

2.1 Computational Toolbox—Tools of the Trade:

Mathematica Tutorial 1

File: *MathematicaTutorial1.nb*

Introduction to Computational Science: Modeling and Simulation for the Sciences
Angela B. Shiflet and George W. Shiflet
Wofford College
© 2006 by Princeton University Press

Introduction

This first tutorial on *Mathematica*® gives an introduction to the system and prepares you to use *Mathematica* to complete several projects in the first few chapters. The tutorial introduces the following functions and concepts: Getting started; evaluation; saving; cells; styles; numbers; arithmetic operators; built-in functions, such as *N*, *Log*, *Sin*, *E*, and *Exp*; variables; assignments; user-defined functions; online documentation with the help menu, *?*, and ***; *Print*; looping with *Do*; *Plot*; differentiation; solving differential equations; and optional material on integration. The module also gives examples along with Quick Review Questions for you to do with *Mathematica*.

We recommend that you work through this introduction with a copy of *Mathematica*. *Mathematica* is available from Wolfram Research, Inc. (<http://www.wolfram.com/>).

Getting Started

Mathematica® is a powerful computer algebra system that can plot graphs and perform a large variety of calculations with symbols as well as numbers. A **Mathematica notebook**, such as this one, is like an interactive book with text and the capability of performing calculations.

To open this notebook, you probably double-clicked the *Mathematica* folder to open it and then double-clicked the *Mathematica* icon to start a new notebook or double-clicked this file, *MathematicaTutorial1.nb*.

If you do not see a palette on the right with symbols, such as the square root sign, go to the **File menu**, **Palettes sub-menu**, and select **BasicInput**. (**BasicCalculations** is another useful palette that gives and explains frequently used *Mathematica* functions.)

The material in a notebook is separated into **cells**, which are marked with a right square bracket. Cells have different tasks, such as input, output, and text. In the following material, evaluate each given expression in an input cell, which is in red in this notebook. To evaluate an expression, with the cursor anywhere in its cell press **<ENTER> on the numeric keypad** (not <ENTER> or <RETURN> on the main keyboard). Execute all input cells to view the results of the examples. Input is labeled with **In**, and output is labeled with **Out**. The number in square brackets indicates the calculation order of the expression. We start by evaluating 25 factorial, which is the product of the positive integers from 1 through 25, using the **exclamation mark !** as in mathematics:

25 !

- **Quick Review Question 1** Also, do anything that is asked of you in cells that look like this one, marked as a **Quick Review Question** in boldface. Because such cells are text cells and not input cells, do not type in these cells. Instead, move the cursor down until it changes from being vertical to being a horizontal-I bar. Click and start typing; a new input cell will form. For this question, evaluate 100-factorial.

Save

Save what you have done by going to the **File menu** and choosing **Save** or **Save As...**. The first time you save a new file, make sure you save the file to your disk using a descriptive name.

After you have saved initially, you can just go to the *File* menu and select *Save* or use the short cut indicated under that menu. **SAVE OFTEN**, particularly before you print, do a long calculation, or display a graph.

- **Quick Review Question 2** Save this notebook on your disk. How did you accomplish this task?

Additional Features

We can select text from a cell and **cut**, **copy**, or **paste** it using items from the **Edit menu** or the short cuts indicated on that menu. We can also select a cell by clicking on the bracket on the right of the screen; we can then cut, copy, or paste the cell.

- **Quick Review Question 3** Select this cell and move it above the preceding cell.

To type text before starting a cell, choose the style **Text** or another heading or section style from **Style** in the **Format menu** or from the top left of the notebook. If you have already started typing and wish to change the style, click the cell bracket and choose the style. The default type is **Input**.

■ Quick Review Question 4 Change the style of this cell to text style.

If you are in one cell, to start another cell, move the mouse down until you get a **horizontal I-bar** and click.

To print, select **Print** from the *File* menu or use the indicated short cut.

To quit *Mathematica*, choose **Quit** from the *File* menu or use the indicated short cut.

