Part I

Introduction

1

Mathematica® Environment and Basic Syntax

1.1 Introduction

Mathematica is a programming language that integrates, through its notebook interface, symbolic and numerical computations, visualization, documentation, and dynamic interactivity. It provides access to a large collection of such diverse and continually updated and expanded data sets as geometric shapes, a searchable dictionary, and individual country attributes. It also permits one to simultaneously program with different programming paradigms, such as procedural, functional, rule-based, and pattern-based. Its interface has a real-time input semantics evaluator that uses styling and coloring to provide immediate visual feedback on such coding aspects as function names, variable selection, and argument structures. Many of the Mathematica functions used for computation and visualization contain a fair amount of high-level automation so that the user has to interact minimally with their inner workings. If desired, many aspects of the automation procedures can be bypassed and specific choices can be selected.

In this book, we shall employ a subset of Mathematica's library of functions and use them to obtain solutions to a variety of engineering applications. It will be found as one becomes more confident with Mathematica that it is most effectively used interactively. In later chapters, emphasis will be placed on displaying the results as dynamically interactive graphical displays so that real-time parametric investigations can be performed.

In this chapter, we shall introduce the fundamental syntax of Mathematica. In Chapters 2 to 7, we shall introduce additional syntax and illustrate its usage. We start by stating that all variables by default are symbols and global in nature, and unless specifically restricted or cleared, are always available in all open notebooks until Mathematica is closed. Also, because Mathematica treats all variables initially as symbolic entities, any undefined symbol appearing in an expression (that is, any variable appearing on the right-hand side of an equal sign) is perfectly acceptable and will not produce an error message. However, depending on how the expression is used, subsequent operations may not perform as expected depending on the intent for this variable.

An Engineer's Guide to Mathematica®, First Edition. Edward B. Magrab.

Companion Website: www.wiley.com/go/magrab

In addition to the functions that are an integral part of Mathematica, each version of Mathematica comes with what are called standard extra packages that provide specific additional functionality. Frequently, the capabilities of these packages become an integral part of Mathematica. What the names of these packages are and a brief description of what they do can be obtained by entering *Standard Extra Packages* into the search area of the *Documentation Center Window*, which is found in the *Help* menu. Each package is loaded by using the **`Needs`** function. One such case is illustrated in Example 4.11.

1.2 Selecting Notebook Characteristics

Interaction with Mathematica occurs through its notebook interface. As we shall be concerned primarily with presenting graphically solutions to engineering analyses, our discussion will be directed to one type of use of the notebook: entering, manipulating, and numerically evaluating equations typically encountered in engineering.

Upon opening Mathematica, the window shown in Figure 1.1 appears on the computer screen. Since virtually all types of mathematical symbols can appear in Mathematica expressions, it is beneficial to also have its *Special Characters* palette open. As indicated in Figure 1.2, the letters and symbols are accessed by selecting *Palettes* from the Mathematica menu strip and then choosing *Special Characters*. These operations produce the windows shown in Figure 1.2.

To increase or decrease the font size of the characters displayed in the notebook, *Window* from the Mathematica menu strip is selected, then *Magnification* is chosen, and the amount of magnification (or reduction) is clicked. These operations are illustrated in Figure 1.3. As shall be discussed in what follows, various types of expression delimiters are used in constructing expressions: parentheses, brackets, and braces. When nested expressions are employed and various combinations of these delimiters are used, one frequently needs to verify that these delimiters are grouped as intended. A tool that performs this check by highlighting the region that appears between the delimiter selected and its closing delimiter is accessed from the *Edit* menu and then by clicking on *Check Balance*, as shown in Figure 1.4. In Mathematica 9, the placement of the cursor adjacent to either an opening or closing delimiter will highlight them in green. This is a very valuable editing tool; however, it can be disabled by going to *Preferences* in the *Mathematica* menu strip, selecting *Interface*, and then deselecting *Enable dynamic*

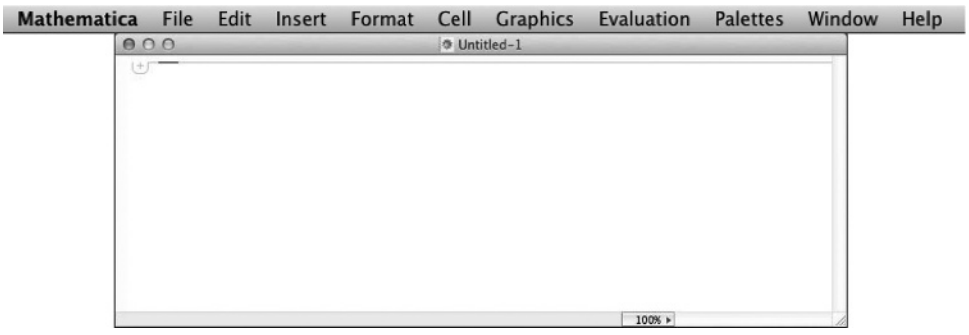
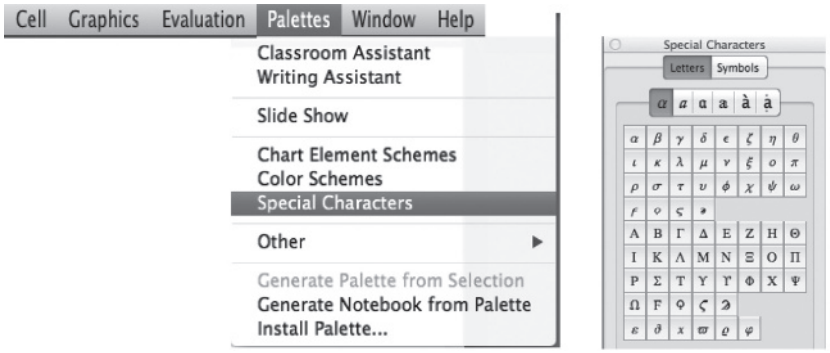
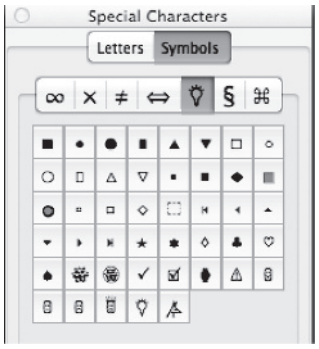


Figure 1.1 Window appearing upon opening Mathematica



(a)



(b)

Figure 1.2 (a) Opening the *Special Characters* window to select various alphabet symbols; (b) Accessing various types of symbols; shown here are shapes that can be used as plot markers

