Making Everything Easier!™

2nd Edition

Raspberry Pi®

FOR DUMMIES®

A Wiley Brand

Learn to:

- Connect the Raspberry Pi and install the OS
- Learn programming with Scratch® and Python®
- Create electronics projects connected to the Raspberry Pi's GPIO port
- Make virtual worlds in Minecraft® and computer music with Sonic Pi

Sean McManus
Mike Cook

Bonus Chapter 1

Mathematica

In This Chapter

- Discovering the free hidden gem you already have
- Finding out about the interactive notebook
- Seeing how to manipulate algebraic equations
- Producing plots of functions
- Making dazzling 3D graphics

In the mid-eighties, the phenomenon of the killer app was born. (A *killer app* meant you bought the computer because you wanted to run a specific piece of software.) The computer was the Apple II and the software was VisiCalc. VisiCalc was the first spreadsheet software, although at that time, the term *spreadsheet* had not yet been invented. Now the Raspberry Pi has its own killer app, Mathematica.

Getting to Know Mathematica

Mathematica is described as a "system for doing mathematics" and, alongside its sister application, the Wolfram Language, is perhaps one of the best systems there is for exploring anything to do with numbers. That includes just about everything from adding up numbers to create complex, multidimensional graphics and music.

Mathematica does not come cheap (a professional version will cost you about $2,500), but the Wolfram Foundation has decided to release a version for the Raspberry Pi for free! Killer apps do not come much more lethal than that.

Mathematica is a *symbolic package*, or a CAS (*Computer Algebra System*) and, although many systems now offer symbolic manipulation, Mathematica was one of the first languages to make a computer operate like this, way

back in the late eighties. In a CAS, operations are done on symbols, not on numbers. For example, if you want to integrate the expression $2x + 1$, a symbolic package would give you the expression $x^2 + x$. A typical language will give you just a numeric answer for the problem, if it could give any answer at all, and only then provided the value of x is known. For a list of current CAS applications, see `http://en.wikipedia.org/wiki/List_of_computer_algebra_systems`.

Discovering the Notebook Interface

When you write a program, you normally do it in an editor, then the program is run, and it has an output window of some sort. Mathematica is different. It uses what is known as a *notebook*, which is an interactive document that acts as both an editor for creating the input as well as an output window. You edit that and it generates the output of the subject you are currently exploring. You can save this and add to it later, or export it in a variety of formats, including PDF. Of course, after you have a PDF file, it becomes a fixed read-only record of your notebook and you can't edit it anymore.

However, if you have a Macintosh or PC, you can replay your notebooks on them if you download the free Wolfram CDF player. CDF stands for Computer Document Format and used to be simply called the Mathematica player. This allows you to view the results of a notebook. Notebooks are interactive and allow you to record what you have done. Unlike looking at a notebook in Mathematica, if you have a CDF player, you won't be able to change anything in the notebook, but you can look at the graphics, change the view point, and change the slider bars, if you have any. This will benefit from the much faster processor you have on your main machine as opposed to the slower Raspberry Pi. However, the player is for version 9 of Mathematica, whereas the Raspberry Pi implementation is version 10, so it doesn't always fully work with your Raspberry Pi-generated notebooks.

Mathematica, as run on the Raspberry Pi, is not the fastest of programs. A good tip is to keep your eye on the CPU usage on the icon bar at the bottom of the screen. If it is fully green, you know Mathematica is working on the problem you just gave it. If it is not and you don't see any output, it means that your input did not produce an output. Maybe that was deliberate or the result of some error.

Mathematica, like all other computer languages, has a tightly defined syntax. Rather than trudge through it all, it's best if you pick it up as you go along. In the next section, we go through a sample session and get a feel of what Mathematica can do.

