1 Introduction to *Mathematica*

1.1 What is *Mathematica*?

Mathematica is a computer program (or, more correctly, a system of computer programs) for performing mathematical operations such as symbolic manipulation, numerical calculations, graphics, and programming. *Mathematica* is sometimes described as a computer algebra system, probably in deference to the strong symbolic but somewhat limited numerical capabilities of its earliest versions, but it has evolved into system that is now a very practical tool for large scale numerical calculations. Its applications can range from simple calculations to complicated programs, and *Mathematica* can be a powerful tool for quantitative geoscientists such as geologists, geographers, geophysicists, hydrologists, and oceanographers. Because it is a general mathematics system rather than a task-specific application, however, its utility may not be apparent at first glance. It does not, after all, solve specific geoscientific problems any more than do programming languages such as C or FORTRAN or spreadsheet programs. Over the years I have encountered many geoscientists who know that Mathematica exists, but are not sure what it does with regard to the specific problems they face in their research or professional practice. The short answer is that it does nothing geoscientific in particular, but has the potential to do just about anything that a user can imagine. *Computational Geosciences with Mathematica* is intended to help fill that gap a by illustrating how *Mathematica* can be used to formulate, solve, and visualize a variety of problems of interest to geoscientists.

Mathematica was first published by Wolfram Research in 1988 and the latest version, 5.0, was released during Summer 2003. This book was written using versions of *Mathematica* ranging from 4.1 through 5.0, and care has been taken to ensure that the examples will work with both version 5.0 and its immediate predecessor, version 4.2. Version 5.0 is available for Windows 98/Me/NT 4.0/2000/XP and several versions of the Unix (including Macintosh OS X) and Linux operating systems. Specialized versions are available for web applications, grid computing using computer clusters or multiprocessor computers, and student use.

The functionality of *Mathematica*, which includes many standard packages with specialized functions in areas such as statistics and graphics, can be expanded with packages available through Wolfram Research, other commercial sources, or the public domain. Packages that may be of interest to geoscientists, but which are not covered in this book, include a database access kit, digital image processing, exper-

imental data analysis, real time 3-D graphics, fuzzy logic, neural networks, signal processing, time series analysis, and wavelets. Readers interested in those packages should contact Wolfram Research for more information about their capabilities and availability.

1.2 Getting Help

Mathematica offers several kinds of documentation and online help. One source is *The Mathematica Book* (Wolfram, 1999), the current edition of which is written for version 4.0. A paper copy is included with professional (but not student) versions of *Mathematica*. *Mathematica* also offers a Help menu that includes a Help Browser (which can access a digital copy of *The Mathematica Book*), a ten-minute tutorial, a hyperlink to a web information center maintained by Wolfram Research (http://library.wolfram.com/infocenter), and a hyperlink to the general Wolfram Research web site (http://www.wolfram.com). If you know the name of a function, typing **?***function* then pressing Enter will return a brief description of the function with a hyperlink to more information. If you only know part of the function name, typing a question mark followed at least the first letter of the function name and then pressing Command-k will return a list of *Mathematica* functions that match the letters typed.

There are a number of good introductory books about *Mathematica*, in particular *The Beginner's Guide to Mathematica Version 4* (Glynn and Gray, 2000). Another useful source of information is the comp.soft-sys.math.mathematica newsgroup, which can be accessed through http://groups.google.com. For those wishing to retain expert help, Wolfram Research maintains a list of accredited *Mathematica* consultants.

1.3 Installing and Running Mathematica

Virtually any modern personal computer or workstation should have enough memory and computational speed to run *Mathematica* although, as with all software, the more memory and the faster the processor the better. The examples in this book were all developed using an iMac computer with a 700 MHz G3 processor, 512 Mb RAM, and Macintosh OS X. Free 30 day trial versions of *Mathematica* are available from Wolfram Research (http://www.wolfram.com).

If you are installing *Mathematica* on your own personal computer, insert the CD and follow the directions that appear on your screen. Start up *Mathematica* as you would any other program. If you are using *Mathematica* over a computer network, consult your system administrator or help desk for details.

The Getting Started directory in the *Mathematica* help browser offers several introductory lessons that may be useful for first-time users. To access them, select the Help Browser from the Help menu. The Help Browser item Tour includes a 10 minute introduction that covers many features of *Mathematica*, and Getting

Started offers an introduction to entering and executing basic *Mathematica* commands.

Although it may not be apparent, *Mathematica* consists of two programs: a kernel that performs calculations and a front end that handles input and output. Although it is possible to use *Mathematica* with a front end on one computer and the kernel on another, this book assumes that both are running on the same computer. When you click on the *Mathematica* icon, you will start the front end. The first time you execute a statement from the front end, there will be a short pause while the kernel starts.

To install the CompGeosci.m package included with this book, copy it from the CD to one of the directories along *Mathematica*'s default file search path. This will differ among operating systems and versions. To obtain a list of the default paths, type **`$Path`** and press the Enter key. Some of the directories listed may be accessible only to system administrators on multi-user systems, in which case the package may have to be installed in a local user library. On the system being used to write this book, the package has been put into the directory /Users/bill/Library/Mathematica/Applications. Consult your system administrator or help desk for guidance if you are using a multi-user system.

1.4 How the Book is Organized

Each of the chapters in *Computational Geosciences with Mathematica* was prepared as a *Mathematica* document known as a notebook. Therefore, readers with copies of *Mathematica* can open the accompanying digital versions of the chapters and follow the calculations as they read. Notebooks can contain combinations of text, mathematical input and output, graphics, and even sound. Mathematical variables are generally denoted by *italics*, whereas *Mathematica* functions and references to specific variables within the *Mathematica* examples are denoted using the same **`courier`** font that *Mathematica* uses for its default input and output. Appendix A is a list of *Mathematica* functions included in the CompGeosci package accompanying this book and Appendix B (CD only) is an overview of color in *Mathematica* graphics.

