2.1 Computational Toolbox—Tools of the Trade: *Maple* Tutorial 1

File: *MapleTutorial1.mw*

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 *Maple*® gives an introduction to the system and prepares you to use *Maple* to complete several projects in the first few chapters. The tutorial introduces the following functions and concepts: getting started; evaluation; saving; execution groups; styles; numbers; arithmetic operators; built-in functions, such as *evalf*, *ln*, *sin*, and *exp*; variables; assignments; user-defined functions; online documentation with ?; printing; looping with *for*; plotting; differentiation; solving differential equations; and optional material on integration. The module also gives examples along with Quick Review Questions for you to do with *Maple*.

We recommend that you work through this introduction with a copy of *Maple*, which is available from Waterloo Maple, Inc. (http://www.maplesoft.com/).

Getting Started

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

To open this worksheet, you probably double-clicked the *Maple* folder to open it and then double-clicked the *Maple* icon to start a new notebook or double-clicked this file, *MapleTutorial1.mw* .

Most of this worksheet is organized into **sections**. Each section begins with a heading and a **plus** (+) or **minus** (-) **icon** to the left of the heading. If the icon displays a plus, click the icon to open the section and reveal its contents. Similarly, clicking a minus icon, closes the section.

If you do not see a palette with symbols, such as the square root sign, go to the ***View menu***, ***Palettes submenu***, and select ***Expression***.

The material in a worksheet is separated into **execution groups**, which are marked with a left square bracket. Execution groups have different tasks, such as input, output, and text. In the following material, evaluate each given expression in an input execution groups, which is in red and starts with the **greater than prompt** (>) in this worksheet. To evaluate an expression, with the cursor anywhere in its execution group press **<ENTER> or <RETURN>** . Execute all input execution groups to view the results of the examples. The resulting output execution group is in blue. After an execution, *Maple* jumps to the next execution group, so that you may need to scroll back to read text between groups.

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 and terminating the command with a **semicolon (;)**:

```
> 25!;
```

Quick Review Question 1 Do anything that is asked of you in execution groups that look like this

one, marked as a Quick Review Question in boldface. Because such execution groups are text and not input style, do not type in these execution groups. Instead, if a greater than prompt does not appear in an otherwise empty execution group, from the *Insert* menu, *Execution Group* submenu, select *After Cursor*. Alternatively, use the shortcut indicated on that menu or click the icon (|>) to "Insert executable *Maple* input after the current paragraph". For this question, evaluate 100-factorial. Remember to end the command with a semicolon.

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.

Additional Features

We can select text from a execution group and **cut**, **copy**, or **paste** it using items from the ***Edit menu*** or the short cuts indicated on that menu. To delete an execution group, with the cursor in the group, from the ***Edit*** menu, select ***Delete Element*****,** or use the indicated shortcut.

We can insert a new input execution group before or after the current one from the ***Insert* menu**, ***Execution Group* submenu**, by selecting ***Before Cursor*** or ***After Cursor*****,** respectively, or by using the indicated shortcut. We can also employ the icon (**|>**) to "Insert executable *Maple* input after the current paragraph."

Quick Review Question 3

a. Copy the text in the execution group that contains your answer for your answer to Quick Review Question 1. Place the cursor on Quick Review Question 3, insert an execution group above this one, and paste in the copied material.

b. Delete Quick Review Question 2 in the "Save" section above.

To type text before starting a execution group, choose the style ***Text*** or another heading or section style from ***Styles...*** in the ***Format menu*** or from the drop-down list at the top left of the worksheet. Alternatively, click the **text icon** (**T**) to "Insert text after the current paragraph." If you have already started typing and wish to change the style, select the contents of the execution group and choose the style. The default type is ***Maple Input*****.**

Frequently, for readability in *Maple* input, a command should appear on more than one line. However, if we press <ENTER> or <RETURN> before a semicolon, *Maple* attempts to execute the command and displays an error or warning message about the missing semicolon. Besides having one command on two

lines, often several commands are appropriate to have together in one execution group. Although we can place several commands on one line, it is usually much clearer to type the commands on separate lines. However, if we end one command with a semicolon and press <ENTER> or <RETURN>, *Maple* attempts to execute the first command before allowing us to type another. To overcome these difficulties and **to continue to the next line** without executing, we press **<SHIFT> + <ENTER>** or **<SHIFT> + < RETURN>**.