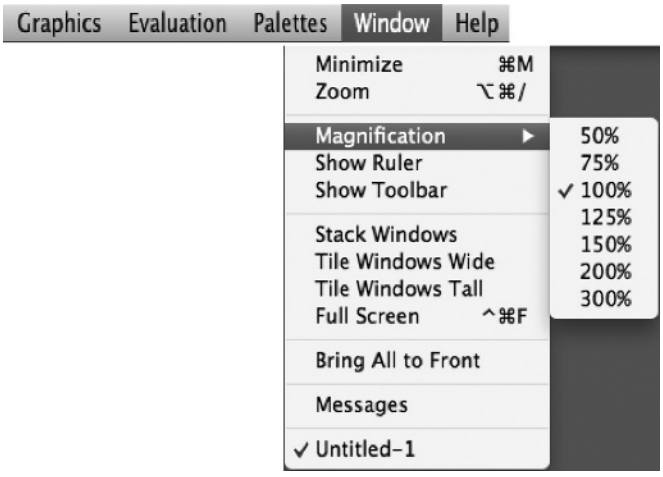


Figure 1.3 Setting the notebook font size

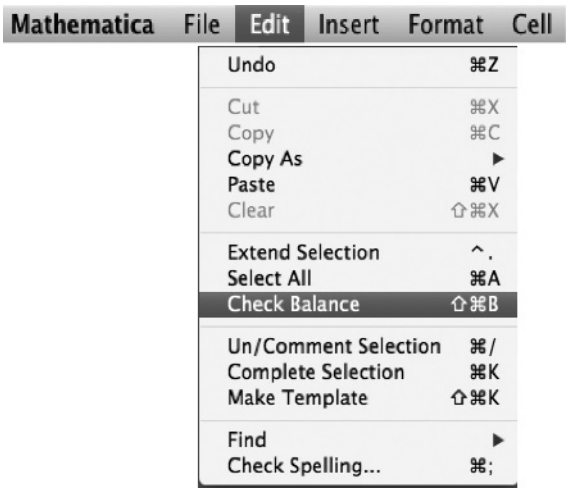


Figure 1.4 Selecting *Check Balance* for implementation of delimiter region identification for (…), […], and {…}

highlighting. Just below *Check Balance* is another useful tool: *Un/Comment Selection*. This feature comments out text highlighted or removes the comment symbols if the selected text had been commented out. The commenting is produced by the system by placing the highlighted text between the asterisks of the set (*…*). (See also Table 1.2.)

Since Mathematica has such a large selection of functions to choose from and since the arguments and their individual form and purpose vary, one should keep the *Documentation Center* window and/or the *Function Navigator* window open for easy access to descriptions of these functions. The *Documentation Center* window is accessed by selecting *Help* from the Mathematica menu strip and then selecting *Documentation Center*. The *Function Navigator* is accessed either by selecting *Function Navigator* from this same menu or by selecting the fourth icon from the left at the top of the *Documentation Center*'s menu strip, which is labeled F[…]. Performing these operations, the windows shown in Figure 1.5 are obtained. Entering either the function name or several descriptive words in the *Documentation Center* search entry area will bring up the appropriate information. In the *Function Navigator*, one will see the candidate functions by selecting the appropriate topic. Using the search function in the *Function Navigator* is the same as using the search function in the *Documentation Center* window; that is, the results appear in the *Documentation Center* window.

After some proficiency has been attained with Mathematica, one can also access the types of functions available for certain tasks and what their arguments are from the *Basic Math Assistant*. The *Basic Math Assistant* is accessed from the *Palettes* menu as shown in Figure 1.6. Visiting the region labeled *Basic Commands*, one can find what arguments are required for many commonly used Mathematica functions. The functions are grouped into seven areas as indicated by the seven tabs. The two rightmost tabs refer to plotting commands. There are two other programming aids that have been added in Mathematica 9. They are the *Next Computation Suggestions Bar* and the *Context-Sensitive Input Assistant*; these are discussed in Section 1.3.

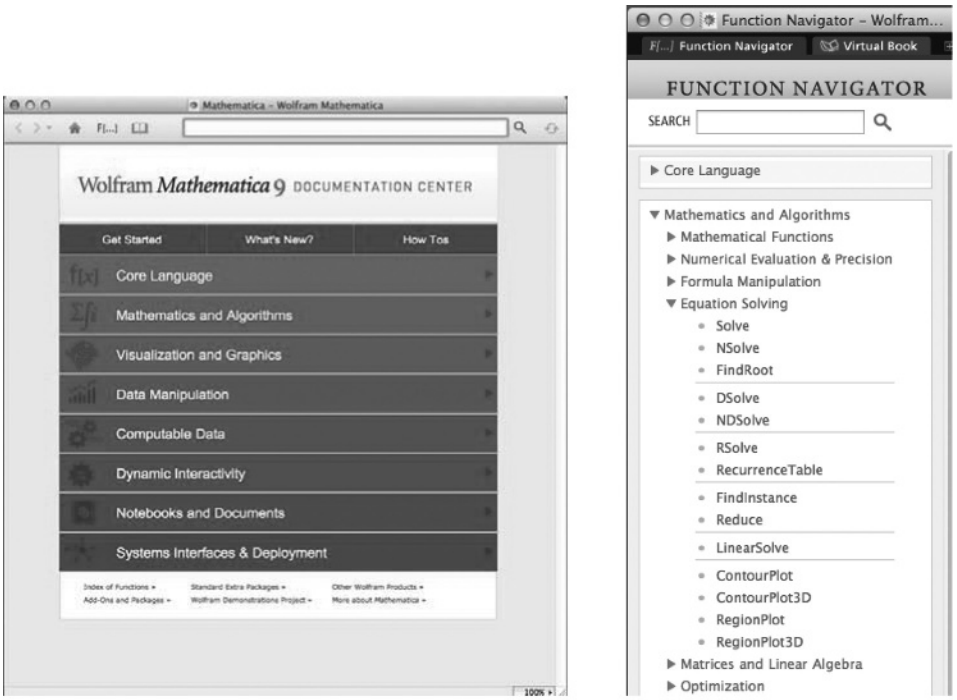


Figure 1.5 (a) *Documentation Center* window and (b) *Function Navigator* window

The *Documentation Center* window also provides access to tutorials on various topics concerning the usage of classes of functions and also has a page that summarizes a collection of functions that can be applied to solve specific topics. Listed in Table 1.1 are selected search entries that can be used as a starting point in determining what is available in Mathematica for obtaining solutions to a particular topic or class of problems. In addition, entering *tutorial/VirtualBookOverview* in the *Documentation Center* search box provides a table of contents to a "how to" introduction to the Mathematica language and contains a very large number of examples illustrating the options available for a specific function.

Lastly, the appearance of the code and the numerical results displayed in the notebook can be altered by selecting *Preferences* in the *Edit* menu. In the *Preferences* window, the *Appearance* tab is chosen and the appropriate tab is selected. For example, the default value of the number of decimal digits to be displayed is 6. To change this value, one goes to the *Numbers* tab and then to the *Formatting* tab. In the box associated with *Displayed precision*, the desired integer value it entered.