The beginning of each chapter notebook includes a series of **`Needs`** statements, which tell *Mathematica* which of its standard packages will be used in that notebook. If you are following the examples on your own computer, make sure to execute the **`Needs`** statement before any others in the notebook. Many of the examples in this book make use of data sets contained on the accompanying CD. If you plan to follow the examples, you can copy the data files to a directory on your hard disk or access them directly from the CD. You will, however, have to change the file path given in the examples to correspond to the location on your computer.

1.5 A Brief Tour of *Mathematica*

1.5.1 Symbolic and Numerical Operations

When *Mathematica* is started with its default settings, two things appear: a blank window named Untitled and a Basic Input palette filled with mathematical operators and Greek letters. *Mathematica* accepts expressions written either in standard text or using operators and symbols pasted from the Basic Input palette. To enter an expression, position the cursor near the top of the blank window, click the mouse to make it active, type in a simple expression, and press either Enter or Shift-Return. Using 2/3 as an example, the result is:

```
In[1]:= 2/3
```

$$\text{Out[1]=}\ \frac{2}{3}$$

After pressing enter, the intial expression is assigned an input number (in this case 1) and a corresponding output line is shown immediately below. *Mathematica* distinguishes between exact integer expressions and approximate numerical expressions, and therefore returned a value of 2/3 rather than 0.666667. Important irrational numbers such as π are also manipulated as symbols unless *Mathematica* is forced to assign a numerical approximation. Purely symbolic expressions can also be used, for example

```
In[2]:= a/b
```

$$\text{Out[2]=}\ \frac{a}{b}$$

Input and output numbers are reset each time the *Mathematica* kernel is started. Therefore, if you start *Mathematica,* save and close the window, and then open a new window the input and output numbers will continue in sequence because the kernel was not restarted.

One of *Mathematica*'s strengths is its ability to perform symbolic manipulation, for example algebra and calculus. It can find symbolic solutions to many kinds of equations, for example

```
In[3]:= Solve[a/b == 4, b]
```

$$\text{Out[3]=}\ \left\{\left\{b \to \frac{a}{4}\right\}\right\}$$

Likewise, the solution to $3x + 7 = 18$ is

```
In[4]:= Solve[3 x + 7 == 18, x]
```

$$\text{Out[4]=}\ \left\{\left\{x \to \frac{11}{3}\right\}\right\}$$

Note that multiplication can be specified using an asterisk (**3 * x**), by placing a space between two variables (**3 x**), or by using the multiplication operator from the

Basic Input palette (**3 × x**). As discussed in Chapter 3, matrix and vector multiplication is slightly more specific and the multiplication operators cannot be switched indiscriminantly. The same approach works for sets of equations

```
In[5]:= Solve[{2 x + 6 y == 18, 7 x - 8 y == 7}, {x, y}]
```

$$\text{Out[5]= } \left\{\left\{x \to \frac{93}{29}, y \to \frac{56}{29}\right\}\right\}$$

and equations involving real numbers

```
In[6]:= Solve[a/b == 4.0, b]

Out[6]= {{b → 0.25 a}}
```

Solve is one of *Mathematica*'s standard functions, which all begin with uppercase letters and have arguments enclosed in square brackets. There are hundreds of standard functions, and hundreds more in packages accompanying the standard *Mathematica* distribution. They are listed alphabetically in *The Mathematica Book* and can also be viewed using the Help Browser. *Mathematica* uses curly braces, **{}**, to enclose lists of expressions or variables such as the lists of two equations and two variables above. It can also evaluate just about any derivative or integral that is likely to be included in standard mathematical references. A simple example, the derivative of x^2 with respect to x, is

```
In[7]:= ∂x x^2

Out[7]= 2 x
```

Integrating the result to recover the original expression,

```
In[8]:= ∫ 2 x dx

Out[8]= x^2
```

The derivative and integral symbols were pasted into the *Mathematica* notebook by clicking on the Basic Input palette. If the limits of integration are specified, *Mathematica* will also calculate a definite integral.

```
In[9]:= ∫_a^b 2x dx

Out[9]= -a^2 + b^2
```

Say we know the values of a and b. They can be substituted into the result above using a replacement rule specified with the **/.** operator. For example, if $a = 3.0$ and $b = 7.2$

```
In[10]:= % /. {a → 3., b → 7.2}

Out[10]= 42.84
```

Using the replacement rule evalutes the expression with $a = 3.0$ and $b = 7.2$ only in this instance, and does not permanently change the value of the expression. The

% sign is shorthand for the previous output, and **%%** is shorthand for the output line before that. Output lines in general can be referenced using either **%***n* or **Out[***n***]**, where *n* is the output line number. Alternatively, the the definite integral could have been evaluated numerically by using real numbers for the limits of integration.

```
In[11]:= ∫_3.^7.2 2 x dx
Out[11]= 42.84
```

The **=** sign is used to permanently assign values to variables. Variables can be numerical values

```
In[12]:= x = 7.2
Out[12]= 7.2
```

lists or tables of values

```
In[13]:= data = {1.2, 4.6, 9.2, 4.9}
Out[13]= {1.2, 4.6, 9.2, 4.9}
```

or the results of operations

```
In[14]:= solution = Solve[3 z == 4.2, z]
Out[14]= {{z → 1.4}}
```

Once a value is assigned to a variable name, it can be used like any other variable. For example,

```
In[15]:= √x
Out[15]= 2.68328
```

because we previously assigned the value of 7.2 to **x**. To ensure that it does not cause confusion further on, we can also clear the value of **x**.

```
In[16]:= Clear[x]
```

In can sometimes be desirable to suppress output, which can be done with a semicolon.

```
In[17]:= sinx = Sin[10.°];
```

In this case, a result is calculated and assigned to the variable name **sinx** but is not displayed. Entering the variable name will display the result

```
In[18]:= sinx
Out[18]= 0.173648
```

Like other computer languages, *Mathematica* requires angular measurements to be specified in radians. The built-in variable **Degree** is a conversion factor ($\pi/180$)

that converts angular measurements in degrees to radians. To convert radians to degrees, divide by **Degree**. *Mathematica* also recognizes commonly used mathematical symbols such as π, i, ∞, and e.

