# 1 An introduction to *Mathematica*

*Mathematica* is a very large and seemingly complex system. It contains hundreds of functions for performing various tasks in science, mathematics, and engineering, including computing, programming, data analysis, knowledge representation, and visualization of information. In this introductory chapter, we introduce the elementary operations in *Mathematica* and give a sense of its computational and programming breadth and depth. In addition, we give some basic information that users of *Mathematica* need to know, such as how to start *Mathematica*, how to get out of it, how to enter simple inputs and get answers, and finally how to use *Mathematica*'s documentation to get answers to questions about the system.

## 1.1 A brief overview of *Mathematica*

### *Numerical computations*

*Mathematica* has been aptly described as a sophisticated calculator. With it you can enter mathematical expressions and compute their values.

```
In[1]:= Sin[.86] - Log[π] (1 + .08/12)^12
Out[1]= -0.481899
```

You can store values in memory.

```
In[2]:= rent = 350
Out[2]= 350

In[3]:= food = 175
Out[3]= 175

In[4]:= heat = 83
Out[4]= 83
```

```
In[5]:= rent + food + heat
Out[5]= 608
```

Yet *Mathematica* differs from calculators and simple computer programs in its ability to calculate exact results and to compute to an arbitrary degree of precision.

```
In[6]:= 1/15 + 1/35 + 1/63
Out[6]= 1/9

In[7]:= 2^500
Out[7]= 32733906078961418700131896968275991522166420460430647894832913
        68096133796404674554883270092325904157150886684127560071009
        217256545885393053328527589376

In[8]:= N[π, 500]
Out[8]= 3.14159265358979323846264338327950288419716939937510582097494
        45923078164062862089986280348253421170679821480865132823066
        70938446095505822317253594081284811174502841027019385211055
        96446229489549303819644288109756659334461284756482337867831
        52712019091456485669234603486104543266482133936072602491412
        37245870066063155881748815209209628292540917153643678925903
        60011330530548820466521384146951941511609433057270365759591
        30921861173819326117931051185480744623799627495673518857527
        48912279381830119491
```

## *Symbolic computations*

One of the more powerful features of *Mathematica* is its ability to manipulate and compute with symbolic expressions. For example, you can factor polynomials and simplify trigonometric expressions.

```
In[9]:= Factor[x^5 - 1]
Out[9]= (-1 + x) (1 + x + x^2 + x^3 + x^4)

In[10]:= TrigReduce[Sin[θ]^3]
Out[10]= 1/4 (3 Sin[θ] - Sin[3 θ])
```

You can simplify expressions using assumptions about variables contained in those expressions. For example, if $k$ is assumed to be an integer, $\sin(2\pi k + x)$ simplifies to $\sin(x)$.

```
In[11]:= Simplify[Sin[2 π k + x], k ∈ Integers]
Out[11]= Sin[x]
```

This computes the conditions for which a general quadratic polynomial will have both roots equal to each other.

```
In[12]:= Reduce[∃x, a x^2+b x+c==0 (∀y, a y^2+b y+c==0 x == y), {a, b, c}]
```

$$\text{Out[12]=}\quad (a == 0 \;\&\&\; b \neq 0) \;||\; (a == 0 \;\&\&\; b\, c \neq 0) \;||\; \left(a \neq 0 \;\&\&\; c == \frac{b^2}{4\,a}\right)$$

You can create functions that are defined piecewise.

```
In[13]:= Piecewise[ {{1, x == 0}}, Sin[x] / x]
```

$$\text{Out[13]=}\quad \begin{cases} 1 & x == 0 \\ \frac{\text{Sin}[x]}{x} & \text{True} \end{cases}$$

The knowledge base of *Mathematica* includes algorithms for solving polynomial equations, and computing integrals.

```
In[14]:= Solve[x^3 - a x + 1 == 0, x]
```

$$\text{Out[14]=}\quad \left\{\left\{x \to \frac{\left(\frac{2}{3}\right)^{1/3} a}{\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}} + \frac{\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}}{2^{1/3}\,3^{2/3}}\right\},\right.$$

$$\left\{x \to -\frac{\left(1 + i\,\sqrt{3}\right) a}{2^{2/3}\,3^{1/3}\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}} - \frac{\left(1 - i\,\sqrt{3}\right)\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}}{2\,2^{1/3}\,3^{2/3}}\right\},$$

$$\left.\left\{x \to -\frac{\left(1 - i\,\sqrt{3}\right) a}{2^{2/3}\,3^{1/3}\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}} - \frac{\left(1 + i\,\sqrt{3}\right)\left(-9 + \sqrt{3}\,\sqrt{27 - 4\,a^3}\right)^{1/3}}{2\,2^{1/3}\,3^{2/3}}\right\}\right\}$$

```
In[15]:= ∫ 1/(1 + x^4) dx
```

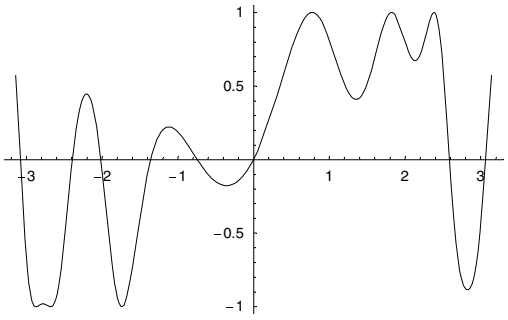
$$\text{Out[15]=}\quad \frac{1}{4\sqrt{2}}\left(-2\,\text{ArcTan}\left[1 - \sqrt{2}\,x\right] + 2\,\text{ArcTan}\left[1 + \sqrt{2}\,x\right] - \text{Log}\left[-1 + \sqrt{2}\,x - x^2\right] + \text{Log}\left[1 + \sqrt{2}\,x + x^2\right]\right)$$

## *Graphics*

The ability to visualize functions or sets of data often allows us greater insight into their structure and properties. *Mathematica* provides a wide range of graphing capabilities. These include two- and three-dimensional plots of functions or data sets, contour and density plots of functions of two variables, bar charts, histograms and pie charts of data sets, and many packages designed for specific graphical purposes. In addition, the *Mathematica* programming language allows you to construct graphical images "from the ground up" using primitive elements, as we will see in Chapter 9.

Here is a simple two-dimensional plot of the function $\sin\left(x + \sqrt{2}\,\sin(x^2)\right)$.