Creating New Notebooks or Opening Existing Notebooks

To create a new notebook, one clicks on *File* on the Mathematica menu strip and selects *New* and then *Notebook*. A new notebook window will appear. To open an existing notebook, one clicks on *File* on the Mathematica menu strip and selects *Open* or *Open Recent*. Selecting

Figure 1.6 Opening the *Basic Math Assistant* window to access the 2D palette of plotting commands

Open will bring up a file directory window, whereas *Open Recent* will bring up a short list of the most recently used notebooks.

Saving Notebooks

To save a notebook that was created during a Mathematica session, one clicks on *File* on the Mathematica menu strip and selects *Save As*… . This brings up a file directory from which an appropriate directory is selected and a notebook name is entered. This procedure is also used for renaming an existing notebook. For an existing notebook that has been modified and the existing notebook name is to remain the same, one clicks on *File* on the Mathematica menu strip and selects *Save*.

1.3 Notebook Cells

To execute an expression or a series of expressions, one has two ways to do it. To execute each expression separately, one types the expression and then simultaneously depresses *Shift*

Table 1.1 Selected topical search entries for the *Documentation Center*

Topic	Search Entry
Trigonometric and inverse trigonometric functions	guide/TrigonometricFunctions
Hyperbolic and inverse hyperbolic functions	guide/HyperbolicFunctions
Special functions	guide/SpecialFunctions
Statistics	guide/DescriptiveStatistics guide/FunctionsUsedInStatistics
Minimum, maximum, optimization, curve fitting, least squares	guide/Optimization
Differentiation and integration	guide/Calculus tutorial/Differentiation tutorial/Integration
Differential equations, roots of polynomials, and roots of transcendental functions	guide/DifferentialEquations guide/EquationSolving tutorial/SolvingEquations tutorial/DSolveOverview
Matrices, vectors, and linear algebra	guide/MatricesAndLinearAlgebra
Fourier and Laplace transforms	guide/IntegralTransforms
Interactive graphical output	Manipulate guide/DynamicVisualization tutorial/IntroductionToManipulate
Lists	guide/ListManipulation
Plotting: 2D and 3D	guide/VisualizationAndGraphicsOverview guide/FunctionVisualization guide/DataVisualization guide/DynamicVisualization guide/PlottingOptions guide/Legends guide/Gauges
Listing of all Mathematica functions	guide/AlphabeticalListing (or click on the *Index of Functions* label at the bottom left of the *Documentation Center* window)
Mathematica's syntax	guide/Syntax
Function creation	tutorial/DefiningFunctions
Program debugging and speed	guide/TuningAndDebugging
Manipulation of symbolic expressions	tutorial/PuttingExpressionsIntoDifferentForms
Controls	guide/ControlSystems
Signal processing	guide/SignalProcessing
Units and units conversion	tutorial/UnitsOverview
Export graphics	tutorial/ExportingGraphicsAndSounds

and *Enter*. The system response appears directly below. When one wants to execute a series of expressions after all the expressions have been entered, each expression is typed on a separate line and after each expression has been typed it is followed by *Enter*. When the collection of expressions is to be executed, the last expression entered is followed by simultaneously depressing *Shift* and *Enter*. Each expression in this group of expressions is executed in the order that they appear and the results from each expression (if not followed by a semicolon) appear directly after the last expression entered.

In the first case, the single expression constitutes an individual cell and is so indicated by a closing bracket that appears at the rightmost edge of the notebook window. The system response also appears in its own cell. However, these two individual cells are part of another cell that is composed of these two individual cells. This is illustrated in Figure 1.7a. In the process of obtaining these cells, Mathematica provided two programming aids automatically. The first is the *Context-Sensitive Input Assistant*, which appeared after the first two letters of **`Sin`** were typed. As shown in Figure 1.7b, a short list of common Mathematica commands appears that can be expanded to all appropriate Mathematica commands that begin with **`Si`** by clicking on the double downward facing arrows. Additional information regarding the *Context-Sensitive Input Assistant* can be found in the *Documentation Center* using the entry *tutorial/UsingTheInputAssistant*.

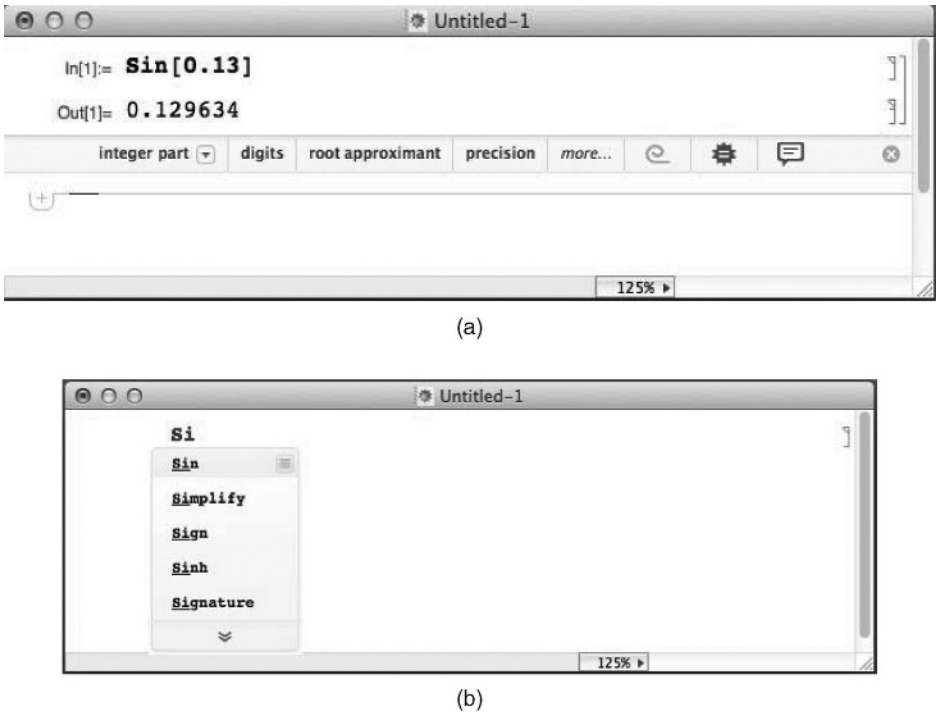


Figure 1.7 (a) Cell delimiters, which appear on the right-hand edge of the window and the *Next Computation Suggestions Bar*; (b) the *Context-Sensitive Input Assistant*, which appeared after the two letters **`Si`** were typed

Note: The *Context-Sensitive Input Assistant* can be disabled by selecting *Preferences* in the *Mathematica* menu. In the *Preferences* window, the *Interface* tab is chosen and then the check mark adjacent to *Enable autocompletion with a popup …* is removed.