NOTE: **If you quit and re-open a notebook later, you must re-execute any function definition or variable assignment to use it again.**

Numbers and Arithmetic Operations

Multiplication is indicated by an **asterisk (*)**, a space, or nothing when the expression is not ambiguous. An expression can be raised to a power using the power template on the *BasicInput* palette or using a **caret (^)** as follows:

$$3 \left(\frac{5}{8} - 1.25 \right)^2$$

$$3 * \left(\frac{5}{8} - 1.25 \right) ^ 2$$

- **Quick Review Question 5 Add the fractions (not decimal numbers) one-half and three-fourths. Use the *BasicInput* palette to have each fraction in stack notation with the numerator directly over a line and the denominator. When possible, *Mathematica* does not return a decimal expansion but an exact answer, such as with this fractional answer.**

Built-in functions, such as **Sin** for the sine function, and built-in constants, such as **Pi** representing π , begin with a capital letter. Functions use square brackets instead of parentheses around the argument(s), as in the following expression to compute $5\sin(\pi/3)$:

$$5 * \text{Sin}[\text{Pi}/3]$$

- **Quick Review Question 6** We do not need to type `*` for multiplication in the previous expression. Also, we can use the *BasicInput* palette to write *Pi* as π and a fraction with stack notation, such as $\frac{\pi}{3}$. Retype the expression making these changes.

- **Quick Review Question 7** `Log[x]` is the natural logarithm of x . Evaluate the natural logarithm of 23.4.

We can obtain a decimal expansion of the number using `N` in one of the two following ways.

```
N[3π + 5]
```

```
3π + 5 //N
```

- **Quick Review Question 8** Euler's number, e , is **E** in *Mathematica*. **E^x** and **$Exp[x]$** are the same and represent e^x . Evaluate the number e^2 with a decimal expansion.

Variables and Assignments

We can employ **variables** to store values for future use. We use the same rules for naming variables as we do for other symbols, such as user-defined function names, as follows:

1. Symbols must begin with a letter of the alphabet.
2. Any combination of letters and digits or nothing at all can follow. The name can also contain dollar signs (\$), although we avoid such a practice.
3. *Mathematica* is case sensitive, regarding uppercase and lowercase letters as different. Therefore, the variable names *Exp*, *exp*, and *EXP* are three separate symbols in *Mathematica*.

To avoid using the same name as a built in *Mathematica* symbol, we begin user-defined names with a lowercase letter. A compound name, such as *rateOfChange*, has the first letter of each subsequent word capitalized.

We can **assign** a value of an expression to a variable using the following format with the variable receiving the value always appearing on the left of an **equal sign (=)**:

```
variable = expression
```

For example, the following assignment statement gives the value 5 to the variable *var*:

```
var = 5
```

Mathematica calculates the value of the expression on the right, such as 5, and then assigns the value to the variable on the left, such as *var*. If we subsequently use *var*, *Mathematica* replaces *var* with its value.

■ **Quick Review Question 9**

a. Assign 4.8 to the variable *time* and execute the statement by pressing <ENTER> on the numeric keypad.

■ b. Type *time* and execute.

■ c. Type *time* + 3 and execute.

■ d. Type *time* and execute. Note that execution of Part c did not change the value of *time*.

If we use a variable before it has a value, *Mathematica* uses the symbol itself.

■ **Quick Review Question 10**

a. Type *vel* and execute.

■ b. Type $(7 + \text{vel} / 3)^2$ and execute.

The assignment statement

```
units = 12 * numberOfDozens
```

is not an algebraic formula. An illustration of the difference between an algebraic equality and a *Mathematica* assignment is the following program segment:

```
counter = 5
counter = counter + 1
```

The last statement does **not** say that *counter* is one more than itself. When *Mathematica* encounters the second statement, it looks up the present value of *counter* (5), evaluates the expression *counter* + 1 (6), and stores the result of the computation in the variable on the left, here *counter*. The effect of the statement is to increment *counter* by 1 from 5 to 6. We see later that incrementing by 1 is important when we need to count. In many situations, we need an old value of a variable to compute the new value of the same variable.

■ **Quick Review Question 11** Write a segment to assign 34 to variable *time* and then to add 0.5 to *time*, changing its value.

Above, we do not need to display the value of *counter* after the assignment of 5 to the variable; we know the value of *counter* is 5. To suppress the output from a statement, we follow the statement with a **semicolon** (;), as follows:

```
counter = 5;
```

- **Quick Review Question 12** Repeat the previous question suppressing the output from the initial assignment.