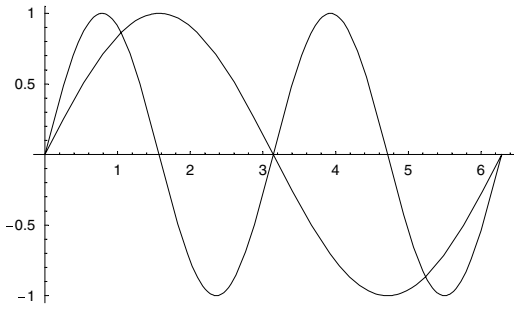
```
In[16]:= Plot[Sin[x + Sqrt[2] Sin[x^2]], {x, -π, π}]
```




```
Out[16]= - Graphics -
```

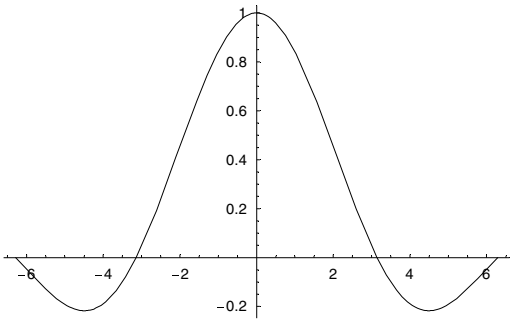
You can combine two or more plots in a single graphic by enclosing them inside curly braces.

```
In[17]:= Plot[{Sin[x], Sin[2 x]}, {x, 0, 2 π}];
```



Here is a plot of the sinc function, given in the previous section.

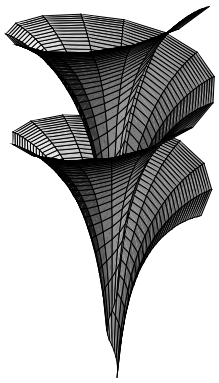
```
In[18]:= Plot[Piecewise[ {{1, x == 0}}, Sin[x] / x], {x, -2 π, 2 π}];
```




Here is a surface of constant negative curvature, represented parametrically by the three functions $\rho$, $\sigma$, and $\tau$. This surface is often referred to as Dini's surface.

```
In[19]:= ρ = Cos[ϕ] Sin[θ];
         σ = Sin[ϕ] Sin[θ];
         τ = 0.2 ϕ + Cos[θ] + Log[Tan[θ / 2]];
```

```
In[22]:= ParametricPlot3D[{ρ, σ, τ}, {ϕ, 0, 4 π}, {θ, .05, 1}, Axes → False,
           Boxed → False, PlotPoints → 30, AspectRatio → 1.75];
```



### *Working with data*

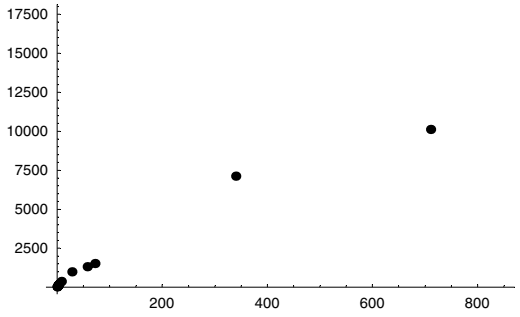
The ability to plot and visualize data is extremely important in engineering and all of the social, natural, and physical sciences. *Mathematica* can import and export data from other applications, plot the data in a variety of forms, and be used to perform numerical analysis on the data.

The file `dataset.m` contains pairs of data points, in this case representing body mass vs. heat production for 13 different animals. The data are given as ($m$, $r$), where $m$ represents the mass of the animal and $r$ the heat production in *kcal* per day. First we set up a platform independent path to the file and then import that file.

```
In[23]:= datafile = ToFileName[{$BaseDirectory,
            "Applications", "IPM3", "DataFiles"}, "dataset.m"]

Out[23]= C:\Documents and Settings\All Users\Application Data\
           Mathematica\Applications\IPM3\DataFiles\dataset.m

In[24]:= data = Import[datafile, "Table"]

Out[24]= {{0.06099, 6.95099}, {0.403, 28.189},
          {0.62199, 41.1}, {2.50999, 120.799},
          {2.95999, 147.9}, {3.33, 182.8}, {8.19999, 368.8},
          {28.1999, 981.299}, {57.4, 1303.29}, {72.2999, 1512.5},
          {340.199, 7100.29}, {711, 10101.1}, {5000., 29894.9}}
```

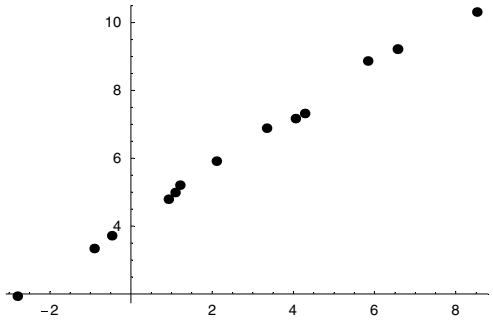
You can immediately plot the data using the `ListPlot` function.

```
In[25]:= ListPlot[data, PlotStyle → PointSize[.02]];
```



This plots the data on log–log axes.

```
In[26]:= logplot = ListPlot[Log[data], PlotStyle → PointSize[.02]];
```

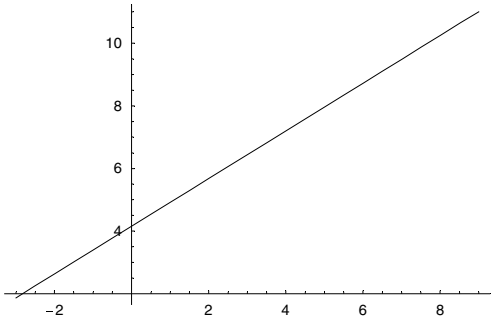



You can then fit a straight line to the log-data by performing a linear least squares fit. In this example, we are fitting to the model $a + m\,x$, where $a$ and $m$ are the parameters to be determined in the model with variable $x$.

```
In[27]:= f = FindFit[Log[data], a + m x, {a, m}, x]
Out[27]= {a → 4.15437, m → 0.761465}
```

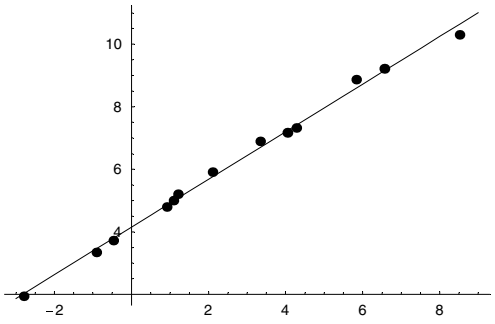
Here is a plot of the linear fit function.

```
In[28]:= fplot = Plot[a + m x /. f, {x, -3, 9}];
```