Starting Up

Go to the desktop and double-click the Mathematica icon. You get a splash screen and eventually two windows pop up: a blank notebook and, in front of that, an invitation to visit three Mathematica websites. Click the notebook window to bring it to the front, but at this stage, it will not have finished initializing. You have to wait until the CPU usage box stops being solid green. This might take another 100 seconds or so. When it does, you are ready to type your first input. So what is two to the power of eight? Type

```
2^8
```

and then press Return. Nothing happens: You see, what you have done is to input that number into Mathematica. In order to tell Mathematica to evaluate what you have typed, you should hold down Shift while you press Return. Click just after the 8 to get back to the line and hold down Shift and press Return. You will see this:

```
2^8
256
```

By the way, your input is shown in bold and the computer's output is not. Try another couple of calculations:

```
2^37
137438953472
Sin[1.2]
0.932039
```

Don't forget to hold down Shift as you press Return. Note here that the Sin function, like all the functions in Mathematica, has an initial uppercase character. The rest is in lowercase. Some functions that are a concatenation of two words are written in what is known as *camel case*; that is, while there is no space between the letters, each word starts with a capital letter. Also, another rule is that all the function's parameters are in square brackets. You might also have noticed that the Sin function, along with all the other trigonometry functions, uses radians for the input. Lists are bounded by braces or curly brackets, like this: { }. Parentheses, or smooth brackets ( ), are used to group arithmetic expressions when needed.

Doing Symbolic Math

Take a look at some symbolic mathematical manipulation:

```
Expand[(1+x)^6]
1 + 6x + 15x^2 + 20x^3 + 15x^4 + 6x^5 + x^6

Expand[(3+x)^2 (1 -y)^2]
9 + 6x + x^2 - 18y - 12xy - 2x^2y + 9y^2 + 6xy^2 + x^2y^2

Expand[(1+x)^2]
1 + 2x + x^2
```

As you can see, it is simple to expand a polynomial expression, but you can also do factorization and many other forms of algebraic manipulation. Notice that you enter a power with a carat (^), but it prints out in the conventional superscript format. You can even do calculus. Figure BC-1 is an example of symbolic integration.

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

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

Figure BC-1: Symbolic integration.

Plotting Functions

Where Mathematica really scores is with its graphic output: It can plot functions, draw graphs, and create graphics, all with the minimum of user input. You have to supply only the minimum amount of information, with many parameters having reasonable default values.

For example, typing

```
Plot[Sin[x], {x, 0, 2 Pi}]
```

produces the graph in Figure BC-2.

Look at the syntax of this line. It starts with the function to plot, and then it's followed by a list. The first thing in the list is the parameter to change. This is followed by the start value and finally the end value. Note the parameter list is in curly braces ({ }) and the whole function, the plot, is enclosed in square brackets ([]).

This is a programming language, so you can put that plot in a loop:

```
For[i=1, i<=10, i++, Plot[Sin[i*x], {x, -Pi, Pi}] // Print]
```

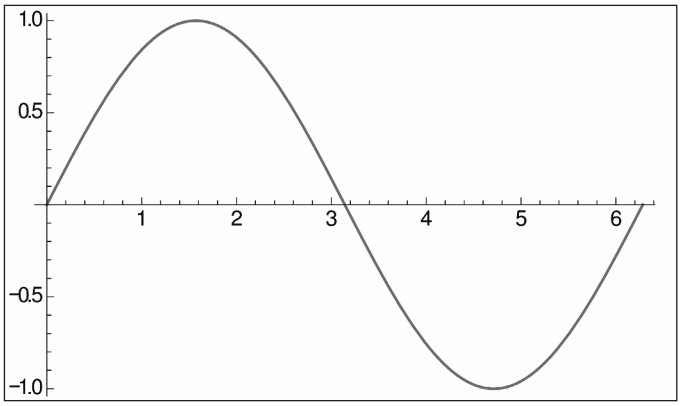

Figure BC-2: A sine wave.

Note that when you type two keys (the < and the =), this will be replaced automatically by a ≤ symbol after Mathematica processes this. The whole line is an example of a `For` loop. Just like in other languages, there is a loop variable. Here it's `i` with an initial value of 1. The loop repeats while the value of `i` is less than or equal to 10, and every time one circuit of the loop is completed, the loop index is incremented. This loop index is used to multiply the value of `x` in each plot and so in effect increases the frequency of the sine wave. Successive plots have one more cycle over the two-Pi range the function is plotted over. Finally, the // Print tells Mathematica to print the graph in each iteration of the loop, so ten plots are produced. These are shown in Figure BC-3.