There are several methods that can be used to force *Mathematica* to return a numerical approximation of an exact integer. First, an integer expression can be followed by the expression **//N**.

```
In[19]:= 2/3 //N

Out[19]= 0.666667
```

Another way to force numerical output is to use *Mathematica*'s **N** function.

```
In[20]:= N[2/3]

Out[20]= 0.666667
```

A third way to force numerical output is to make at least one of the integers into a real number by adding a decimal point.

```
In[21]:= 2/3.

Out[21]= 0.666667
```

Mathematica will approximate a value for irrational numbers such as π

```
In[22]:= N[π]

Out[22]= 3.14159
```

or e

```
In[23]:= N[e]

Out[23]= 2.71828
```

If asked to give a numerical value for the imaginary number i, *Mathematica* returns

```
In[24]:= N[i]

Out[24]= 0. + 1. i
```

Mathematica's early versions used text input and output of expressions, but recent versions have included sophisticated mathematical notation and typsetting capabilities. The result is that many *Mathematica* functions can be specified using fairly traditional mathematical notation or simple text-only input. For example, the derivative and integral above can also be expressed as

```
In[25]:= D[x^2,x]

Out[25]= 2 x
```

and

```
In[26]:= Integrate[2 x, x]
Out[26]= x^2
```

The definite integral of $2x$ from $a = 3.0$ to $b = 7.2$ can be specified as

```
In[27]:= Integrate[2 x, {x, 3.0, 7.2}]
Out[27]= 42.84
```

Likewise, the square root of 2.8 can be represented by either

```
In[28]:= √2.8
Out[28]= 1.67332
```

or

```
In[29]:= Sqrt[2.8]
Out[29]= 1.67332
```

or

```
In[30]:= 2.8^(1/2)
Out[30]= 1.67332
```

Special symbols such as π, i, and e can be represented using the text equivalents **Pi**, **I**, and **E**.

1.5.2 Vector and Matrix Operations

Mathematica treats vectors of symbols, integers, and real numbers as lists and matrices as lists of lists. A list of data might be

```
In[31]:= data = {1.2, 4.8, 2.8, 7.2, 9.1, 6.5}
Out[31]= {1.2, 4.8, 2.8, 7.2, 9.1, 6.5}
```

whereas one list is used to represent each row of a matrix using a **Table**.

```
In[32]:= m = {{a, b}, {c, d}}
Out[32]= {{a, b}, {c, d}}
```

Elements of lists or tables can be isolated using either **Part** or double square brackets **[[]]**. The first element in the second row of **m** is

```
In[33]:= Part[m, 2, 1]
Out[33]= c
```

or, equivalently,

```
In[34]:= m[[2, 1]]
Out[34]= c
```

Matrices can also be filled with values following some functional relationship by using the **Table** function.

```
In[35]:= Table[i * j, {i, 1, 3}, {j, 1, 3}]

Out[35]= {{1, 2, 3}, {2, 4, 6}, {3, 6, 9}}

In[36]:= MatrixForm[%]

Out[36]//MatrixForm=
1 2 3
2 4 6
3 6 9
```

Matrices can be displayed in more traditional form using **//MatrixForm** or **MatrixForm[]**.

```
In[37]:= m// MatrixForm

Out[37]//MatrixForm=
a b
c d

In[38]:= MatrixForm[m]

Out[38]//MatrixForm=
a b
c d
```

They can also be constructed by clicking on the matrix button in the Basic Input palette. Many of *Mathematica*'s functions are listable, meaning that they can be applied to lists (or lists of lists). To calculate the square root of each element in **data**, for example, apply the square root function to the entire list.

```
In[39]:= √data

Out[39]= {1.09545, 2.19089, 1.67332, 2.68328, 3.01662, 2.54951}
```

Squaring the list returns the original values.

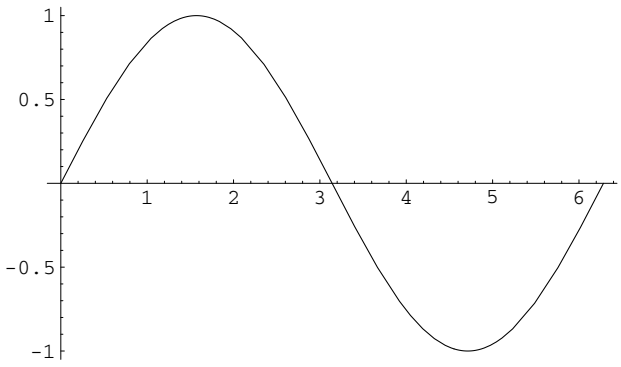
```
In[40]:= %^2

Out[40]= {1.2, 4.8, 2.8, 7.2, 9.1, 6.5}
```

Chapter 3 discusses mathematical operations on matrices, including dot and cross products.