Finally, you can see how well the fitted function approximates the log plot by combining these last two graphics.

```
In[29]:= Show[fplot, logplot];
```




## *Programming*

With a copy of *The Mathematica Book* (Wolfram 2003) or one of the many tutorial books (see, for example, Glynn and Gray 1999) describing the vast array of computational tasks that can be performed with *Mathematica*, it would seem you can compute just about anything you might want. But that impression is mistaken. There are simply more kinds of calculations than could possibly be included in a single program. Whether you are interested in computing bowling scores or finding the mean square distance of a random walk on a torus, *Mathematica* does not have a built-in function to do everything that a user could possibly want. What it *does* have – and what really makes it the amazingly useful tool it is – is the capability for users to define their own functions. This is called *programming*, and it is what this book is all about.

Sometimes, the programs you create will be succinct and focused on a very specific task. *Mathematica* possesses a rich set of tools that enable you to quickly and naturally translate the statement of a problem into a program. For example, the following program defines a test for perfect numbers, numbers that are equal to the sum of their proper divisors.

```
In[30]:= PerfectQ[n_] := Apply[Plus, Divisors[n]] == 2 n
```

We then define another function that selects those numbers from a range of integers that pass this `PerfectQ` test.

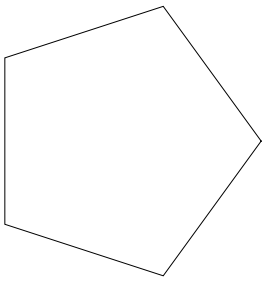
```
In[31]:= PerfectSearch[n_] := Select[Range[n], PerfectQ]
```

This then finds all perfect numbers less than 1,000,000.

```
In[32]:= PerfectSearch[10^6]

Out[32]= {6, 28, 496, 8128}
```

Here are two functions for representing regular polygons. The first defines the vertices of a regular *n*-gon, while the second uses those vertices to create a polygon graphics object that can then be displayed with the built-in `Show` function.

```
In[33]:= vertices[n_Integer, r_: 1] :=
          Table[{r Cos[2 α π / n], r Sin[2 α π / n]}, {α, 0, n - 1}]

In[34]:= RegularPolygon[n_] :=
          Graphics[Line[vertices[n] /. {a_, b__} → {a, b, a}],
           AspectRatio → Automatic]

In[35]:= Show[RegularPolygon[5]]
```



```
Out[35]= - Graphics -

In[36]:= NewtonZero[f_, xi_] := NestWhile[(# - f[#] / f'[#]) &, xi, Unequal, 2]

In[37]:= g[x_] := x^3 - 2 x^2 + 1

In[38]:= NewtonZero[g, 2.0]

Out[38]= 1.61803
```

Of course, sometimes the task at hand requires a more involved program, stretching across several lines (or even pages) of code. For example, here is a slightly longer program to compute the score of a game of bowling, given a list of the number of pins scored by each ball.

```
In[39]:= BowlingScore[pins_] :=
          Module[{score}, score[{x_, y_, z_}] := x + y + z;
           score[{10, y_, z_, r___}] := 10 + y + z + score[{y, z, r}];
           score[{x_, y_, z_, r___}] :=
            x + y + z + score[{z, r}] /; x + y == 10;
           score[{x_, y_, r___}] := x + y + score[{r}] /; x + y < 10;
           score[If[pins[[-2]] + pins[[-1]] ≥ 10, pins, Append[pins, 0]]]]
```

Here is the computation for a "perfect" game – 12 strikes in a row.

```
In[40]:= BowlingScore[{10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10}]
Out[40]= 300
```

These examples use a variety of programming styles: functional programming, rule-based programming, the use of anonymous functions, and more. We do not expect you to understand the examples in this section at this point – that is why we wrote this book! What you should understand is that in many ways *Mathematica* is designed to be as broadly useful as possible and that there are many calculations for which *Mathematica* does not have a built-in function, so, to make full use of its many capabilities, you will sometimes need to program. The main purpose of this book is to show you how.

Another purpose is to teach you the basic principles of programming. These principles – making assignments, defining rules, using conditionals, recursion, and iteration – are applicable (with great differences in detail, to be sure) to all other programming languages.

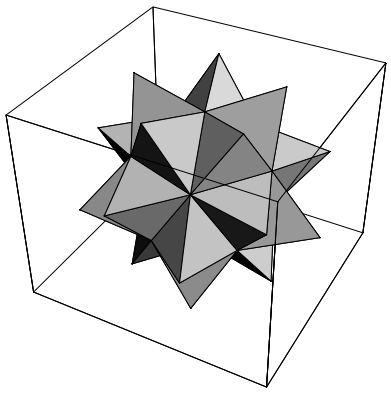
### *Symbolic and interactive documents*

In addition to the computational tools that *Mathematica* provides for what many professionals associate with technical computing, it also contains tools for creating and modifying the user interface to such tasks. These tools include hyperlinks for jumping to other locations within a document or across files, buttons to perform tasks that you might normally associate with a command-line interface, and tools to modify and manipulate the appearance and functionality of your *Mathematica* notebooks directly. In this section we will give a few short examples of what is possible, waiting until Chapter 10 for a methodical look at how to program these elements.

The first example takes the code necessary to display a polyhedron and puts it in a button. The two lines of code that could be evaluated normally in a notebook first load a package and then display an icosahedron in the notebook.

```
In[41]:= Needs["Graphics`Polyhedra`"]
```

```
In[42]:= Show[Stellate[Polyhedron[Icosahedron]]]
```



```
Out[42]= - Graphics3D -
```

Here is a short program that creates a button containing the above two expressions.

```
Cell[BoxData[
    ButtonBox[
      RowBox[{"Stellate", " ", "Icosahedron"}],
      ButtonFunction:>CompoundExpression[
          Needs[ "Graphics`Polyhedra`"],
          Show[Stellate[Polyhedron[Icosahedron]]]
          ],
      ButtonEvaluator->Automatic],
  "Input",
  Active->True]
```

The formatted version of the above cell can be displayed by choosing Show Expression from the Format menu. When you do that, it will look like the following:

**Stellate Icosahedron**

Clicking the button will cause the *Mathematica* code in the `ButtonFunction` to be immediately evaluated and the following graphics will then be displayed in your notebook.