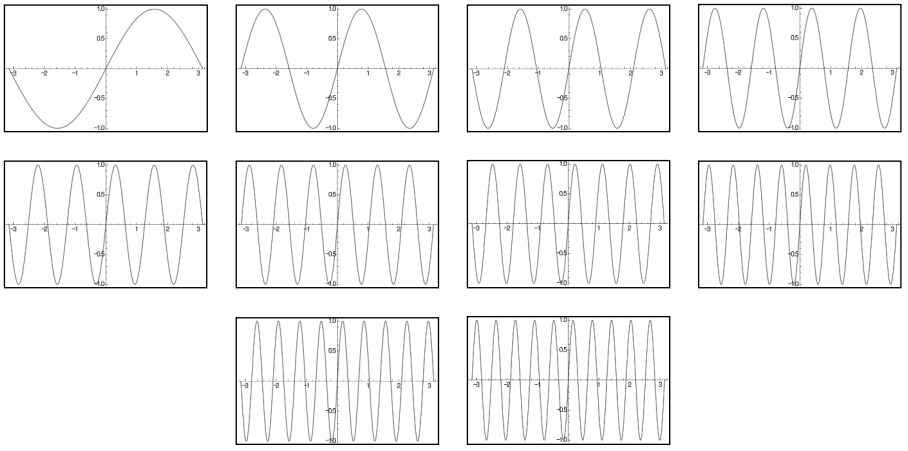

Figure BC-3: Ten plots with an increasing frequency in each.

You can feed different values into functions, but it would be great to be able to do this in real time. In Mathematica, you can have interactive control of variables in a function through the use of a slider or, as it is known in Mathematica, a *manipulate*. Figure BC-4 shows the code for doing this on a sine wave.

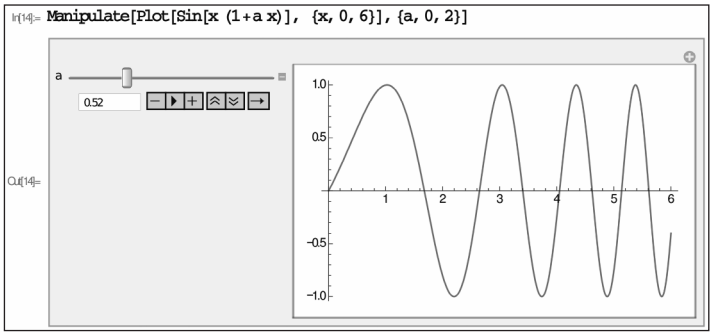

Figure BC-4: A manipulate slider.

The parameter `a` controls how quickly the sine wave increases in frequency. You can drag the slider and see the results instantly.

Many functions

So far in this chapter, we have discussed plotting just a single function, but you can plot many functions on the same axis to easily compare the shapes you get. The only problem is that without differentiating between the lines used to draw each function, it's hard to see what is happening. You can get around this problem in a few ways — for example, by defining the line's colors or by defining a dash pattern. Figure BC-5 is a plot of the first three powers of `x` shown on the same axis.

Notice the arrow after the `PlotStyle` statement. This is formed by typing a minus followed by a right chevron or angle bracket (`>`). The Mathematica notebook converts this into a right-pointing arrow. In the same way, Mathematica changes input such as `<=` into ≤ along with other similar substitutions.