Quick Review Question 4

a. Change the style of this execution group to text style.

b. After 9299.446 in the following *Maple* input, continue the command to the next line, adding in two more numbers, and terminating the command with a semicolon before executing.

```
> 23.5 + 34.2 + 123.8 + 9299.446
```

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

To quit *Maple*, 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 (*)**. An expression can be raised to a power using the power template on the *Expression* palette or using a **caret (^)** as follows:

```
> 3*(5/8 - 1.25)^2;
```

Quick Review Question 5 Add the fractions (not decimal numbers) one-half and three-fourths. Use the division operator (/) in each fraction. When possible, *Maple* does not return a decimal expansion but an exact answer, such as with this fractional answer.

Maple has many built-in functions, such as the sine function ***sin***, and built-in constants, such as ***Pi*** or ***pi*** representing π. As with mathematical notation, functions use parentheses around the argument(s), as in the following expression to compute 5 $sin(\pi/3)$:

```
> 5 * sin(Pi/3);
```

Quick Review Question 6 We can use the *Symbol* palette from the *View* menu, *Palette* submenu for π and the *Expression* palette for the arithmetic operators, such as for multiplication and division, and for several functions, such as *sin*. *Maple* displays a %? in place of any value we are to replace. We can tab between %? symbols, typing in desired values. Retype the expression 5 $sin(\frac{\pi}{3})$ using the *Symbol* and *Expression* palettes where possible.

Quick Review Question 7 The natural logarithm of *x* is written as *ln(x)* in *Maple* and mathematical notation. Evaluate the natural logarithm of 23.4.

We can obtain a decimal expansion of the number using the function ***evalf***, as in the following example:

```
> evalf(5 + 3*Pi);
```

Quick Review Question 8 Euler's number to the *x* power, e^x, is *exp*(*x*) in *Maple*. 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, digits, underscores (_), or nothing at all can follow.
3. *Maple* is case sensitive, regarding uppercase and lowercase letters as different. Therefore, the names *Exp*, *exp*, and *EXP* are three separate symbols in *Maple*.
4. The name cannot be a *Maple* reserved word, such as *exp*.

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 **assignment operator (:=)** , a colon followed by an equal sign:

variable := expression

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

```
> var := 5;
```

Maple 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*, *Maple* replaces *var* with its value.

Quick Review Question 9

a. Assign 4.8 to the variable *myTime* and execute the statement by pressing <RETURN> or <ENTER>.

b. Type *myTime*, a semicolon, and execute.

c. Type *myTime* + 3, a semicolon, and execute.

d. Type *myTime*, a semicolon, 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, *Maple* uses the symbol itself.

Quick Review Question 10

a. Type *x*, a semicolon, and execute.

b. Type $\left(7+\frac{vel}{3}\right)^2$ in *Maple* input notation and execute.

The assignment statement

```
> units := 12 * numberOfDozens;
```

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

```
> counter := 5;
  counter := counter + 1;
```

The last statement does **not** say that *counter* is one more than itself. When *Maple* 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 *myTime* and then to add 0.5 to *myTime*, 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 following the statement with a **colon (:)**, as follows:

```
> counter := 5 :
```

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

User-defined Functions

Frequently, we need to define our own functions that we can use again and again. Suppose we wish to define the function $f(x) = x^2$ in *Maple*. In case *f* has a definition in memory from previous work, we clear any earlier definition of *f* with ***unassign***, as follows:

```
> unassign(f);
```

or with assignment of 'f' to *f*, as follows:

```
> f := 'f';
```

We remove values of variables using this latter method, such as follows:

```
> x := 'x';
```