Functions are available to jump around to different parts of a *Mathematica* notebook and perform various actions. Here is a short piece of code that creates a button which, upon being clicked, moves the selection to the next cell and then evaluates that cell.

```
Cell[TextData[{
  Cell[BoxData[
      ButtonBox["EVALUATE",
        ButtonFunction:>FrontEndExecute[ {
            FrontEnd`SelectionMove[
              ButtonNotebook[ ], All, ButtonCell],
            FrontEnd`SelectionMove[
              ButtonNotebook[ ], Next, Cell],
            FrontEnd`SelectionEvaluate[
              ButtonNotebook[ ]]}],
        Active->True]]],
  StyleBox[" MATHEMATICA INPUT"]
}], "Text"]
```

The formatted version of the above cell can be displayed by choosing Show Expression from the Format menu. When you do that, it will look somewhat like the following (although we have removed some of the text formatting above to improve readability of the code). Clicking the EVALUATE button will cause the input cell immediately following to be selected and then evaluated.

**EVALUATE** ***MATHEMATICA INPUT***

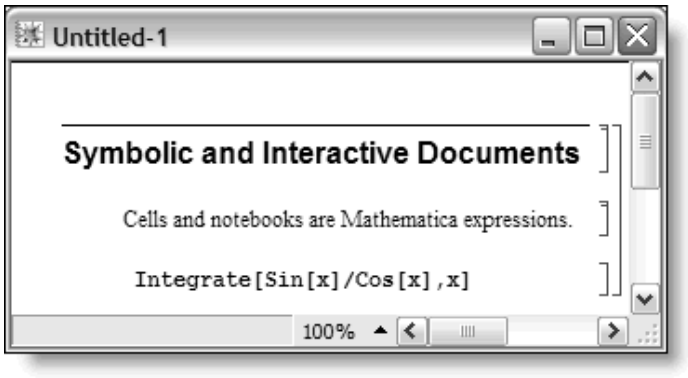
*In[43]:=* **3 (4 + 5)**

*Out[43]=* 27

The following example demonstrates how you can use *Mathematica* functions to perform some of the user interface actions that you would normally associate with keyboard and mouse events. By using such techniques, you can create a specific set of actions that will follow certain evaluations. For example, if you were creating an electronic quiz for your students, you could include "hint" buttons within your class notebooks that would open a new notebook with hints and suggestions upon clicking.

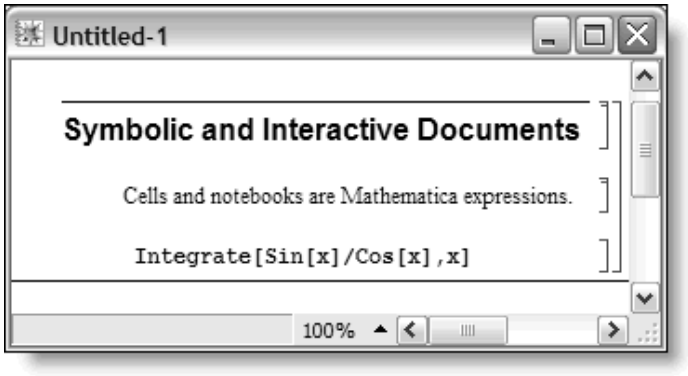
This creates a new notebook that contains three cells – a `Section` cell, a `Text` cell, and an `Input` cell. Upon evaluation, the `NotebookPut` command below will cause a new notebook to appear, containing the three specified cells. The screen shots below show what appears in the user interface after evaluating each of the preceding inputs.

```
In[44]:= nb = NotebookPut[
            Notebook[{
              Cell["Symbolic and Interactive Documents", "Section"],
              Cell["Cells and notebooks are Mathematica expressions.",
               "Text"],
              Cell["Integrate[Sin[x]/Cos[x],x]", "Input"]
             }]]
Out[44]= NotebookObject[ <<Untitled-1>> ]
```



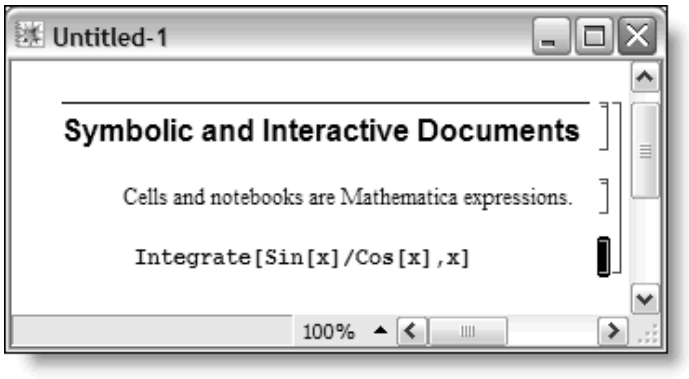

This moves the selection bar past the last cell in the above notebook.

```
In[45]:= SelectionMove[nb, Next, Cell, 4]
```



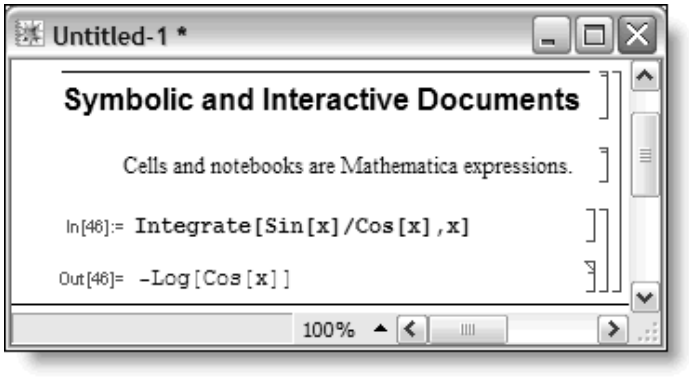
We then select the most previous cell.

```
In[46]:= SelectionMove[nb, Previous, Cell]
```




Finally, we evaluate the selected cell.

```
In[47]:= SelectionEvaluate[nb]
```




In Chapter 10 we will give a detailed discussion of how to modify and manipulate the user interface through the use of the symbolic programming techniques that are discussed throughout this book.

## 1.2 Using *Mathematica*

Before you can do any serious work, you will need to know how to get a *Mathematica* session started, how to stop it, and how to get out of trouble when you get into it. These procedures depend somewhat on the system you are using. You should read the system-specific information that came with your copy of *Mathematica*; and you may need to consult a local *Mathematica* guru if our advice here is not applicable to your system.