After the execution of a line of code and the display of the result, there appears on a separate line a system-provided set of choices. This line is called the *Next Computation Suggestions Bar*. It can be suppressed for this calculation by clicking on the encircled × that appears at its right edge. The *Next Computation Suggestions Bar* remains suppressed for all subsequent program executions until reactivated; that is, until one clicks on the arrow at the right end of one of the results. It is context dependent, and in this case the system suggests to the user that if additional processing of the result is desired, typical operations could be: **`digital`** – to find one of the various forms the numerical value (floor, ceiling, round, and fractional part in this case); **`digits`** – obtain a list of the digits appearing in the numerical result, and so on. Each choice results in the appearance of another *Next Computation Suggestions Bar*. Depending on the complexity of the result, the *Next Computation Suggestions Bar* will provide appropriate suggestions, such as converting radians to degrees or plotting the result. Additional information about the *Next Computation Suggestions Bar* can be found in the *Documentation Center* by using the entry *guide/WolframPredictiveInterface*.

Note: The *Next Computation Suggestions Bar* can be disabled by selecting *Preferences* in the *Edit* menu. In the *Preferences* window the *Interface* tab is chosen and then the check mark adjacent to *Show Suggestions Bar after last output* is removed.

In Figure 1.8, the case of executing a series of expressions after all the expressions have been entered is shown. In this case, the cursor in placed under and outside of the rightmost cell, which is delineated by the horizontal line. It is seen that the numerical evaluation of each of these trigonometric functions appears in its own cell and corresponds to the order in which they appear. Thus, the first numerical value corresponds to sin(0.13), the second to cos(0.13), and the third to tan(0.13). In addition, it is seen that the three expressions and the three system

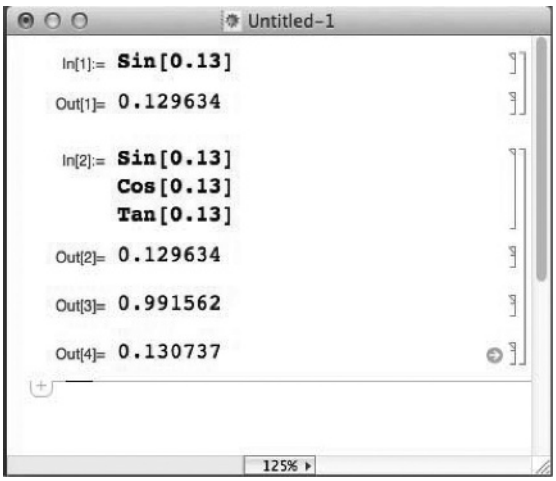


Figure 1.8 Creation of a cell composed of several expressions by using *Enter* following the first two expressions; that is, for **`Sin`** and **`Cos`**, and then *Shift* and *Enter* after **`Tan`** (the *Next Computation Suggestions Bar* has been hidden)

responses reside in a cell that is distinct from the cell of the single computation preceding it. For this case, the *Next Computation Suggestions Bar* has been suppressed.

From Figures 1.7 and 1.8, it is seen that every time *Shift* and *Enter* are simultaneously depressed, the system provides an input identifier, in the first case `In[1]` and in the second case `In[2]`. A similar identifier is created for the output (system response). In the first case, it is `Out[1]` and, in the second case, each cell gets its own identifier: `Out[2]`, `Out[3]`, and `Out[4]`. It is seen that the numerical values of the output identifiers do not have to correspond to the input identifiers. When illustrating how the Mathematica language is used, these input and output identifiers will be omitted.

Aborting a running program

In the event that a calculation appears to be running excessively long, one can abort the calculation by selecting *Evaluation* from the Mathematica menu strip and then choosing *Abort Evaluation*. A calculation is in progress when the thickness of the cell bracket on the rightmost edge of the notebook window increases and appears as a thick black vertical line.

1.4 Delimiters

There are three types of delimiters that are used by Mathematica and each type indicates a very specific set of operational characteristics: ( ) – open/closed parentheses; [ ] – open/closed brackets; and { } – open/closed braces. The parentheses are used to group quantities in mathematical expressions. The brackets are used to delineate the region containing the arguments to all Mathematica and user-created functions and their usage is discussed in Section 3.2. Lastly, the braces are used to delineate the elements of lists, which are defined and discussed in Chapter 2. A summary of these and several other special characters are given in Table 1.2.

1.5 Basic Syntax

1.5.1 Introduction

Mathematica's syntax is different in many respects from traditional programming languages such as Basic, Fortran, and MATLAB® and takes a fair amount of usage to get used to it. In addition, it should be realized that there are often different ways a solution can be coded to obtain one's end results. The "best" way can be judged by such criteria as execution time, code readability, and its number of instructions. The beginner is encouraged to experiment with Mathematica's syntax to see what can be done and how it is done. One's programming sophistication typically increases with continued usage.

The mathematical operators are the traditional ones: + for addition, − for subtraction, / for division, * for multiplication (or a space between variables, which we shall illustrate subsequently), and ^ for exponentiation. In addition, numbers without a decimal point are considered integers and are treated differently from those with a decimal point.

Consider the formula

$$z = \frac{a+b}{c} - \frac{3}{4}ef^2$$

Table 1.2 Special characters and their usage

Character	Name	Usage	Introduced in
`,`	Comma	Separator of elements in a list {…, …} or arguments of a function […, …]	Section 2.2 Section 3.2
_	Underscore	Appended to variable name(s) in the argument of a user-defined function and in a few Mathematica functions	Section 3.2.1
`:=`	Colon equal	Delays the evaluation of the expression on the right-hand side of the equal sign	Section 3.2.1
`;`	Semicolon	Suppresses the display of an excuted expression	Section 1.5.1
		Required when employing more than one expression in the argument of certain functions (denoted in Mathematica as a **`CompoundExpression`**)	Section 3.2.1
		Part of the syntax to access elements of lists	Table 2.4
.	Period	Decimal point; for expressions with decimal numbers, Mathematica will attempt a numerical evaluation	Section 1.5.1
		Performs matrix/vector product when the entities have the appropriate dimensions	Section 2.4.1
`" "`	Quotation marks	Defines a string expression of all characters appearing within the quotation marks	Section 1.9.1
`%`	Percent	Gives the last result generated; to access **`Out[n]`**, enter **`%n`**	Section 1.5.1
`{ }`	Braces	Defines a list, which is a collection of comma-separated objects	Section 2.2
`[ ]`	Brackets	Defines the region containing the arguments of a function	Section 3.2.1
`( )`	Parentheses	Groups terms in equations	Section 1.5.1
`[[ ]]`	Double brackets	Argument indicates the locations of elements of a list	Section 2.3.3
`(* *)`	Parenthesis asterisk	Nonexecutable comment composed of the characters placed between the asterisks	Section 1.6
`/.`	Slash period	Shorthand notation for **`ReplaceAll`**, which replaces all occurrences of a quantity as specified by a rule construct and is used in the form **`/.a->b`**	Section 1.10