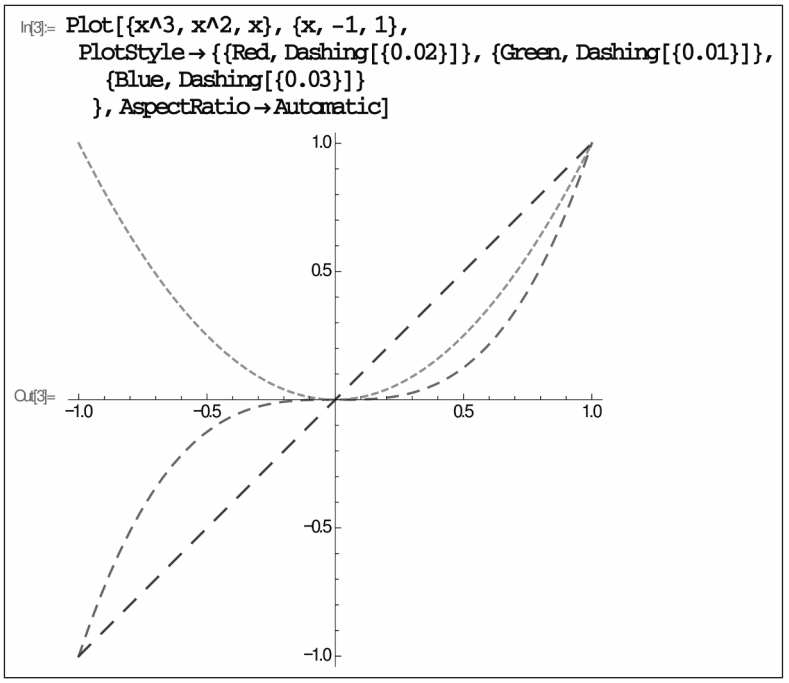

Figure BC-5: Three powers of x.

Parametric plotting

The function plots we have used so far are all functions plotted against a range of input values. You can plot one function against another by using a parametric plot. If these functions are Sin or Cos functions, you get the well-known family of Lissajous figures. If a Sin function defines the displacement in the X direction and another defines the displacement in the Y direction, you get a straight diagonal line. This is because both X and Y reach the maximum displacement at the same time. However, if the two waves are 90 degrees out of phase, you get a circle. Now if the frequency of the X and Y sine waves are multiples of each other, you get much more interesting curves. The code to do this is

```
For[n=1, n<4, n++,
ParametricPlot[ {Sin[n t], Sin[(n+1) t]}, {t, 0, 2Pi}] //
               Print]
```

Figure BC-6 shows the output of this.

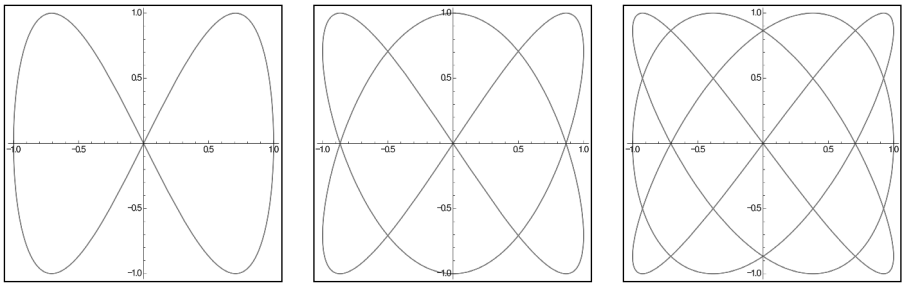

Figure BC-6: Three Lissajous figures.

As a challenge, see whether you can make a version that uses a manipulate slider to control the ratio of the two Sin functions.

So far, all the functions we have shown you are made up from a single input of code. One of the strengths of Mathematica is that you can define a function and then use that function in future programs. It becomes part of the language. These functions can take in parameters, which makes them even more flexible. In fact, they act in rather the same way as functions do in any other computer language. First of all, define a function by typing the following:

```
  star[n_,radius_,twist_]:=
  Polygon[
  Flatten[
  Table[ {  {Cos[t-twist],
             Sin[t-twist]},
             radius {Cos[t + twist +Pi/n],
             Sin[t+twist+Pi/n]}
         },
  {t, 0,2Pi,2Pi /n}
  ],
  1
] ]
```

Now place your cursor at the end, hold down Shift, and press Return. Exactly nothing happens. To see the results of this, you have to tell Mathematica to draw the figure and give it some parameters. The command

```
Show[Graphics[star[15,3,0.5]],AspectRatio -> 1]
```

produces the output shown in Figure BC-7.