### *Getting into and out of Mathematica*

The most commonly used interface is often referred to as a notebook interface in which the user creates and works in interactive documents. Personal computers running Windows, Macintosh operating systems, Linux, and most flavors of Unix all support this graphical user interface, which normally starts up automatically when you begin your *Mathematica* session.

There are some situations where you may want to start up *Mathematica* from a command prompt and issue commands directly through that interface, bypassing the notebook interface entirely. For example, you may have a very long computation that you need to run in batch mode. Typically, *Mathematica* is started up on these systems by typing `math` at a command prompt. We will not discuss using *Mathematica* through a command prompt any further. If you are interested in this mode you should consult the documentation that came with your copy of *Mathematica*.

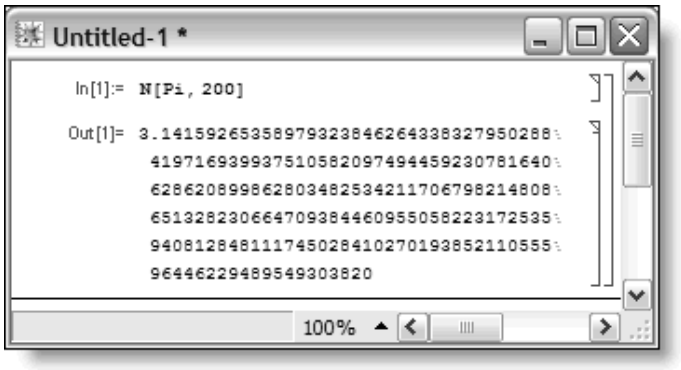
#### Starting *Mathematica* and first computations

To start *Mathematica* you will have to find and then double-click on the *Mathematica* icon on your computer, which will look something like this:

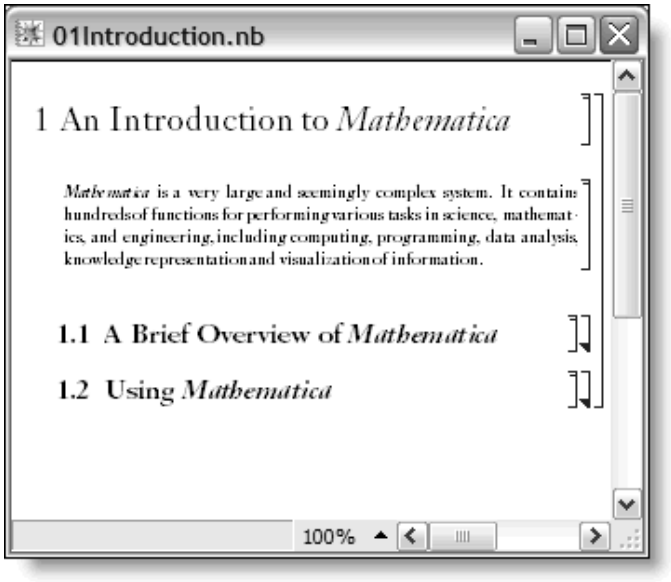
The computer will then load parts of *Mathematica* into its memory and soon a blank window will appear on the screen. This window is the visual interface to a *Mathematica* notebook and it has many features that are useful to the user.

Notebooks allow you to write text, perform computations, write and run programs, and create graphics all in one document. Notebooks also have many of the features of common word processors, so those familiar with word processing will find the notebook interface easy to learn. In addition, the notebook provides features for outlining material which you may find useful for giving talks and demonstrations.

When a blank notebook first appears on the screen (either from just starting *Mathematica* or from selecting New in the File menu), you can start typing immediately. For example, if you type `N[Pi,200]` press SHIFT+ENTER (hold down the Shift key while pressing the Enter key) to evaluate an expression. *Mathematica* will evaluate the result and print the 200-decimal digit approximation to $\pi$ on the screen.

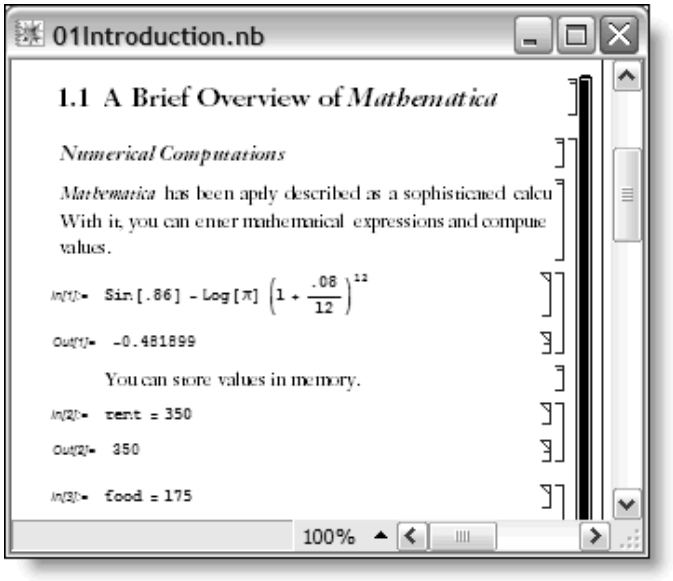



Notice that when you evaluate an expression in a notebook, *Mathematica* adds input and output prompts. In the example notebook above, these are denoted `In[1]:=` and `Out[1]=`. These prompts can be thought of as markers (or labels) that you can refer to during your *Mathematica* session.



You should also note that when you started typing *Mathematica* placed a *bracket* on the far right side of the window that enclosed the *cell* that you were working in. These *cell brackets* are helpful for organizational purposes within the notebook. Double-clicking on cell brackets will open any collapsed cells, or close any open cells as can be seen in the previous screen shot.

Double-clicking on the cell bracket containing the **1.1 A Brief Overview of *Mathematica*** cell will open the cell to display its contents:




Using cell brackets in this manner allows you to organize your work in an orderly manner, as well as to outline material. For a complete description of cell brackets and many other interface features, you should consult the documentation that came with your version of *Mathematica*.

For information on other features such as saving, printing, and editing notebooks, consult the manuals that came with your version of *Mathematica*.

**Entering input**

New input can be entered whenever there is a horizontal line that runs across the width of the notebook. If one is not present where you wish to place an input cell, move the cursor up and down until it changes to a horizontal bar and then click the mouse once. A horizontal line should now appear across the width of the window. You can immediately start typing and an input cell will be created.