(*continued*)

Table 1.2 (*Continued*)

Character	Name	Usage	Introduced in
`->`	Hyphen greater than	Indicates a rule construct: `a->b` means that `a` will be transformed to `b`	Section 1.10
`//`	Double slash	Shorthand notation for `Postfix`: the instruction following the double slash is often used to specify how an output will be displayed or to simplify an expression or to time the execution of an expression	Section 2.2.1 Table 2.1 Table 2.2
`<>`	Greater than less than	Concatenates string objects appearing on each side of these symbols	Section 1.9.1
`/@`	Slash at symbol	Shorthand notation for `Map` and used in the form `f/@h`	Section 3.5.4
`#,#n`	Number sign	Represents the *n*th symbol in a pure function; when $n = 1$, the "1" can be omitted	Section 3.2.2
`##`	Number signs	Represents the sequence of arguments supplied to a pure function	Table 7.1
∈	Element	The Mathematica specification that follows the symbol indicates the domain of the symbol that precedes it.	Table 4.2 Example 4.4
≈	-	Special symbol used in `NProbability`	Section 9.2.1

It is entered as

```
z=(a+b)/c-3/4 e f^2
```

Note that we chose to use a space[1] to indicate multiplication instead of the asterisk (*); that is, there is a space between the 4 and `e` and between the `e` and `f`. The system responds with

$$\frac{a+b}{c} - \frac{3\,e\,f^2}{4}$$

Use of the cursor on the above equation would indicate that there are spaces between the `3` and `e` and between the `e` and `f`. Notice that the system has suppressed the display of `z` in the output. However, it is accessible by simply typing in a new cell either `z` or `%` and then *Shift* and *Enter* simultaneously. When either of these operations is performed, the above expression is displayed. Also, it should be noted that the system had no trouble dealing with five undefined variable names. It simply treated them as symbolic quantities and used them accordingly.

[1] In actuality, a number preceding a symbol does not require a space. It is employed here for consistency. However, with a space used, `2 a` means `2×a` and is the same as `a 2`. On the other hand, without a space `2a` is the same as `2×a`, but `a2` is the name of a variable.

On the other hand, if a decimal point was added to either or both of the integers, that is,

```
z=(a+b)/c-3./4 e f^2
```

was typed into the notebook and *Shift* and *Enter* depressed simultaneously, the system's response is

$$\frac{a+b}{c} - 0.75\, e\, f^2$$

The inclusion of the decimal point forces the system to evaluate all numerical values in the expression when possible. Another way to have the system evaluate all integer expressions is with **N**, which is discussed subsequently. It is mentioned that Mathematica makes a distinction between a decimal number and an integer. An integer is a number without a decimal point and a number with a decimal point is labeled internally a real number. Thus, if one searches for real numbers, only those numbers using a decimal point will be identified as such. All integers will be ignored.

The advantages of Mathematica's ability to seamlessly integrate symbolic manipulation and numerical calculations are now illustrated. We shall use the expression given above for **z** except this time it will be preceded by two additional expressions as follows

```
a=9+1.23^3;
e=h/(11. f);
z=(a+b)/c-3./4 e f^2
```

where the semicolons at the end of the first two expressions were used to suppress their output. The system responds with

$$\frac{10.8609 + b}{c} - 0.0681818\, f\, h$$

It is seen that Mathematica did the substitutions for the variables **a** and **e**, performed all numerical calculations that it could, and did the algebraic simplification that resulted in the cancellation of one **f**.

1.5.2 Templates: Greek Symbols and Mathematical Notation

Greek symbols and symbols with subscripts can be used as variable names. This has the advantage of making portions of the code more readable. However, it can take a bit longer to write the code because of the additional operations that are required to create these quantities. In this book, we shall use the Greek alphabet and the subscripts when practical so that one can more readily identify the code with the equations that have been programmed.

Greek Symbols

Greek symbols can be used directly for variable names with the use of the *Special Characters* palette. To insert a special character, one places the cursor in the notebook at the location at which the character is to be placed. Then one selects the character from the palette and the

selected character will appear at the location selected in the notebook. For example, using the palette to create the expression

```
x=Ω^γ
```

displays

Ω^{γ}

These expressions can be used in a regular manner; thus, the execution of

```
Ω=7.;
γ=2.;
x=Ω^γ
```

yields `49` and we have used the semicolon (;) to suppress the display of the first two lines of the program.

Mathematical Notation

Mathematica provides a means to easily create program instructions using standard mathematical notation such as exponents, square roots, and integrals. This notation is accessed by using the *Typesetting* (or *Calculator*) portion of the *Basic Math Assistant*, which is selected from the *Palettes* menu, as shown in Figure 1.9. To illustrate the palette's use, we shall repeat the example given in Section 1.5.2. Additional applications of mathematical notation are given in the subsequent chapters. Thus,

```
Ω=7.
γ=2.
x=Ω^γ
```

yields `49`. The last instruction was obtained from clicking on the template superscript symbol ($\square^{\square}$).

For another example, consider the cube root of 27. In this case, the use of the *Basic Math Assistant* palette results in

```
x = ∛27
```

which yields `3`. This instruction was obtained from clicking on the template *n*th root symbol ($\sqrt[\square]{\square}$).

The use of the *Typesetting* portion of the *Basic Math Assistant* is also very useful in annotating the graphical display of results as illustrated in Table 6.8.

One can also use the *Basic Math Assistant* to create variables that contain subscripts and superscripts. Thus, using the *Basic Math Assistant* to create the relation $d_a = e^{b+c}$, we have

```
d_a=E^(b+c)
```

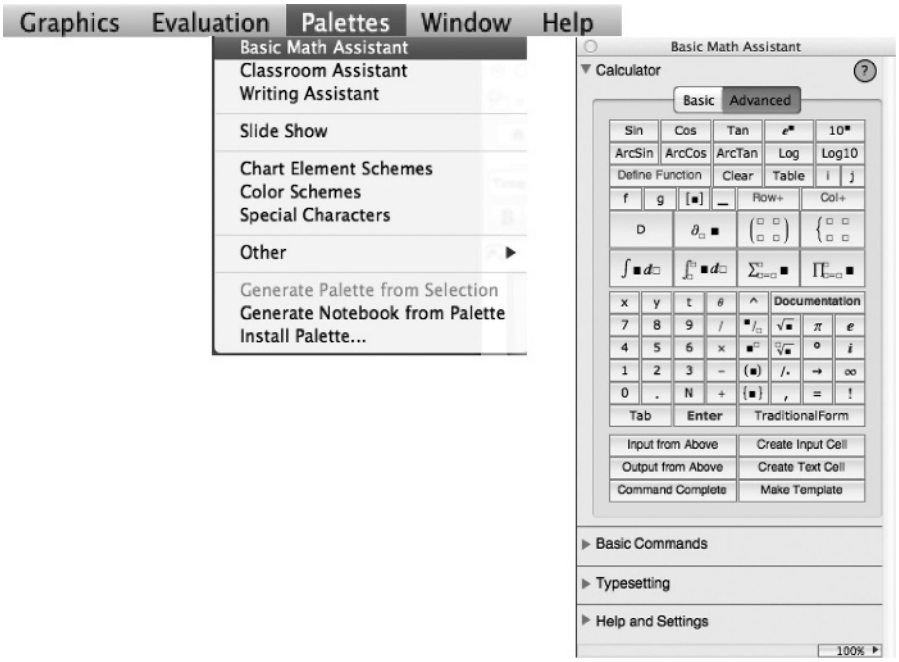


Figure 1.9 Opening the *Basic Math Assistant* window to access advanced mathematical notation templates

which displays

```
e^(b+c)
```

and

```
f=1+d_a
```

displays

```
1+e^(b+c)
```

The symbol **e** is an approximation to the way that Mathematica displays *e*.