1.5.3 2-D and 3-D Graphing

Mathematica contains functions for 2-D and 3-D graphing of functions, lists, and arrays of data. The following statement plots sin x over the range of $0 \le x \le 2\,\pi$.

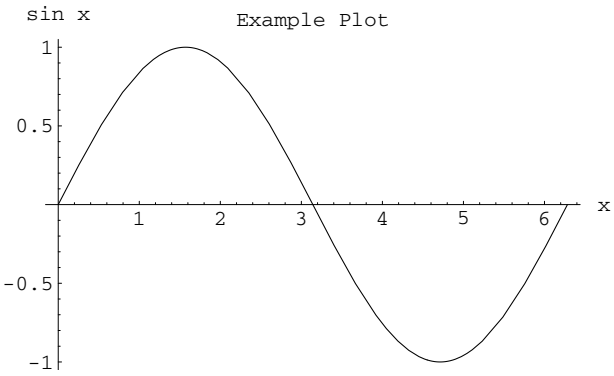
```
In[41]:= Plot[Sin[x], {x, 0, 2 π}]
```




```
Out[41]= -Graphics-
```

This statement adds a title and labels to the two axes.

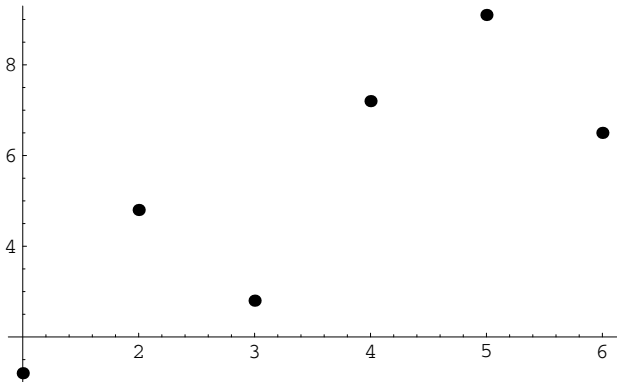
```
In[42]:= Plot[Sin[x], {x, 0, 2 π}, PlotLabel- > "Example Plot",
           AxesLabel → {"x", "sin x"}]
```




```
Out[42]= -Graphics-
```

A different function, **ListPlot**, is used for lists of data. If a list of single values is given, for example the list **data** defined above, **ListPlot** will assume that they are dependent variables and that the independent variable has the values 1, 2, 3 ...

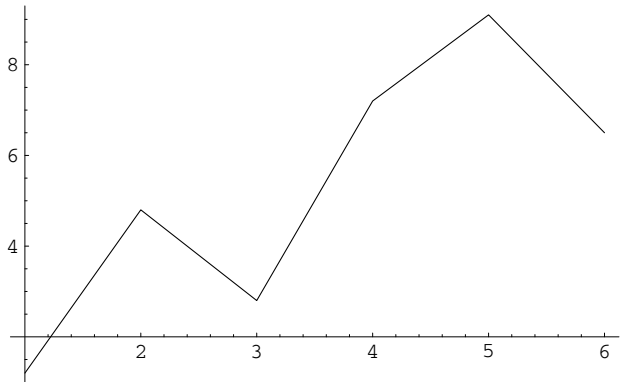
```
In[43]:= ListPlot[data, PlotStyle- > PointSize[0.02]]
```




```
Out[43]= -Graphics-
```

The dots can be connected using the option **PlotJoined → True**.

```
In[44]:= ListPlot[data, PlotJoined → True]
```



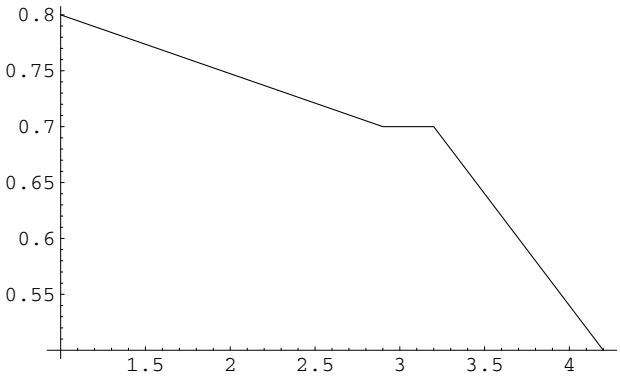

```
Out[44]= -Graphics-
```

Here is a data list with x and y values.

```
In[45]:= {{1., 0.8}, {2.9, 0.7}, {3.2, 0.7}, {4.2, 0.5}}
Out[45]= {{1., 0.8}, {2.9, 0.7}, {3.2, 0.7}, {4.2, 0.5}}
```

In this case, *Mathematica* plots the first element of each pair as the independent variable and the second element as the dependent variable.

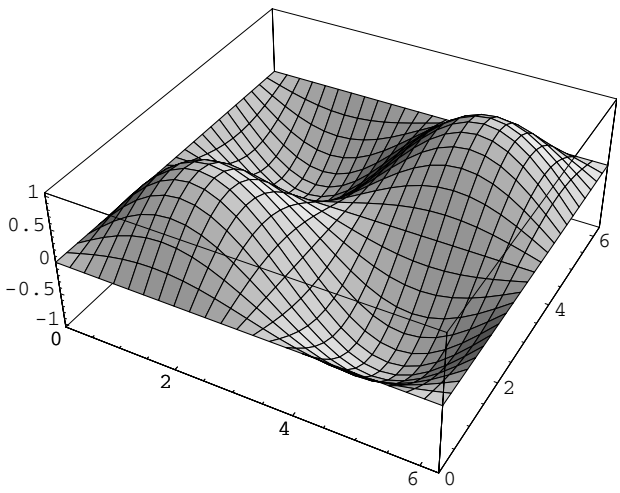
```
In[46]:= ListPlot[%, PlotJoined → True]
```




```
Out[46]= -Graphics-
```

Functions of two variables can be visualized as 3-D surface plots, contour plots, or density plots.