Input can be entered exactly as it appears in this book. To get *Mathematica* to evaluate an expression that you have entered, press SHIFT-ENTER; that is, hold down the Shift key and then press the Enter key.

You can enter mathematical expressions in a traditional looking two-dimensional format using either palettes for quick entry of template expressions, or keyboard equivalents. For example, the following expression can be entered by using the Basic Input palette, or through a series of keystrokes. For details of inputting mathematical expressions, read your user documentation or read the section on 2D Expression Input in the Help Browser.

```
In[1]:= Integrate[1/(1 - x^3), x]
```

$$\text{Out[1]}= \frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} - \frac{1}{3}\text{Log}[-1+x] + \frac{1}{6}\text{Log}[1+x+x^2]$$

As noted previously, *Mathematica* enters the `In` and `Out` prompts for you. You do not type these prompts. You will see them *after* you evaluate your input.

You can refer to the result of the previous calculation using the symbol `%`.

```
In[2]:= 2^64

Out[2]= 18446744073709551616

In[3]:= % + 1

Out[3]= 18446744073709551617
```

You can also refer to the result of any earlier calculation using its `Out[`*i*`]` label or, equivalently, `%`*i*.

```
In[4]:= Out[1]
```

$$\text{Out[4]}= \frac{\text{ArcTan}\left[\frac{1+2x}{\sqrt{3}}\right]}{\sqrt{3}} - \frac{1}{3}\text{Log}[-1+x] + \frac{1}{6}\text{Log}[1+x+x^2]$$

```
In[5]:= %2

Out[5]= 18446744073709551616
```

### Ending a *Mathematica* session

To end your *Mathematica* session, choose Exit from the File menu. You will be prompted to save any unsaved open notebooks.

**Getting out of trouble**

From time to time, you will type an input which will cause *Mathematica* to misbehave in some way, perhaps by just going silent for a long time (if, for example, you have inadvertently asked it to do something very difficult) or perhaps by printing out screen after screen of not terribly useful information. In this case, you can try to "interrupt" the calculation. How you do this depends on your computer's operating system:

- Macintosh: type CMD-. (the Command key and the period) and then type `a`
- Windows 95/98/NT/2000/XP: type ALT-. (the Alt key and the period)
- Unix: type CTRL-. and then type `a` and then RET

These attempts to stop the computation will sometimes fail. If after waiting a reasonable amount of time (say, a few minutes), *Mathematica* still seems to be stuck, you will have to "kill the kernel." (Before attempting to kill the kernel, try to convince yourself that the computation is really in a loop from which it will not return and that it is not just an intensive computation that requires a lot of time.) Killing the kernel is accomplished by selecting Quit Kernel from the Kernel menu. The kernel can then be restarted without killing the front end by first selecting Start Kernel ▸ Local under the Kernel menu, or you can simply evaluate a command in a notebook and a new kernel should start up automatically.

## *The syntax of inputs*

You can enter mathematical expressions in a linear syntax using arithmetic operators common to almost all computer languages.

```
In[6]:= 39 / 13

Out[6]= 3
```

Alternately, you can enter this expression in the traditional form by typing `39`, CTRL-/, then `13`.

$$\text{In[7]:=}\quad \frac{39}{13}$$

```
Out[7]= 3
```

The caret (^) is used for exponentiation.

```
In[8]:= 2^5

Out[8]= 32
```

You can enter this expression in a more traditional typeset form by typing `2`, CTRL[^], and then `5`.

```
In[9]:= 2^5

Out[9]= 32
```

*Mathematica* includes several different ways of entering typeset expressions, either directly from the keyboard as we did above, or via palettes available from the File menu. Below is a brief table showing some of the more commonly used typeset expressions and how they are entered through the keyboard. You should read your documentation and become comfortable using these input interfaces so that you can easily enter the kinds of expressions in this book.

| Expression | FullForm | Keyboard shortcut |
|---|---|---|
| $x^2$ | `SuperscriptBox[x, 2]` | `x` CTRL[6], `2` |
| $x_i$ | `SubscriptBox[x, i]` | `x` CTRL[ - ], `i` |
| $\frac{x}{y}$ | `FractionBox[x, y]` | `x` CTRL[ / ], `y` |
| $\sqrt{x}$ | `SqrtBox[x]` | CTRL[2], `x` |
| $x \geq y$ | `GreaterEqual[x, y]` | `x` ESC >= ESC, `y` |

**Table 1.1:** Entering typeset expressions

You can indicate multiplication by simply putting a space between the two factors, as in mathematics. You can also use the asterisk (`*`) for that purpose, as is traditional in most computer languages.

```
In[10]:= 2 5

Out[10]= 10

In[11]:= 2 * 5

Out[11]= 10
```

*Mathematica* also gives operations the same precedence as in mathematics. In particular, multiplication and division have a higher precedence than addition and subtraction, so that `3 + 4 * 5` equals 23 and not 35.

```
In[12]:= 3 + 4 5

Out[12]= 23
```

Functions are also written as they are in mathematics books, except that function names are capitalized and their arguments are enclosed in square brackets.

```
In[13]:= Factor[x^5 - 1]

Out[13]= (-1 + x) (1 + x + x^2 + x^3 + x^4)
```

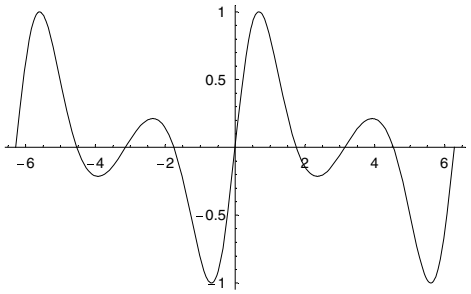
Almost all of the built-in functions are spelled out in full, as in the above example. The exceptions to this rule are well-known abbreviations such as `D` for differentiation, `Sqrt` for square roots, `Log` for logarithms, and `Det` for the determinant of a matrix. Spelling out the name of a function in full is quite useful when you are not sure whether a function exists to perform a particular task. For example, if we wanted to compute the conjugate of a complex number, an educated guess would be:

```
In[14]:= Conjugate[3 + 4 i]

Out[14]= 3 - 4 i
```

Whereas square brackets `[` and `]` are used to enclose the arguments to functions, curly braces { and } are used to indicate a *list* or range of values. Lists are a basic data type in *Mathematica* and are used to represent vectors and matrices (and tensors of any dimension), as well as additional arguments to functions such as in `Plot` and `Integrate`.

```
In[15]:= {a, b, c}.{x, y, z}

Out[15]= a x + b y + c z
```

```
In[16]:= Plot[Sin[x + Sqrt[2] Sin[x]], {x, -2 π, 2 π}];
```




```
In[17]:= Integrate[Cos[x], {x, a, b}]

Out[17]= -Sin[a] + Sin[b]
```

In the `Plot` example, the list `{x,-2π,2π}` indicates that the function $\sin\left(x + \sqrt{2}\,\sin(x)\right)$ is to be plotted over an interval as $x$ takes on values from $-2\,\pi$ to $2\,\pi$. The `Integrate` expression above is equivalent to the mathematical expression $\int_a^b \cos(x)\,dx$.