User-defined Functions

Frequently, we wish to define our own functions that we can use again and again. As with variable names, we begin the name of such a **user-defined function** with a lowercase letter.

Suppose we wish to define the function $f(x) = x^2$ in *Mathematica*. In case f has a definition in memory from previous work, we clear any earlier definition of f with **Clear**, as follows:

```
Clear[f]
```

To clear function definitions or variable values for several symbols, we separate the symbols by commas, as follows:

```
Clear[f, x]
```

We define a function with the dummy variable, here x , in **square brackets**, `[]`, followed by an underline. Several lines can appear in the definition. The definition of a function uses `:=` instead of the assignment equal `=`. The **colon-equal sign**, `:=`, is a **delayed assignment**. With its use, the right hand side is not evaluated immediately but is reevaluated each time the function is called. The *Mathematica* definition of the function $f(x) = x^2$ follows:

```
f[x_] := x2
```

■ **Quick Review Question 13** Evaluate the function f at 5. Be sure to use square brackets around 5.

```
f[5]
```

We can use symbols as well as numbers for a function's argument. For example, the following call to the function f uses an undefined symbol, z , in the argument:

```
f[3z]
```

■ Quick Review Question 14

- a. Clear any possible values for p and t .
- b. Define the function $p(t) = 100e^{0.1t}$. Be sure to use t on both sides of the definition and E or Exp for e .
- c. Evaluate p at $t = 12$.

Online Documentation

We can obtain extensive online help under the **Help menu** by selecting **Help Browser...**. We can type the name of a term and click the **Go button** or can manouver through the various menus to the desired help.

To find out information about a symbol, we can also type a **question mark (?)** followed by the symbol, such as the following statement to obtain information on the *Mathematica* function *Log*:

```
? Log
```

An **asterisk *** is a **wildcard**, standing for any string of characters. We can find a list of commands starting with *Log* as follows:

```
? Log*
```

■ Quick Review Question 15

- a. Obtain information about the function p , which you defined in the previous section.

- b. Obtain the list of *Mathematica* symbols that contain "Exp" somewhere in their names.

- c. Using the *Help Browser*, obtain information about *Exp*.

Printing

Sometimes, particularly when doing error checking, we wish to display intermediate results. To do so, we can employ the `Print` function, whose format is as follows:

```
Print[expr1, expr2, ... ]
```

The command displays the arguments ($expr1, expr2, \dots$) with no spaces between them and then advances to a new line. Each argument can be any expression: a string constant, such as "time of day"; variable, such as x ; or a more involved expression, such as $f[5] + 2\text{Sin}[x]$. For a **string constant**, which is the sequence of characters within quotation marks, *Mathematica* prints the characters between the quotation marks. An argument that is a variable does not have quotation marks surrounding it; and if the variable has a value, output contains this value. For a more involved expression argument, *Mathematica* also evaluates the expression and then displays its value. The following segment displays string constant documentation along with the value of a variable, *timeOfExp*:

```
timeOfExp = 55.4;  
Print["Time = ", timeOfExp, " seconds"];
```

- **Quick Review Question 16** Write a statement to assign 3 to t . Then, employ *Print* to display "Velocity is ", the result of the computation $-9.8t$, and " m/sec." For proper spacing, be sure to type blanks in the string constants after "is" and before "m/sec". Output for the *Print* command should be as follows:

Velocity is -29.4 m/sec.

Looping

It is often advantageous to be able to execute a segment of code a number of times. For example, to obtain the velocity for each integer time ranging from 1 to 1000 seconds, it would be inconvenient for the user to have to execute one thousand times a cell assigning a time and computing the corresponding velocity. Some method of automating the procedure is far more preferable. A segment of code that is executed repeatedly is called a **loop**.