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

```
> unassign(f, g);
```

To remove definitions and values of all functions and variables, we can employ the ***restart*** command. In the following segment, we give a value to *x*, *restart*, and print *x* again to show that it no longer contains a

value:

```
> x := 3:
  print("x before restart is", x);
  restart;
  print("x after restart is", x);
```

The definition of a function uses an **assignment equal :=** after the function name, here *f*. Then, the parameter, such as *x*, appears followed by the **arrow operator (->)**, which is a hyphen and a greater-than symbol. The *Maple* definition of the function $f(x) = x^2$ follows:

```
> f := x -> (x^2);
```

Quick Review Question 13 After executing the previous execution group, evaluate the function *f* at 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(5 * z);
```

This call to *f* returns the square of 5 *z*, which is $25z^2$.

Functions can have more than one parameter, too, such as function of *x* and *y* that returns the sum of their squares. In the definition, we place the parameters in parentheses before the arrow, as follows:

```
> sumSquares := (x, y) -> x^2 + y^2;
```

To call the function *sumSquares*, we use the function name and arguments in parentheses, as in the following example:

```
> sumSquares(3, 4);
```

Quick Review Question 14

a. Clear possible values for *x*, *pop*, *initPop*, *r*, and *t*.

b. Define the function $pop(t) = 100\, e^{(0.1\, t)}$. Be sure to use *t* on both sides of the definition and *exp* for the exponential function.

c. Evaluate *pop* at *t* = 12.

d. Define a function $pop(initPop, r, t) = initPop\, e^{rt}$.

e. Call the function for *initPop* = 1000, *r* = 0.15, and *t* = 12.

Online Documentation

To find out information about a symbol, we type a **question mark (?)** followed by the symbol and an optional semicolon, such as the following statement to obtain information on the *Maple* function *log*:

```
> ?log
```

If the beginning of several terms match the symbol, *Maple* allows us to pick from a list of matching topics. We can find a list of commands starting with *lo* as follows:

```
> ?lo
```

We can also obtain online help through the ***Help* menu** or by clicking the ***Help*** (**?**) icon.

Quick Review Question 15

a. Display the definition of the function *p*, which you defined in the previous section, by entering *p*(*x*).

b. Obtain the list of *Maple* symbols that begin with "ex".

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*, ...) 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*sin(*x*). For a **string constant**, which is a sequence of characters within quotation marks, *Maple* prints the characters and 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, *Maple* 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 In one execution group, write a statement to assign 3 to *t* without displaying *t*, and employ print to display "Velocity is ", the result of the computation -9.8 *t*, and " m/sec." Be sure to use * for multiplication. 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 an execution group 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**.

Definition **loop** A loop is a segment of code that is executed repeatedly.

Several types of loops exist in *Maple*. When we know exactly how many times to execute the loop, the ***for loop*** is often a good choice for implementing the loop. One form of the command, which follows, does not even employ "for":

```
to imax do
    expr
end do
```

The body of the loop is expr, and imax is an integer indicating the number of times to execute expr. The loop itself has no return value. However, as with other commands, if we end the command (after "end do") with a semicolon, *Maple* displays the results of the commands in the body of the loop. If we terminate with a colon, *Maple* does not display these results.

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 *for* 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. To display all intermediate values of dist, we end the loop and the assignment statement in the body of the loop with semicolons, as follows:

```
> dist := 0:

  to 7 do
     dist := dist + 2.25 ;
  end do;
```

We may not want to display all the intermediate values of *dist*. To suppress this display, we end the assignment statement in the body of the loop or the loop itself or both with a colon. In this case, no output occurs. To display the final value of *dist*, we have a command with *dist* (and a semicolon) after the loop.

```
> dist := 0:

  to 7 do
    dist := dist + 2.25:
  end do:

  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 *Maple* displays *d*'s final value. Have no other displays. Before executing the loop, determine the final value so you can check your work.