A subscript and superscript appearing on the right hand of the equal sign are treated the same. Using the *Basic Math Assistant* template, consider the following

```
a=7;

b=c^a+d_a
```

which displays

```
c^7+d_7
```

It is important to note that, while **d** and **a** are each symbols, the variable d_a is not a symbol entity. To convert it to a symbol, the following steps have to be taken. First, the *Notation* package has to be loaded by using

```
Needs["Notation`"]
```

This opens a *Notation Palette*, which must be used to convert the subscripted variable to a symbol. From this palette, **Symbolize[□]** is selected. In the square, the subscripted symbol is entered, which in our case is

```
Symbolize[d_a]
```

From this point on, d_a is a symbol.

This conversion is required for several of the templates appearing in the *Typesetting Palette* if in subsequent use it is necessary to treat them as a single symbol. For the use of subscripted symbols in user-created functions, see Section 3.2.1.

1.5.3 Variable Names and Global Variables

User-created names for variables and functions must start with a letter, are case sensitive, and are permanent for the duration of the Mathematica session unless specifically removed or appear in certain Mathematica commands. There does not appear to be a restriction on the number of alphanumeric characters that can be used to create a variable name. It is good practice to remove the variables after one has finished using them and before proceeding further. This removal is done with **Clear** or with **ClearAll**. The arguments of these commands are comma-separated names of the variables to be deleted (cleared). Either of these commands can be used in one of two ways. They can be employed after the completion of a procedure to delete the variable names that were just used or they can be employed before a new procedure to ensure that the variable names to be used do not have another definition.

The naming convention in Mathematica it that *all* Mathematica function names begin with a capital letter and following the last letter of the function name are a pair of open/closed brackets [ ]. Between these brackets, one places expressions, procedures, and lists according to the specifications regarding the usage of that function. Consequently, some care should be exercised when creating variable names and function names. One way to eliminate the possibility of a conflict is to start each variable name with a lower case letter. In any case, do not use the following single capital letters as variable names: **C**, **D**, **E**, **I**, **N**, and **O**.

We shall now show the care that has to be exercised when choosing variable names since, as previously mentioned, all variable names and their respective definitions or assignments remain available until either they are redefined or cleared. Suppose that earlier in the notebook one evaluated the expression

```
a=0.13^2
```

and the system responded with

```
0.0169
```

Then, later in the notebook, one enters the expression

```
z=(a+b)/c-e f^2
```

where it was thought that **a**, **b**, **c**, **e**, and **f** were symbols; that is, no assignment has been given them. However, the system response is

$$\frac{0.0169+b}{c} - e\,f^2$$

since the variable **a** had already been assigned the numerical value shown above. To avoid this type of unintended substitution, one uses **Clear** as follows

```
Clear[a,b,c,e,f]
z=(a+b)/c-e f^2
```

Although it may not always be explicitly shown, it is implicit that for all examples presented in this book the **Clear** *function will have been employed using the appropriate variable names and user-created function names to ensure that this type of unintended substitution doesn't occur.*

One can clear all user-created variable names and function names without specifying them individually by using

```
ClearAll["Global`*"]
```

The "backwards prime" is required and the asterisk (*) indicates that all global names are to be included.

Importance of global variables

Global variables play in important role in the creation and use of functions. It will be seen that when used properly, the fact that variables are by default global variables can simplify the implementation of user-created functions and programming in general. These topics are introduced and discussed in Chapter 3.

1.6 Mathematical Constants

The common mathematical constants $j = i = \sqrt{-1}$, π, ∞, e, $\pi/180$ (conversion from degrees to radians) and $\gamma = 0.57721566$ (Euler's constant) are represented in Mathematica as listed in Table 1.3. Thus, to enter the expression $e^{j\pi/4}$, we type

```
E^(I π/4)
```

Table 1.3 Mathematical constants

Constant	Mathematica function
$i = j = \sqrt{-1}$	`I`
π	`Pi` or† π, which is obtained from the *Special Characters* palette under the α tab within the *Letters* tab
∞	`Infinity` or ∞, which is obtained from the *Special Characters* palette under ∞ tab within the *Symbols* tab
e	`E` (or `Exp[1]`)
$\pi/180$	`Degree` (See text for usage)
γ	`EulerGamma` (Euler's constant)

† Note: `Pi^2.` and `π^2.` return the same numerical values; however, `Pi^2` and `π^2` return π^2.

and the response is

$$e^{\frac{i\pi}{4}}$$

where `e` is used to represent the way that Mathematica displays e.

To convert this symbolic expression to a numerical value, we use the Mathematica function

```
N[expr,n]
```

where `expr` is the expression that is to be converted to a numerical value and n is the number of digits of precision that is used during its evaluation of the expression. Thus,

```
N[E^(I π/4)]
```

gives

```
0.707107+0.707107 i
```

where the symbol `i` is an approximation to the way Mathematica displays `I`. By omitting n in `N`, we have settled for machine precision. However, the number of digits displayed is that specified in *Preferences*, which in this case is six. The same result will be obtained without `N` if the integer 4 was replaced by 4.0; that is, by its decimal form. The use of the decimal point to force numerical evaluation of all numbers will be used frequently.

To obtain the previous results using 20 digits of precision, we modify the above expression as

```
N[E^(I π/4),20]
```

which produces

```
0.70710678118654752440+0.70710678118654752440 i
```

To convert an angular value from degrees to radians, the constant **Degree** is used in the following manner. Let the angle be 30°; then to express this angle in radians we use

```
N[Degree 30]
```

which results in

```
0.523599
```

The function **N** is not required if its argument is of the form **Degree 30.0**; that is, if the decimal form is used. However, **Degree** can also be used to append to 30 the degree symbol. Hence, entering

```
Degree 30 (* or 30 Degree *)
```

results in

```
30°
```

We have indicated an equivalent syntax as a comment.