Several types of loops exist in *Mathematica*. When we know exactly how many times to execute the loop, the function **Do** is often a good choice for implementing the loop. One form of the function is as follows:

```
Do[expr, {imax}]
```

The body of the loop is *expr*, and *imax* is an integer indicating the number of times to execute *expr*. The function *Do* itself has no return value. For example, suppose, as the basis for a more involved segment, we wish to increment distance by 2.25 for 7 times. We initialize the distance variable, say *dist*, to be 0. Within a *Do* loop that executes 7 times, we calculate the sum of *dist* and 2.25 and assign the result of the expression to *dist*, giving the variable an updated value. Because the *Do* loop does not return a value, we display the final value of *dist* after the loop. The segment is as follows:

```
dist = 0;  
  
Do[  
  dist = dist + 2.25,  
  {7}]  
  
dist
```

- **Quick Review Question 17** Write a segment to assign 1 to a variable d without displaying 1. In a loop that executes 10 times, change the value of d to be double what it was before the previous iteration. After the loop, type d so that *Mathematica* displays d 's final value. Before executing the loop, determine the final value so you can check your work.

While a comma separates the body from the number of times the loop is to execute, semicolons separate statements in the loop body. In the next loop, we increment *dist* by 2.25, compute time as $\frac{24.5 - \sqrt{600.25 - 19.6 \text{ dist}}}{9.8}$, and then display distance and time. As the segment shows, a semicolon appears between each pair of commands:

```

dist = 0;

Do[
  dist = dist + 2.25;
  t =  $\frac{24.5 - \sqrt{600.25 - 19.6 \text{ dist}}}{9.8}$ ;
  Print["For distance = ", dist, " time = ", t, " seconds."],
  {7}]

```

- **Quick Review Question 18** This question is a variation of the previous Quick Review Question. Write a segment to assign 1 to a variable d without displaying 1. In a loop that executes 10 times, change the value of d to be double what it was before the previous iteration, and then print the value of d . Thus, output appears on 10 lines.

Another form of the *Do* loop, which is useful for animations, specifies an **index** or **loop variable**, such as i , that takes on values from 1 through a terminal value, such as $imax$ in the following:

```
Do[expr, {i, imax}]
```

We can also specify a different initial value as follows:

```
Do[expr, {i, imin, imax}]
```

The following example of *Print* and *Do* with an index i displays i and i -factorial ($i!$) with blanks between the values for i going from 1 through 9:

```

Do[
  Print[i, " ", i!],
  {i, 9}]

```

To start the display with the value of $0!$, which is 1, we indicate a beginning value of 0, as follows:

```

Do[
  Print[i, " ", i!],
  {i, 0, 9}]

```

- **Quick Review Question 19** For this question, complete another version of the segment above that displays distance and time. In this version, do not initialize *dist*. Employ a loop with an index *i* that takes on integer values from 1 through 7. Within the loop, the value of *dist* is computed as $2.25i$. After replacing each `xxxxxxxxxx` with the proper code, execute the cell, and compare the results with the similar segment above.

```
Do[
  dist = xxxxxxxxxxxx;
  t =  $\frac{24.5 - \sqrt{600.25 - 19.6 \text{ dist}}}{9.8}$ ;
  Print["For distance = ", dist, " time = ", t, " seconds."],
  xxxxxxxxxxxx]
```

- **Quick Review Question 20**
 - a. Clear possible values for the symbols *g* and *x*.
 - b. Define the function $g(x) = \ln(3x + 2)$. Recall that *Log* is the *Mathematica* function for the natural logarithm.
 - c. Write a loop that prints the value of *k* and $g(k)$ for *k* taking on integer values from 1 through 8.

Plotting

We employ the `Plot` command for graphing two-dimensional functions. The basic form of the command gives the function to graph (*g*), the independent variable (*x*), and the endpoints of the interval on which to graph the function, as follows:

```
Plot[g, {x, min, max}]
```

For example, suppose $f(t)$ is defined as t^2 . The following command graphs f with t varying from -1 to 2 :

```
Plot[f[t], {t, -1, 2}]
```

Notice that we use $f[t]$, not just f ; and the independent variable (t) along with the endpoints of the interval to plot appear in braces. Instead of employing the name of the function, such as $f[t]$, we can obtain the same results by using its definition, as follows:

```
Plot[t2, {t, -1, 2}]
```

The graph is actually a side effect, and "-Graphics-" appears an *Out* line following the plot. As we do to suppress other output, we follow the *Plot* command with a semicolon to eliminate the *Out* line but not the graph.

- **Quick Review Question 21** Graph $e^{\sin(x)}$ from -3 to 3. Do not define a function for this expression but place the expression in the *Plot* command.