Frequently, a loop contains more than one statement. In the next loop, we increment *dist* by 2.25, compute time as $\frac{24.5 - \sqrt{600.25 - 19.6\ dist}}{9.8}$, and then display distance and time. We suppress all displays by following commands with colons because execution of the *print* command prints the desired output.

```
> dist := 0:

  to 7 do
    dist := dist + 2.25:
    t := (24.5 - sqrt(600.25 - 19.6 * dist)) / 9.8:
```

```
    print("For distance = ", dist, " time = ", t, " seconds."):
  end do:
```

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 display the value of *d* - 1 without changing *d*'s value. Thus, output appears on 10 lines.**

Another form of the *for* 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:

```
for i to imax do
   expr
end do
```

We can also specify a different initial value as follows:

```
for i from imin to imax do
   expr
end do
```

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

```
> for i to 9 do
     print(i, i!);
  end do;
```

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

```
> for i from 0 to 9 do
     print(i, i!);
  end do;
```

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 2.25*i*. After replacing each xxxxxxxxxx with the proper code, execute the segment, and compare the results with the similar segment above.**

```
> for xxxxxxxxxx
    dist := xxxxxxxxxx
    t := (24.5 - sqrt(600.25 - 19.6 * dist)) / 9.8:
    print("For distance = ", dist, " time = ", t, " seconds.");
  xxxxxxxxxx
```

Quick Review Question 20

a. Clear possible values for the symbols *s* and *x*.

b. Define the function $g(x) = \ln(3x + 2)$.

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 = xmin..xmax);
```

For example, suppose $f(t)$ is defined as t^2:

```
> restart;
  f := t->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 interval to plot appears after an equal sign (=), not an assignment equal. Instead of employing the name of the function, such as in $f(t)$, we can obtain the same results by using its definition, as follows:

```
> plot(t^2, t = -1..2);
```

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 the interval and a comma in the plot command, a ***labels*** option generates axes labels, which should appear in all scientific graphics. Following *labels*, the option has square brackets containing the labels, which appear in quotation marks. The command that follows plots the graph and labels the horizontal and vertical axes with t and y, respectively:

```
> plot(t^2, t = -1..2,  labels = ["t", "y"] );
```

Quick Review Question 22 Adjust the answer to the previous Quick Review Question to label the x- and y-axes.

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..3);
```

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 *Maple* tutorials, we consider other plotting options, but the basic plot command with the label option supports much modeling work.

Differentiation

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

Maple provides several ways to take the derivative. For example, consider the following function:

```
> restart;
  s := t -> -4.9*t^2 + 15*t + 11;
```

The following command with function ***D*** returns the derivative of the function s with respect to its independent variable, t:

```
> D(s);
```

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

```
> D(s)(1);
```

We can display the derivative function by using the independent variable in parentheses, as follows:

```
> D(s)(t);
```

Quick Review Question 24 Give *Maple* code to do the following:

a. Define the function $f(x) = 2.9 \sin(0.03x)$.

b. Evaluate the derivative of f at 35.

c. Define a function $d(x) = f'(x)$.

d. 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 *Maple* requires that we be able to represent equalities. To avoid confusion with the assignment equal sign (:=), *Maple* uses an **equal sign** (=) for equality in an equation.

We can employ the *Maple* 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 equal signs to express the equalities, we also indicate that we want to solve for $F(t)$ using another set of braces, as follows:

```
> dsolve({D(F)(t) = -t^2 + 10*t +24, F(0) = 30 }, {F(t)});
```

Quick Review Question 25 Write a *Maple* 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.

The *Maple* function ***int*** returns the indefinite and definite integrals of a function. The following command evaluates the indefinite integral of $-t^2 + 10\,t + 24$ with respect to *t:*

```
> int(-t^2 + 10*t +24 , t);
```

To evaluate the definite integral of this function from 1 to 5, we employ the same range notation as in the *plot* command.

```
> int(-t^2 + 10*t +24,  t = 1..5);
```

Using the *Expression* palette, we can request that *Maple* evaluate definite integrals and indefinite integrals using the notation of mathematics.

Quick Review Question 26

a. Plot (sin(x))^2 from 0 to 2π.

b. Obtain the indefinite integral of the function from Part a.

c. Obtain the definite integral from 0 to 2π of the function from Part a.