*Mathematica* has very powerful list-manipulating capabilities that will be explored in detail in Chapter 3.

When you end an expression with a semicolon (;), *Mathematica* computes its value but does not display it. This is very helpful when the result of the expression would be very long and you do not need to see it. In the following example, we first create a list of the integers from 1 to 10,000, suppressing their display with the semicolon, and then compute their sum and average.

```
In[18]:= nums = Range[10000];

In[19]:= Apply[Plus, nums]

Out[19]= 50005000

In[20]:=      %
         ----------------
         Length[nums]

Out[20]= 10001
         -----
           2
```

An expression can be entered on multiple lines, but only if *Mathematica* can tell that it is not finished after the first line. For example, you can enter `3*` on one line and `4` on the next.

```
In[21]:= 3 *

         4

Out[21]= 12
```

But you cannot enter `3` on the first line and `*4` on the second.

```
In[22]:= 3
         *4

Out[22]= 3
```

If you use parentheses, you can avoid this problem.

```
In[23]:= (3
          *4)

Out[23]= 12
```

With the notebook interface, you can input as many lines as you like within an input cell; *Mathematica* will evaluate them all when you enter SHIFT+ENTER still obeying the rules stated above for any incomplete lines.

Finally, you can enter a *comment* – some words that are not evaluated – by entering the words between `(*` and `*)`.

```
In[24]:= D[Sin[x],     (* differentiate Sin[x]   *)
          {x, 1}]   (* with respect to x once *)

Out[24]= Cos[x]
```

### *Alternate input syntax*

There are several different ways to write expressions in *Mathematica*. Usually, you will simply use the traditional notation, *fun*`[`*x*`]`, for example. But you should be aware of several alternatives to this syntax that are widely used.

Here is an example using the standard function notation for writing a function with one argument.

```
In[25]:= N[π]

Out[25]= 3.14159
```

This uses a prefix operator.

```
In[26]:= N@π

Out[26]= 3.14159
```

Here is a postfix operator notation.

```
In[27]:= π // N

Out[27]= 3.14159
```

For functions with two arguments, you can use an infix notation. The following expression is identical to `N[π,30]`.

```
In[28]:= π ~N~ 30

Out[28]= 3.14159265358979323846264338328
```

Finally, many people prefer to use a more traditional syntax when entering and working with mathematical expressions. You can compute an integral using standard *Mathematica* syntax.

```
In[29]:= Integrate[1 / Sin[x], x]

Out[29]= -Log[Cos[x/2]] + Log[Sin[x/2]]
```

The same integral, represented in a more traditional manner, can be entered from palettes or keyboard shortcuts.

```
In[30]:= ∫ 1/Sin[x] dx

Out[30]= -Log[Cos[x/2]] + Log[Sin[x/2]]
```

Many mathematical functions have traditional symbols associated with their operations and when available, these can be used instead of the fully spelled-out names. For example, you can compute the intersection of two sets using the `Intersection` function.

```
In[31]:= Intersection[{a, b, c, d, e}, {b, f, a, z}]

Out[31]= {a, b}
```

Or you can do the same computation using more traditional notation.

```
In[32]:= {a, b, c, d, e} ⋂ {b, f, a, z}

Out[32]= {a, b}
```

To learn how to enter these and other notations quickly, either from palettes or directly from the keyboard using shortcuts, refer to the 2D Expression Input section in the Front End category of the Help Browser.

### *The front end and the kernel*

When you work in *Mathematica* you are actually working with two separate programs. They are referred to as the *front end* and the *kernel*. The front end is the user interface. It consists of the notebooks that you work in together with the menu system, palettes (which are really just notebooks), and any element that accepts input from the keyboard or mouse. The kernel is the program that does the calculations. So a typical operation between the user (you) and *Mathematica* consists of the following steps, where the program that is invoked in each step is indicated in parentheses:

- enter input in the notebook (front end)
- send input to the kernel to be evaluated by pressing SHIFT-ENTER (front end)
- kernel does the computation and sends it back to the front end (kernel)
- result is displayed in the notebook (front end)

There is one remaining piece that we have not yet mentioned; that is *MathLink*. Since the kernel and front end are two separate programs, a means of communication is

necessary for these two programs to "talk" to each other. That communication protocol is called *MathLink* and it comes bundled with *Mathematica*. It operates behind the scenes, completely transparent to the user.

*MathLink* is a very general communications protocol that is not limited to front end – kernel communication, but can also be used to set up communication between the front end and other programs on your computer, programs like compiled C and Fortran code. It can also be used to connect a kernel to a word processor or spreadsheet or many other programs.

*MathLink* programming is beyond the scope of this book, but if you are interested, there are several books and articles that discuss it (see the References at the end of this book).

### *Errors*

In the course of using and programming in *Mathematica*, you will encounter various sorts of errors, some obvious, some very subtle, some easily rectified, and others not. We have already mentioned that it is possible to send *Mathematica* into an infinite loop from which it cannot return. In this section, we discuss those situations where *Mathematica* does finish the computation, but without giving you the answer you expected.