After braces containing the interval and a comma in the *Plot* command, an *AxesLabel* option generates axes labels, which should appear in all scientific graphics. Following *AxesLabel*, the option has a hyphen (-) with a greater-than symbol (>) and braces containing the labels, which usually appear in quotation marks. The command that follows plots the graph and labels the horizontal and vertical axes with t and y , respectively:

```
Plot[f[t], {t, -1, 2},
  AxesLabel -> {"t", "y"}
];
```

- **Quick Review Question 22** Adjust the answer to the previous Quick Review Question to label the x and y axes and to eliminate output of "-Graphics-".

To plot several functions on the same graph using *Plot*, separate the functions by commas and place them in braces, such as follows:

```
Plot[{f[t], 3 t + 4}, {t, -1, 2}];
```

- **Quick Review Question 23** Adjust the answer to the previous Quick Review Question to plot $e^{\sin(x)}$ and $\sin(x)$ on the same graph.

In future *Mathematica* tutorials, we consider plotting more than one function and other plotting options, but the basic *Plot* command with the *AxesLabel* option supports much modeling work.

Differentiation

See Module 2.3 on "Rate of Change" for a discussion of differentiation.

Paralleling calculus, *Mathematica* provides the **prime notation** for taking the derivative. For example, consider the following function:

$$y = s(t) = -4.9t^2 + 15t + 11$$

We define the function in *Mathematica*, as follows:

```
Clear[s, t];  
s[t_] := -4.9 t^2 + 15 t + 11
```

The following prime notation returns the derivative function, $15 - 4.9t$:

```
s'[t]
```

To evaluate the derivative at 1, which is $s'(1) = 5.20$, we call this derivative function at 1:

```
s'[1]
```

Alternatively, to avoid first defining a function, we can employ the **D** command in *Mathematica* to compute the derivative. The first argument is an expression for which we want the derivative, and the second is the independent variable. Thus, to take the derivative of $-4.9t^2 + 15t + 11$ with respect to t , we can employ the following command:

```
D[-4.9 t^2 + 15 t + 11, t]
```

- **Quick Review Question 24** Give *Mathematica* code to do the following:
- Define the function $f(x) = 2.9 \sin(0.03x)$.
 - Evaluate the derivative of f at 35.
 - Define a function $d(x) = f'(x)$ using the prime notation.
 - Evaluate the derivative of $2.9 \sin(0.03x)$ using the D notation.

Solving Differential Equations

See Module 2.3 on "Rate of Change" for a discussion of differential equations.

Representing differential equations in *Mathematica* requires that we be able to represent equalities. To avoid confusion with the assignment equal sign ($=$), *Mathematica* uses a **double-equal sign** ($==$) for equality in an equation. Thus, we write the mathematical equation $y = 5x + 3$ as $y == 5x + 3$ in *Mathematica*.

We can employ the *Mathematica* function **DSolve** to solve a differential equation or a system of differential equations. For example, suppose we wish to solve the differential equation $F'(t) = -t^2 + 10t + 24$ with initial condition $F(0) = 30$. With braces surrounding these equations and double-equal signs to express the equality, we also indicate that we want to solve for $F(t)$ and t is the independent variable, as follows:

```
DSolve[{F'[t] == -t2 + 10 t + 24, F[0] == 30}, F[t], t]
```

- **Quick Review Question 25** Write a *Mathematica* command to solve the differential equation $P'(t) = 0.1P(t)$ with initial condition $P(0) = 100$. The module on "Unconstrained Growth" uses this differential equation.

Integration

See Module 2.4 on "Fundamental Concepts of Integral Calculus" for a discussion of integration.

As indicated with the prime notation for the derivative, input to *Mathematica* can be quite natural. Using the *BasicInput* palette, we can request that *Mathematica* evaluate definite integrals and indefinite integrals using the notation of mathematics, such as with the following examples:

$$\int (-t^2 + 10t + 24) dt$$

$$\int_0^5 (-t^2 + 10t + 24) dt$$

- **Quick Review Question 26**
- Plot $\sin^2(x)$ from 0 to 2π .
- b. Obtain the indefinite integral of $\sin^2(x)$.
- c. Obtain the definite integral of $\sin^2(x)$ from 0 to 2π .