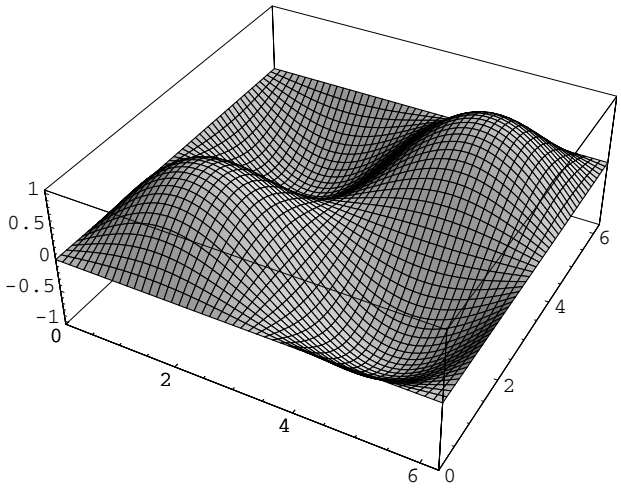
```
In[47]:= Plot3D[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel]
```




```
Out[47]= -SurfaceGraphics-
```

As with other *Mathematica* functions, options can be used to control the details of the plots. The plot below sets the number of points at which the function is evaluate to 50 instead of the default value of 25.

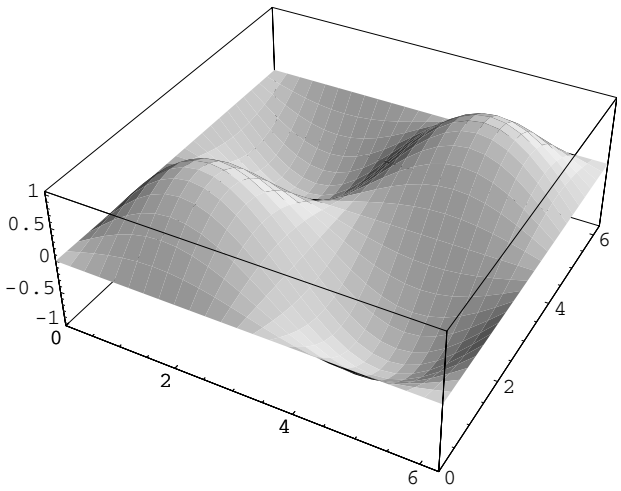
```
In[48]:= Plot3D[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, PlotPoints → 50]
```




```
Out[48]= -SurfaceGraphics-
```

and this one removes the mesh.

```
In[49]:= Plot3D[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, Mesh → False]
```

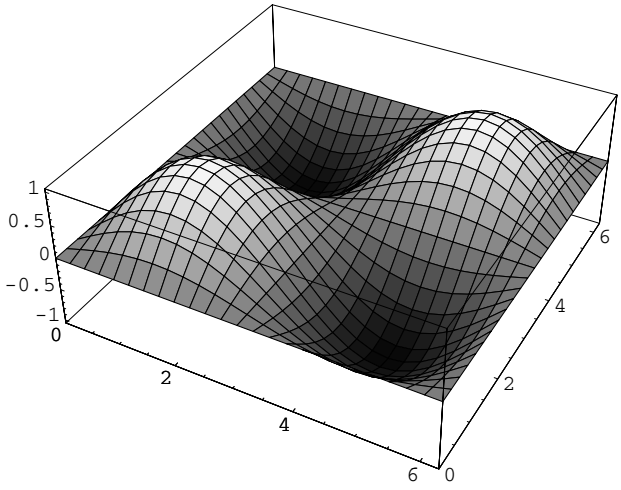



```
Out[49]= -SurfaceGraphics-
```

The **Plot3D** default is to shade surfaces using three simulated colored light sources (rendered here using gray levels; see Appendix C for a detailed discussion of color

and lighting). Setting **Lighting → False** removes the lighting and shades the surface according to its height.

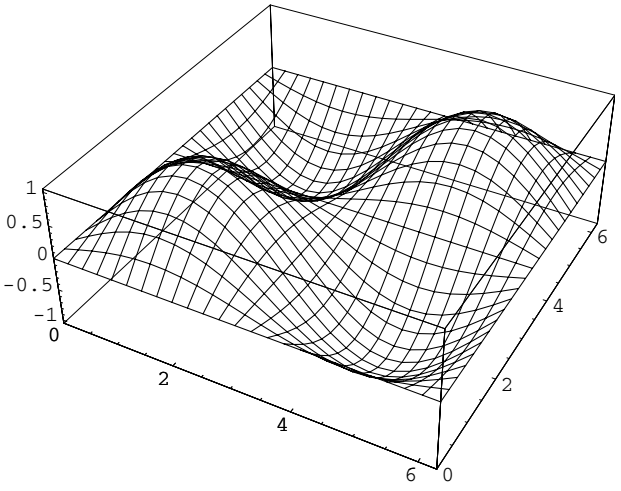
```
In[50]:= Plot3D[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, Lighting → False]
```




```
Out[50]= -SurfaceGraphics-
```

Setting **Shading → False** produces a wire-mesh plot and using **HiddenSurface → False** renders the surface transparent.

```
In[51]:= Plot3D[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, Shading → False,
           HiddenSurface → False]
```