You have displayed a 15-pointed star with a 0.5 radian twist applied to each of the points. This sort of thing cries out for slider control. Figure BC-8 shows the line of code needed to do this, and it also shows the result.

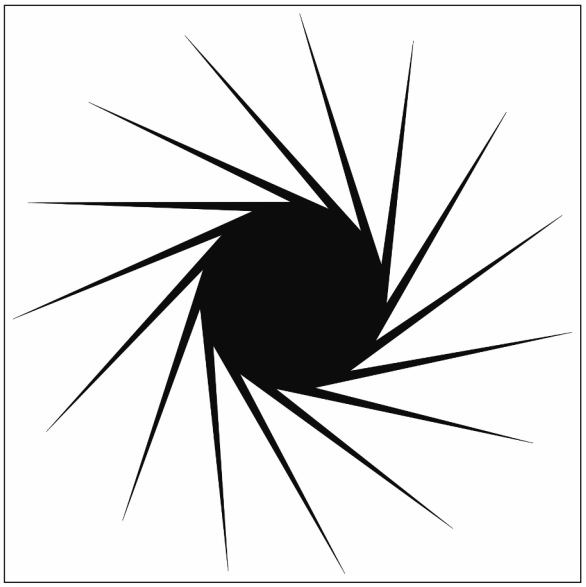

Figure BC-7: The Star output.

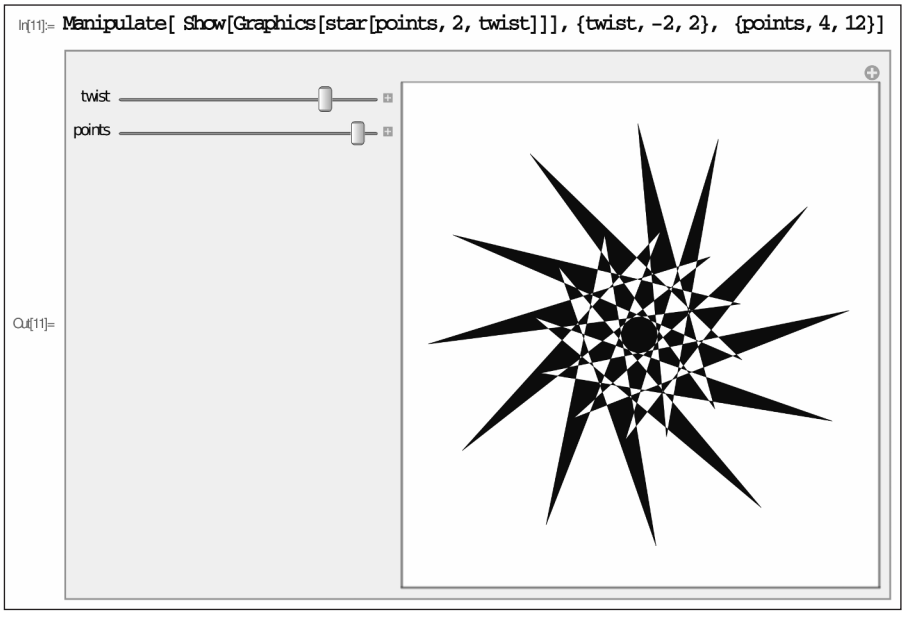

Figure BC-8: Two sliders control the star.

You can see the two parameters used to control the range of the twist, and the number of points. Try changing these ranges to give the sliders even greater control.

3D Graphics

So far, all the graphic output you have been looking at is in 2D. One of the most breathtaking aspects of Mathematica is the ease with which it handles 3D plotting.

To get a feel for this, type in the following line:

```
SphericalPlot3D[1, {e, 0, Pi}, {t, 0, 3 Pi/2}]
```

This plots a sphere, with a quarter of the surface missing. The results are shown in Figure BC-9. Notice, however, when you hover your mouse over the output, the cursor changes into a pair of curved arrow-headed lines. When you see this, click and drag the figure to see it from a different viewpoint.