1.7 Complex Numbers

Complex numbers are formed with the appropriate use of **I**. The resulting quantity can be manipulated and parsed by using the functions shown in Table 1.4. For example, let us determine the numerical value of the magnitude and the imaginary part of

$$r = (1 + 2j)^j$$

Since we are interested in numerical values, we express the numerical values in their decimal form and enter

```
r=(1.+2. I)^I
m=Abs[r]
im=Im[r]
```

Table 1.4 Complex number manipulation functions

		Output		
Operation	Mathematica function	$z = 2 + 3j$	$z = 2.0 + 3j$	$z = x + jy$
Real part	`Re[z]`	`2`	`2.`	`-Im[y]+Re[x]`
Imaginary part	`Im[z]`	`3`	`3`	`Im[x]+Re[y]`
Complex conjugate	`Conjugate[z]`	`2-3 i`	`2.-3 i`	`Conjugate[x]- I Conjugate[y]`
Argument	`Arg[z]`	`ArcTan[3/2]`	`0.982794`	`Arg[x+I y]`

The system responds with

```
0.22914+0.23817 i
0.03305
0.23817
```

The first value corresponds to r, the second value to the magnitude of r, and the last value to the imaginary part of r. It is mentioned again that only *Enter* was depressed after each of the first two lines and *Shift* and *Enter* were simultaneously depressed after the third line. These three lines of code reside in one cell and each of the output results resides in its own cell. However, all six lines reside in one cell.

1.8 Elementary, Trigonometric, Hyperbolic, and a Few Special Functions

Elementary Functions

The syntax of several elementary functions is shown in Table 1.5. The functions have been illustrated in this table for real, integer, symbolic, and complex quantities.

Trigonometric Functions

The names for trigonometric and inverse trigonometric functions are listed in Table 1.6. The arguments of these functions can be integers, real numbers, or complex quantities. Thus, to determine the value of the $\cot(\pi/5)$, we enter

```
Cot[π/5]
```

and obtain

$$\sqrt{1+\frac{2}{\sqrt{5}}}$$

On the other hand, if a numerical value was desired, then one would enter

```
Cot[π/5.]
```

and the system would respond with `1.37638`.

If the numerical value of $\cos(2+3j)$ were desired, we enter

```
Cos[2.+3 I]
```

and obtain

```
-4.18963-9.10923 i
```

Table 1.5 Elementary functions

		Examples of output for explicit forms of z			
Operation	Mathematica function*	$z = 2.0$	$z = x + jy$	$z = 2 + 3j$	$z = 2.0 + 3j$
$\sqrt{z}$	`Sqrt[z]`	`1.41421`	$\sqrt{\texttt{x + i y}}$	$\sqrt{\texttt{2 + 3 i}}$	`1.67415+0.895977 i`
$\sqrt[3]{x}$	`CubeRoot[z]`	`1.25992`	$\sqrt[3]{\texttt{x + i y}}$	-	-
e^z	`E^z` or `Exp[z]`	`7.38906`	$\texttt{e}^{\texttt{x+i y}}$	$\texttt{e}^{\texttt{2+3 i}}$	`-7.31511+1.04274 i`
$\ln_e z$	`Log[z]`	`0.693147`	`Log[x+y i]`	`Log[2+3 i]`	`1.28247+0.982794 i`
$\log_{10} z$	`Log[10,z]` or `Log10[z]`	`0.30103`	$\frac{\texttt{Log[x + y i]}}{\texttt{Log[10]}}$	$\frac{\texttt{Log[2 + 3 i]}}{\texttt{Log[10]}}$	`0.556972+0.426822 i`
$\lvert z \rvert$	`Abs[z]`	`2.`	`Abs[x+y i]`	$\sqrt{13}$	`3.60555`
signum(z)	`Sign[z]`	`1`	`Sign[x+y i]`	$\frac{\texttt{2 + 3 i}}{\sqrt{13}}$	`0.5547+0.83205 i`
$z!$	`Factorial[z]` or `z!`	`2.`	`(x+i y)!`	`(2+i 3)!`	`-0.44011-0.063637 i`

* When used symbolically, `Sqrt[z^2]` will not return `z` and `Log[z^2]` will not return `2 Log[z]`. To obtain these simplifications, `PowerExpand` must be used as shown in Table 4.1.

Table 1.6 Trigonometric and inverse trigonometric functions

Trigonometric function	Mathematica function	Inverse trigonometric function	Mathematica function
$\sin z$	`Sin[z]`	$\sin^{-1} z$	`ArcSin[z]`
$\cos z$	`Cos[z]`	$\cos^{-1} z$	`ArcCos[z]`
$\tan z$	`Tan[z]`	$\tan^{-1} z$ or $\tan^{-1} y/x$	`ArcTan[z]` or `ArcTan[x,y]`
$\csc z$	`Csc[z]`	$\csc^{-1} z$	`ArcCsc[z]`
$\sec z$	`Sec[z]`	$\sec^{-1} z$	`ArcSec[z]`
$\cot z$	`Cot[z]`	$\cot^{-1} z$	`ArcCot[z]`

To obtain the value of a trigonometric function when the angle is given in degrees, we use `Degree`. Thus, if the angle is 35°, then its cosine is found by entering

```
Cos[Degree 35.]
```

which yields

```
0.819152
```

The decimal form of the number is required to produce this result. If the decimal point were not used, the output would have been

```
Cos[35°]
```

For the case when the argument is a complex symbolic quantity $z = x + jy$, refer to Table 4.1.

Hyperbolic Functions

The names for hyperbolic and inverse hyperbolic functions are listed in Table 1.7. The arguments of these functions can be complex quantities. Thus, to determine the value of the $\cosh(2 + 3j)$, we enter

```
Cosh[2.+3 I]
```

and obtain

```
-3.72455+0.511823 i
```

For a symbolic complex quantity $z = x + jy$, refer to Table 4.1.

Table 1.7 Hyperbolic and inverse hyperbolic functions

Hyperbolic function	Mathematica function	Inverse hyperbolic function	Mathematica function
$\sinh z$	`Sinh[z]`	$\sinh^{-1} z$	`ArcSinh[z]`
$\cosh z$	`Cosh[z]`	$\cosh^{-1} z$	`ArcCosh[z]`
$\tanh z$	`Tanh[z]`	$\tanh^{-1} z$	`ArcTanh[z]`
$\text{csch}\, z$	`Csch[z]`	$\text{csch}^{-1} z$	`ArcCsch[z]`
$\text{sech}\, z$	`Sech[z]`	$\text{sech}^{-1} z$	`ArcSech[z]`
$\coth z$	`Coth[z]`	$\coth^{-1} z$	`ArcCoth[z]`

Special Mathematical Functions

Mathematica has a large collection of special functions, which can found in the *Documentation Center* window using the search entry *guide/SpecialFunctions*. It will be found that many functions used to obtain solutions to engineering topics are available. The application of several of these special functions will be illustrated in later chapters. Some of the more common functions used in engineering applications can be found in Table 4.4.