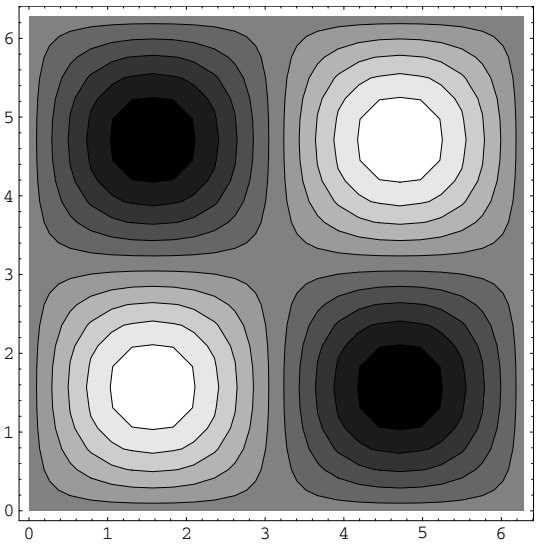

```
Out[51]= -SurfaceGraphics-
```

To see a complete list of the options available for any *Mathematica* function, use **Options[***function_name***]**.

```
In[52]:= Options[Plot3D]

Out[52]= {AmbientLight → GrayLevel[0], AspectRatio → Automatic,
          Axes → True, AxesEdge → Automatic, AxesLabel → None,
          AxesStyle → Automatic, Background → Automatic,
          Boxed → True, BoxRatios → {1, 1, 0.4},
          BoxStyle → Automatic, ClipFill → Automatic,
          ColorFunction → Automatic,
          ColorFunctionScaling → True,
          ColorOutput → Automatic, Compiled → True,
          DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
          DisplayFunction :→ $DisplayFunction, Epilog → {},
          FaceGrids → None, FormatType :→ $FormatType,
          HiddenSurface → True, ImageSize → Automatic,
          Lighting → True, LightSources → {{{1., 0., 1.},
              RGBColor[1, 0, 0]}, {{1., 1., 1.},
              RGBColor[0, 1, 0]}, {{0., 1., 1.},
              RGBColor[0, 0, 1]}}, Mesh → True,
          MeshStyle → Automatic, Plot3Matrix → Automatic,
          PlotLabel → None, PlotPoints → 25,
          PlotRange → Automatic, PlotRegion → Automatic,
          Prolog → {}, Shading → True,
          SphericalRegion → False, TextStyle :→ $TextStyle,
          Ticks → Automatic, ViewCenter → Automatic,
          ViewPoint → {1.3, -2.4, 2.},
          ViewVertical → {0., 0., 1.}}
```

The function **ContourPlot** works in a similar manner, but with different options.

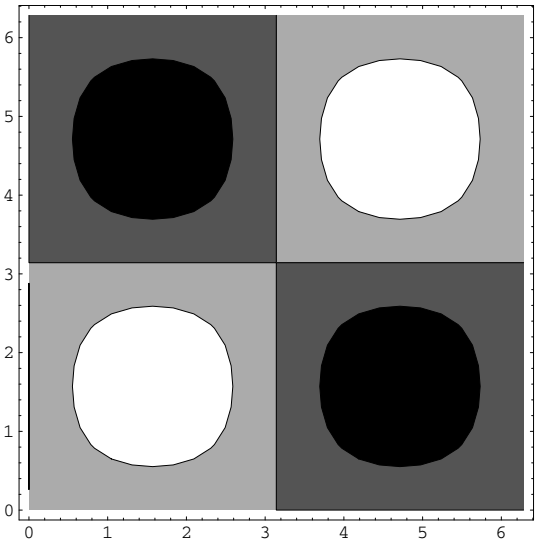
```
In[53]:= ContourPlot[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel]
```




```
Out[53]= -ContourGraphics-
```

Here is the same function plotted with 3, instead of the default 10, contours.

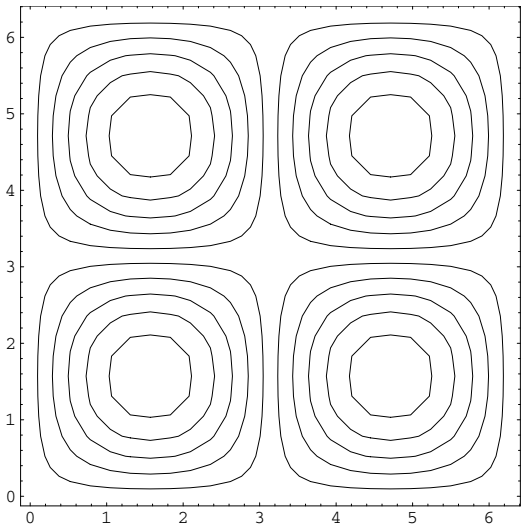
```
In[54]:= ContourPlot[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, Contours → 3]
```




```
Out[54]= -ContourGraphics-
```

This time, without shading the contour intervals.

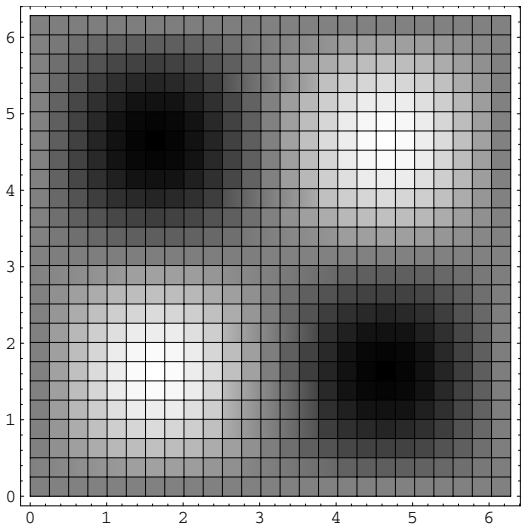
```
In[55]:= ContourPlot[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, ContourShading → False]
```




```
Out[55]= -ContourGraphics-
```