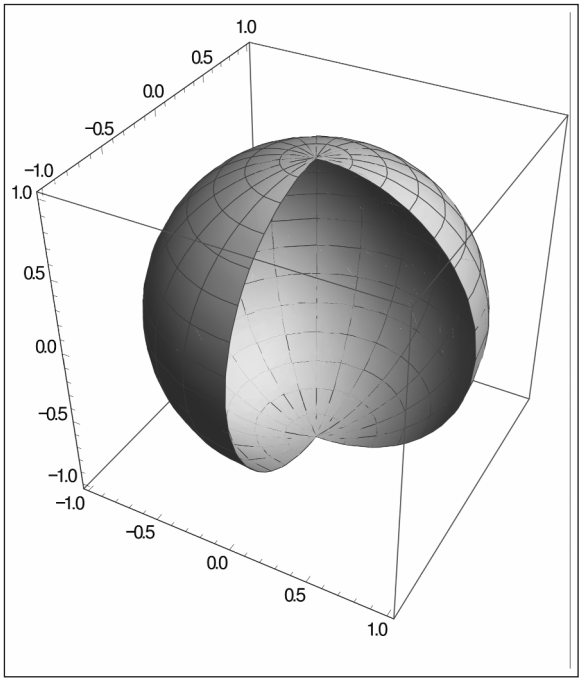

Figure BC-9: Three quarters of a sphere.

If you are a chemist, you might recognize this next figure as a d-orbital. This is one of the wave function shapes an outer electron shell of an atom can take on. This is important because it affects the way that this atom can bond with others.

```
SphericalPlot3D[Sin[t] Cos[t] Sin[f], {t, 0, Pi}, {f, 0, 2
          Pi}]
```

This produces the output shown in Figure BC-10.

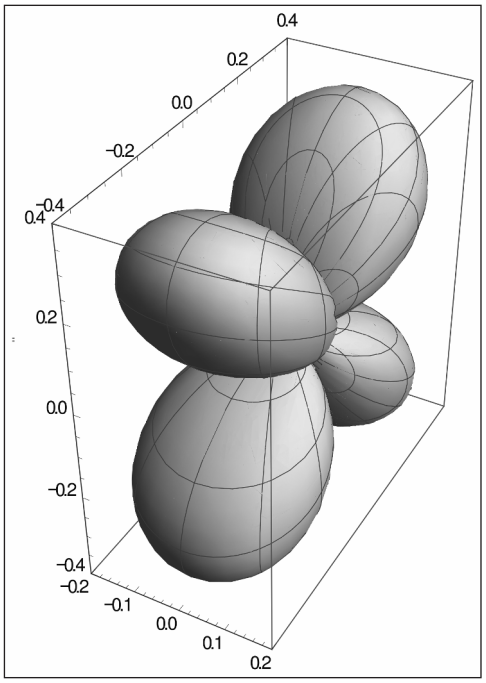

Figure BC-10: A d-orbital shape.

Finally, a favorite shape of ours is a sinc function or Sin x over x. Although this is normally shown in 2D, you can make a 3D version quite easily:

```
   Plot3D[Sin[Sqrt[x^2 + y^2]]/Sqrt[x^2 + y^2],
   {x, -6 Pi, 6 Pi}, {y, -6 Pi, 6 Pi},
   Boxed -> False, Mesh -> False, PlotPoints -> 60,
PlotRange -> All, Axes -> False]
```

The output of this is shown in Figure BC-11. Note once again that you can manipulate the viewing angle. Figure BC-12 shows various views of the output of this program.

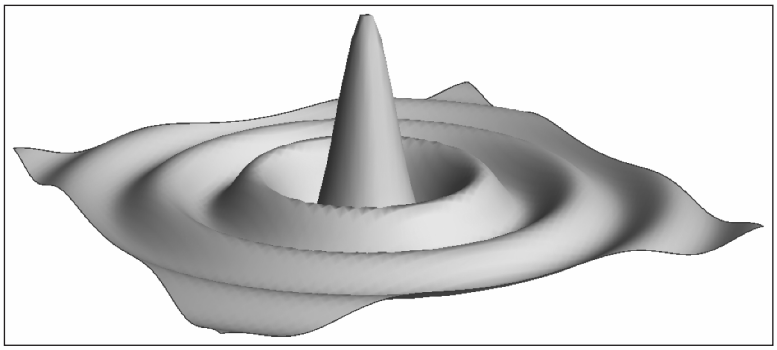

Figure BC-11: A sinc function.