Perhaps the most frequent error you will make is misspelling the name of a function. Here is an illustration of the kind of thing that will usually happen in this case.

```
In[33]:= Sine[1.5]

         General::spell :
          Possible spelling error: new symbol name "Sine" is
            similar to existing symbols {Line, Sin, Sinh}. More…

Out[33]= Sine[1.5]
```

Whenever you type a name that is *close* to an existing name, *Mathematica* will print a warning message like the one above. You may often use such names intentionally, in which case these messages can be annoying. In that case, it is best to turn off the warnings.

```
In[34]:= Off[General::spell]
```

Now, *Mathematica* will not report that function names might be misspelled; and, when it can not find a definition associated with a misspelled function, it returns your input unevaluated.

```
In[35]:= Intergate[x^2, x]

Out[35]= Intergate[x^2, x]
```

You can turn these spell warnings back on by evaluating `On[General::spell]`.

```
In[36]:= On[General::spell]
```

Having your original expression returned unevaluated – as if this were perfectly normal – is a problem you will often run into. Aside from misspelling a function name, or simply using a function that does not exist, another case where this occurs is when you give the wrong number of arguments to a function, especially to a user-defined function. For example, the `BowlingScore` function takes a single list argument; if we accidentally leave out the list braces, then we are actually giving `BowlingScore` 12 arguments.

```
In[37]:= BowlingScore[10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]

Out[37]= BowlingScore[10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
```

Of course, some kinds of inputs cause genuine error messages. Syntax errors, as shown above, are one example. The built-in functions are designed to usually warn you of such errors in input. In the first example below, we have supplied the `Log` function with an incorrect number of arguments (it expects one or two arguments only). In the second example, `FactorInteger` operates on integers only and so the real number argument causes the error condition.

```
In[38]:= Log[2, 16, 3]

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

Out[38]= Log[2, 16, 3]

In[39]:= FactorInteger[12.5]

         FactorInteger::facn : Argument 12.5` in
            FactorInteger[12.5] is not an exact number. More…

Out[39]= FactorInteger[12.5]
```

## *Getting help*

*Mathematica* contains a vast array of documentation that you can access in a variety of ways. It is also designed so that you can create new documentation for your own functions and program in such a way that users of your programs can get help in exactly the same way as they would for *Mathematica*'s built-in functions.

If you are aware of the name of a function but are unsure of its syntax or what it does, the easiest way to find out about it is to evaluate `?function`. For example, here is the usage message for `ParametricPlot`.

```
In[40]:= ?ParametricPlot

ParametricPlot[{fx, fy}, {u, umin, umax}] produces a parametric
   plot of a curve with x and y coordinates fx and fy generated
   as a function of t. ParametricPlot[{{fx, fy}, {gx, gy}, ... },
   {u, umin, umax}] plots several parametric curves. More...
```

Also, if you were not sure of the spelling of a command (`Integrate`, for example), you could type the following to display all built-in functions that start with `Integ`.

```
In[41]:= ?Integ*

System`
Integer        IntegerExponent  IntegerQ  Integrate
IntegerDigits  IntegerPart      Integers
```

Clicking on one of these links will produce a short usage statement about that function. For example, if you were to click on the `Integrate` link, here is what would be displayed in your notebook.

```
Integrate[f, x] gives the indefinite integral of f with respect
   to x. Integrate[f, {x, xmin, xmax}] gives the definite
   integral of f with respect to x from xmin to xmax. Integrate[
   f, {x, xmin, xmax}, {y, ymin, ymax}] gives a multiple
   definite integral of f with respect to x and y. More...
```

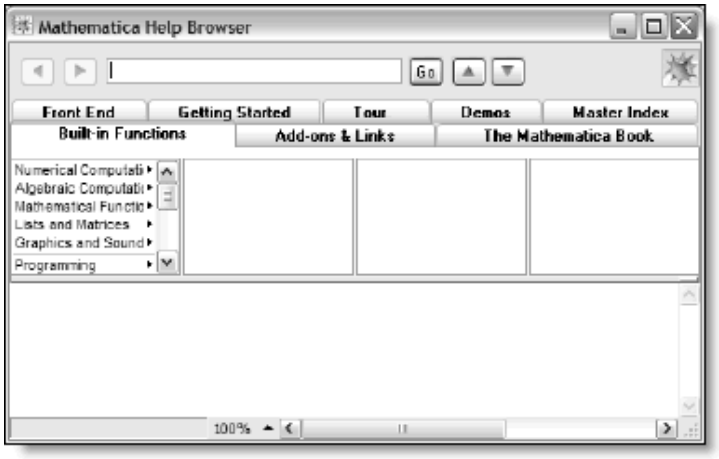
Clicking the More... hyperlink would take you directly to the Help Browser where a much more detailed explanation of this function can be found.

You can also get help by highlighting any *Mathematica* function and pressing the F1 key on your keyboard. This will take you directly into the documentation for that function in the Help Browser.

### *The Help Browser*

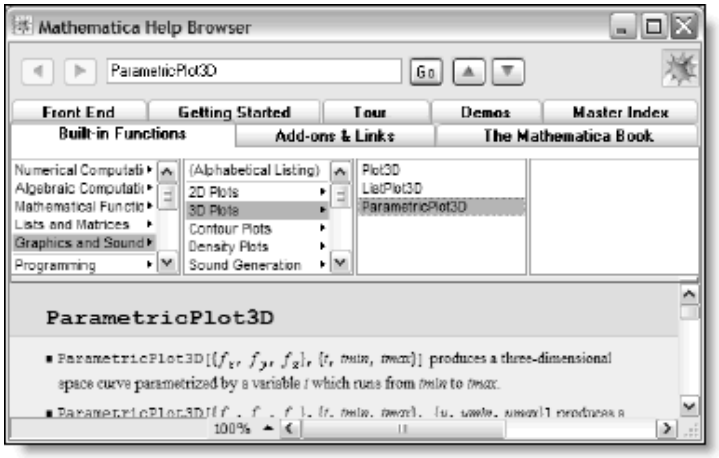
*Mathematica* contains a very useful addition to the help system called the Help Browser. The Help Browser allows you to search for functions easily and it provides extensive documentation and examples.

To start the Help Browser, select Help Browser... under the Help menu. You should quickly see something like the following:




Notice the eight category tabs near the top of the Help Browser window. Choosing the Add-ons & Links tab will give you access to all of the packages that come with each implementation of *Mathematica*. Similarly, choosing The *Mathematica* Book tab will give you access to the entire *Mathematica* book that ships with each professional version of *Mathematica*.

Suppose you were looking for information about three-dimensional parametric graphics. First click the Built-in Functions tab, then select Graphics and Sound on the left, then 3D Plots and finally ParametricPlot3D. The Help Browser should look like this:




Notice that in the main window, the Help Browser has displayed information about the `ParametricPlot3D` function. This is identical to the usage message you would get if you entered `?ParametricPlot3D`.

Alternatively, you could have clicked the Master Index tab and searched for "ParametricPlot3D" or even simply "parametric" and then browsed through the index to find what you were looking for.

Many additional features are available in the Help Browser and you are advised to consult your documentation for a complete list and description.