Density plots display a function of two variables using continuous shades or colors instead of contour intervals. Here is one with the default mesh

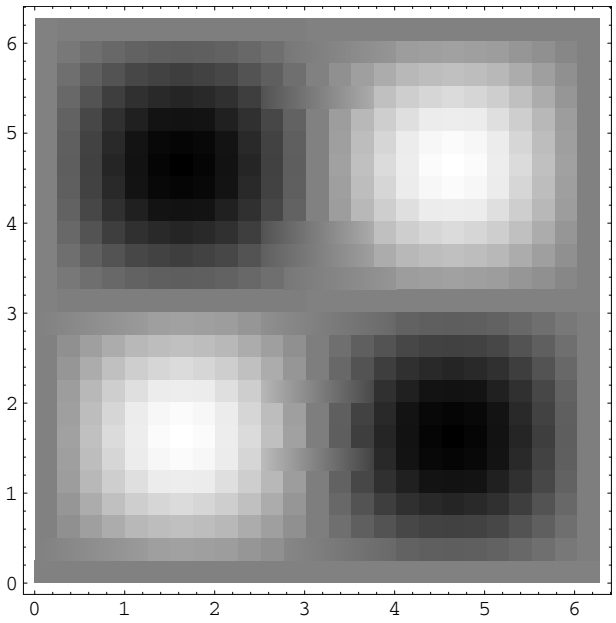
```
In[56]:= DensityPlot[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel]
```




```
Out[56]= -DensityGraphics-
```

and one without the default mesh.

```
In[57]:= DensityPlot[Sin[x] Sin[y], {x, 0, 2 π}, {y, 0, 2 π},
           ColorOutput → GrayLevel, Mesh → False]
```



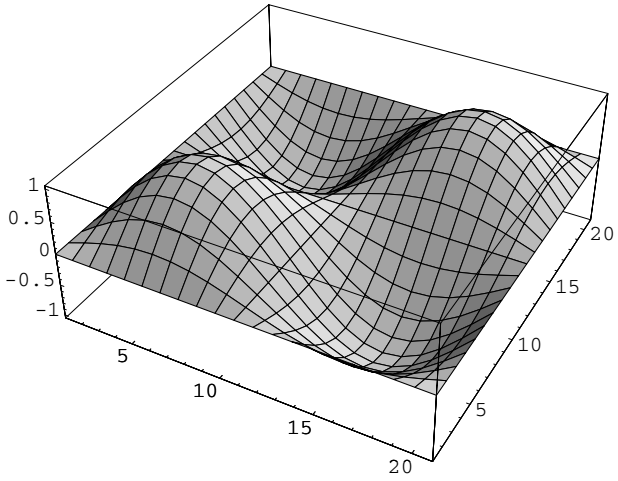

```
Out[57]= -DensityGraphics-
```

The functions **ListPlot3D**, **ListContourPlot**, and **ListDensityPlot** produce similar plots from 2-D matrices or arrays of data. None of these three accept independent values, and the horizontal coordinates are simply row and column numbers. To illustrate, first fill a table with values of sin x sin y over the ranges $0 \le x \le 2\,\pi$ and $0 \le y \le 2\,\pi$, with grid increments of $\pi/10$. The semi-colon is used to suppress the output of the resulting table.

```
In[58]:= Table[Sin[x] Sin[y], {x, 0, 2 π, π/10}, {y, 0, 2 π, π/10}];
```

Then, plot the results using **ListPlot3D**.

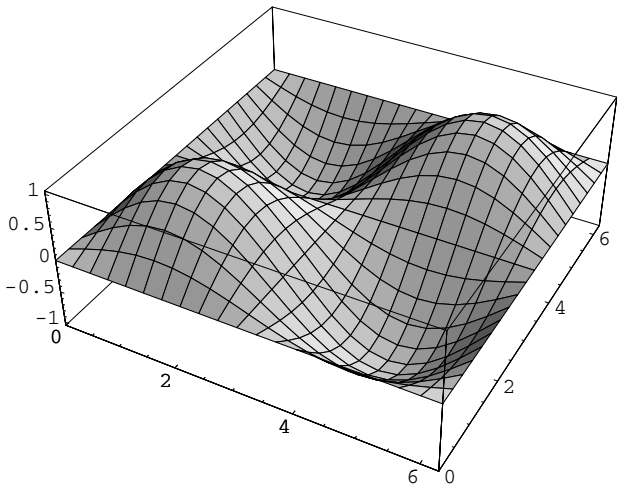
```
In[59]:= ListPlot3D[%, ColorOutput → GrayLevel]
```




```
Out[59]= -SurfaceGraphics-
```

To change the horizontal coordinates from row and column numbers, use the **MeshRange** option.

```
In[60]:= ListPlot3D[%%, ColorOutput → GrayLevel,
           MeshRange → {{0, 2 π}, {0, 2 π}}]
```




```
Out[60]= -SurfaceGraphics-
```

1.5.4 User-Defined Functions

Mathematica allows the definition of functions that can range from one-line statements to complicated programs involving logical operations, calculations, and graphics. A simple example of a function to calculate the square of a numer is:

```
In[61]:= x2[x_] := x^2
```

The combined colon and equal sign, **:=**, delays the assignment of the value $x^2$to **x2** until the function is executed, and is therefore different than **x2** =x^2. Once a function is defined, it can be used just like any of the built-in *Mathematica* functions.

```
In[62]:= x2[9.5]

Out[62]= 90.25
```

An equivalent way to accomplish the same thing is to use the **Function** function

```
In[63]:= Function[x2, x^2]

Out[63]= Function[x2, x^2]

In[64]:= x2[5]

Out[64]= 25
```

or, using *Mathematica* shorthand,

```
In[65]:= (#1^2)&

Out[65]= #1^2&

In[66]:= %[5]

Out[66]= 25
```

The shorthand version can produce very compact programs and is often used by expert *Mathematica* programmers, but can also be very difficult for others to read and understand.