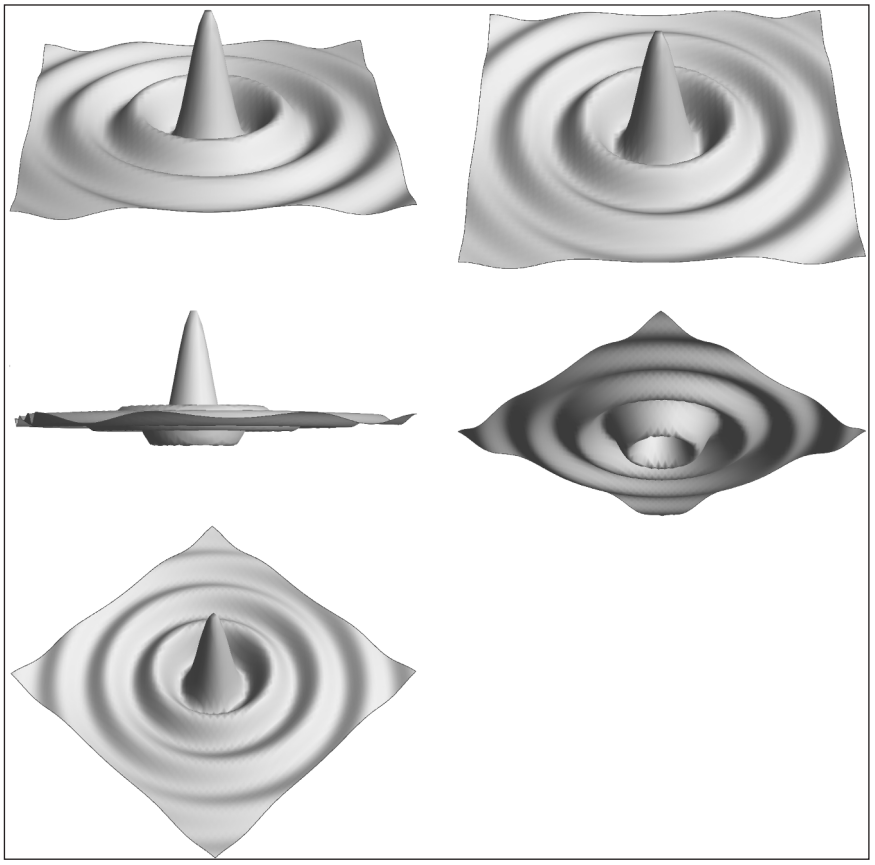

Figure BC-12: A sinc function from various viewing points.

Exploring Further Functions

There is plenty more to Mathmatica that we haven't even begun to mention. For example, there are functions for accessing the GPIO pins of the Raspberry Pi, and there are ways of making and manipulating sounds, although not all of them are implemented in the Raspberry Pi version of Mathematica yet. You can also use Mathmatica to control the Raspberry Pi camera or make a GPS tracker. These are detailed on the Wolfram website at `www.wolfram.com/raspberry-pi/` along with access to the full documentation and loads of examples.

A few points of Mathmatica are less than perfect, but they look like they might be cleared up in later versions. Sometimes, for example, a new notebook is positioned in such a way that the whole of the title bar is just outside the desktop. This means that you can't resize or drag it about. The solution we found was to right-click the window and choose Toggle Full Screen. When it expanded to the full screen, choosing that option again resulted in a correctly placed window.

Sometimes the update rates when using the sliders are not as quick as you might like. A video on the Wolfram website says they are trying to improve this.

We also found that exporting a notebook as a PDF crashed Mathematica if there was too much data in it. This was especially troublesome when dealing with 3D plots. Also, saving a notebook as a web page worked well, but it did take a very long time (more than half an hour for a busy page).

However, all in all, Mathematica is well worth a look. You have many months of wonder ahead exploring just this one package.