2008.

[3] Jonathan Borwein, David Bailey, and Roland Girgensohn. *Experimentation in Mathematics: Computational Paths to Discovery*. AK Peters, 1st edition, 2004.

[4] Jonathan Borwein and Keith Devlin. *The Computer as Crucible: An Introduction to Experimental Mathematics*. AK Peters, 2009.

[5] Walter Gander and Jir Hrebcek. *Solving Problems in Scientific Computing Using Maple and MATLAB*. Springer, 4th edition, 2008.

[6] Frank Garvan. *The Maple 5 Primer Rel 4*. Prentice Hall, 1997.

[7] Frank Garvan. *The Maple Book*. Chapman and Hall/CRC, 2001.

[8] Andre Heck. *Introduction to Maple*. Springer, 3rd edition, 2003.

[9] Grazyna Klimek and Maciej Klimek. *Discovering Curves and Surfaces with Maple*. Springer, 1997.

[10] Vladimir Y. Rovenski. *Geometry of Curves and Surfaces with MAPLE*. Springer, 2000.

[11] James Stewart. *Calculus*. Brooks/Cole, 4th edition, 1999.

[12] Michael Trott. *The Mathematica Guidebooks*. Wolfram, 3rd edition, 2005.

[13] Stan Wagon. *Mathematica in Action*. Springer, 2nd edition, 1999.