Mathematica contains a variety of functions useful for flow control in longer programs – for example **If**, **Do**, **While**, and **For** – that can be used for traditional procedural programming. It also contains functions such as **Map** and **Apply** that can be used for functional programming. Here are four different ways to calculate the sines of a table of real numbers:

```
In[67]:= values = Table[x, {x, 10.°, 40.°, 10.°}]

Out[67]= {0.174533, 0.349066, 0.523599, 0.698132}

In[68]:= Map[Sin, values]

Out[68]= {0.173648, 0.34202, 0.5, 0.642788}

In[69]:= Sin[values]

Out[69]= {0.173648, 0.34202, 0.5, 0.642788}
```

```
In[70]:= Table[Sin[values[[i]]], {i, Length[values]}]

Out[70]= {0.173648, 0.34202, 0.5, 0.642788}

In[71]:= Do[values[[i]] = Sin[values[[i]]],
          {i, Length[values]}]; values

Out[71]= {0.173648, 0.34202, 0.5, 0.642788}
```

Complicated multi-line programs can be built up using the **Module** function, which allows for local variables and functions to be defined. Here is a simple application of **Module** that takes a list of data, determines its length, calculates its mean and standard deviation, and then returns all three results in a list.

```
In[72]:= Example[data_] :=
           Module[{len, mean, dev},
             len = Length[data];
             mean = 1/len Sum[data[[i]], {i, 1, len}];
             dev = 1/(len - 1) Sqrt[Sum[(data[[i]] - mean)^2, {i, 1, len}]];
             Return[{len, mean, dev}]
           ]
```

Using the previously defined list **data**, the results are:

```
In[73]:= Example[data]

Out[73]= {6, 5.26667, 1.30833}
```

Outside of the module, however, the variables **len**, **mean**, and **dev** have no values.

```
In[74]:= {len, mean, dev}

Out[74]= {len, mean, dev}
```

1.5.5 Data Import and Export

Mathematica can import and export many common kinds of data and image files. The general **Import** function can in most cases recognize file types and make the appropriate conversions. **Import** assumes that any file name ending in .dat contains rows and columns of data. For example, the file example.dat contains four rows each consisting of four columns of data.

```
In[75]:= Import["/Users/bill/Mathematica_Book/example.dat"]

Out[75]= {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}}
```

If **List** is specified as the file format, however, *Mathematica* will treat the data as a single list.

```
In[76]:= Import["/Users/bill/Mathematica_Book/example.dat",
           "List"]

Out[76]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
```

The file path name can be pasted into an **Import** statement by selecting Get File Path... from the Input menu. The same syntax works for graphics files.

```
In[77]:= Import["/Users/bill/Mathematica_Book/pako.jpg"]

Out[77]= -Graphics-
```

Using the syntax above, *Mathematica* will use the file suffix to identify the file format. See the *Mathematica* documentation for information on files without suffixes. Graphics files do not appear until they are specifically shown using the **Show** function.

```
In[78]:= Show[%]
```

```
Out[78]= -Graphics-
```

The **Export** function works similarly to the **Import** function except that both an expression (the data or image to be exported) and a file name must be specified.

1.5.6 *Mathematica* Packages

Mathematica functions and programs can be stored as text files known as packages and loaded when needed. The standard distribution of *Mathematica* includes dozens of packages with special functions for algebra, calculus, graphics, linear algebra, numerical mathematics, and statistics. To see a complete list of the standard packages accompanying *Mathematica*, bring up the Help Browser window, choose Add-ons

& Links in the far left column, then Standard Packages in the middle column. The right column will contain a list of directories, each of which contains several add-on packages that can be loaded whenever they are needed. Additional packages are available from Wolfram Research, from other commercial developers, and in the public domain (generally downloadable from the internet). This book includes a package named CompGeosci, which contains a number of functions for specialized plots and calculations as well as color functions that are useful for color graphics. Users can also write their own packages, although the details of package writing are beyond the scope of this book.

Mathematica packages can be loaded in two ways. The first is to use **<<** *package_name*, which loads the specified package. This is generally not the recommended method because problems can arise if a package is loaded more than once during a *Mathematica* session. The preferred method is to use **Needs[***package_name***]**, which loads parts of the package as needed and will not load part of a package more than once.

Package names can be specified either using their complete file path or, if they are located along one of *Mathematica*'s default file paths, using their directory (context in *Mathematica* terms) and package name. For example, to load the package DescriptiveStatistics from the Statistics directory (context) located along one of the default file paths, enter

```
In[79]:= Needs["Statistics`DescriptiveStatistics`"]
```

Note that the ` character is not a single quotation mark! It is the character located beneath the tilde (~) character in the upper left hand corner of most keyboards.To see a listing of the default file path for the installation of *Mathematica* on your computer,type **$Path** and press Enter.To see a list of the packages that have been loaded during a given *Mathematica* session, type **$Packages** and press Enter.

1.6 References and Recommended Reading

Glynn, J. and Gray, T., 2000, *The Beginner's Guide to Mathematica Version 4*: Cambridge University Press.

Wolfram, S., 1999, *The Mathematica Book (4th ed.)*: Cambridge University Press.

http://www.springer.com/978-3-540-40245-9

Computational Geosciences with Mathematica
Haneberg, W.
2004, XIII, 381 p., Hardcover
ISBN: 978-3-540-40245-9