1.9 Strings

1.9.1 String Creation: `StringJoin[]` *and* `ToString[]`

Any combination of numbers, letters, and special characters that are linked together to form an expression that does not perform any Mathematica operation is denoted a string. There are a large number of commands that can be used to manipulate strings. Our use for them will be limited primarily to creating annotated output for graphics. Therefore, we shall introduce only a few string creation and manipulation commands. Additional enhancements to string expressions such as subscripts and superscripts are presented in Table 6.8.

A string is created by enclosing the characters with quotation marks. Thus,

```
s1="text"
```

creates a string variable `s1`, which is composed of four characters. To concatenate two or more strings, we use either

```
StringJoin[s1,s2,...]
```

or

```
s1<>s2<>...
```

where `sN` are strings. Thus, if

```
s1="text";
s2="Example of ";
```

then either

```
s2<>s1
```

or

```
StringJoin[s2,s1]
```

yields

```
Example of text
```

One is also able to convert a numerical quantity to a string by using

```
ToString[expr]
```

where **expr** is a numerical value or an expression that leads to a numerical value. Thus, for example, if we would like to evaluate and display in an annotated form the value of

$$\sqrt{|\sin(x^2)|}$$

when $x = 0.35$, the instructions are

```
x=0.35;
p="When x = "<>ToString[x]<>", √|sin(x²)| = "<>
 ToString[√Abs[Sin[x²]]]
```

which displays

```
When x = 0.35, √|sin(x²)| = 0.349562
```

In obtaining this expression, we have used the appropriate *Basic Math Assistant* templates in the expression for **p**.

1.9.2 Labeled Output: `Print[]`, `NumberForm[]`, `EngineeringForm[]`, *and* `TraditionalForm[]`

An improved way of displaying output is to combine the string conversion commands with **Print**, which is implemented with

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

where **exprN** is a constant, a symbolic expression, the numerical value of a computed variable, a string, or a Mathematica object. To print multiple lines, one can use multiple **Print** commands or one **Print** command and the **Row** command, which is discussed in Figure 6.2 and Table 6.8.

For example, the results of the previous example using complex numbers is modified as follows

```
r=(1+2 I)^I;
Print["r = ",r]
Print["|r| = ",N[Abs[r]],"  Im[r] = ",N[Im[r]]]
```

which yields

```
r=(1+2 i)^i
|r|=0.3305  Im[r]=0.23817
```

The number of digits that are displayed and their form can be controlled by using **NumberForm** or **EngineeringForm**. These commands, respectively, are given by

```
NumberForm[val,n]
EngineeringForm[val,n]
```

where **val** is the numerical quantity to be displayed with **n** digits. To illustrate the usage of these commands, we shall display $|r|$ in the above example with 10 digits using both formats. Then,

```
r=Abs[(1+2. I)^I];
Print["|r| (Number form) = ",NumberForm[r,10]]
Print["|r| (Engrg Form) = ",EngineeringForm[r,10]]
```

which displays

```
|r| (Number form) = 0.3304999676
|r| (Engrg Form) = 330.4999676×10^-3
```

The functions **NumberForm** and **EngineeringForm** are frequently used to format numerical output when annotating graphics.

The function that generates traditional mathematical notation is

```
TraditionalForm[expr]
```

where **expr** is a mathematical expression using Mathematica's syntax. For example, if

```
expre=BesselJ[n,x] Sin[x];
```

then

```
Print["y = ",TraditionalForm[expre]]
```

displays

$$y = \sin(x) J_n(x)$$

Table 1.8 Decimal-to-integer conversion functions

Method	Mathematica function	x (Argument)	y (Output)
Closest integer to x	`y=Round[x]`	`-0.6`	`-1`
		`2.7`	`3`
		`2.49-2.51 I`	`2-3`
Smallest integer $\geq x$	`y=Ceiling[x]`	`-0.6`	`0`
		`2.7`	`3`
		`2.49-2.51 I`	`3-2`
Greatest integer $\leq x$	`y=Floor[x]`	`-0.6`	`-1`
		`2.7`	`2`
		`2.49-2.51 I`	`2-3`
Integer part of x	`y=IntegerPart[x]`	`-0.6`	`0`
		`2.7`	`2`
		`2.49-2.51 I`	`2-2`
Replace a real number $< 10^{-10}$ with integer 0	`Chop[x]`	`1.2 10^(-15)+4.5 I`	`4.5 I`

1.10 Conversions, Relational Operators, and Transformation Rule

Decimal-to-Integer Conversion

There are several ways that one can round a decimal value to an integer depending on the rounding criterion. These methods are summarized in Table 1.8.

Relational Operators

Several relational operators are listed in Table 1.9. These operators are typically used in such functions as **`If`**, **`Which`**, **`While`**, **`Solve`**, **`DSolve`**, and **`Simplify`**.

Substitution: Transformation Rule

Mathematica has a very specific manner in which one or more expressions can be substituted into another expression. The procedure that is introduced here is used throughout

Table 1.9 Relational and logical operators

Mathematical symbol	Mathematica symbol
$=$	`==`
$>$	`>`
$\geq$	`>=`
$<$	`<`
$\leq$	`<=`
$\neq$	`!=`
And	`&&`
Or	`||`

Mathematica's implementation and is one of the primary ways in which one gains access to the expressions and results from many of its built-in functions. It will be illustrated here using some simple examples and then again in later sections with other applications.

There are two distinct sets of symbols that will now be defined. The first is /. (slash period), which is shorthand for replace all (**ReplaceAll**). The second notation is **a** -> **b** (hyphen greater than), which is a rule construct that means that **a** will be transformed to **b**. Then the statement

```
v/.a->b
```

instructs Mathematica to perform the following: in the expression **v**, everywhere **a** occurs replace it with **b**. The quantity **b** can be a number, a symbol, or an expression. If **a** does not appear in **v**, then nothing happens. Furthermore, several substitutions can be made sequentially in the order that they appear. Thus, an extension of the previous expression could be[2]

```
v/.a->b/.c->Sqrt[e+f]
```

For an illustration of the use of this construct and its effect, we return to the previous example given by

```
a=9+1.23^3;
e=h/(11. f);
z=(a+b)/c-3./4 e f^2
```

which is one way that substitution can be performed. The same results can be obtained as follows

```
z=(a+b)/c-3./4 e f^2;
z/.a->9+1.23^3/.e->h/(11. f)
```

which results in

$$\frac{10.8609+b}{c} - 0.0681818\, f\, h$$

and is the same result that was obtained previously.

Finding Built-in Function Options

Many built-in functions have options that can be changed. The options that can be changed and that are currently being used by a built-in function are determined by typing and executing

```
Options[FunctionName]
```

[2] This transformation can also be performed using a list, which is discussed in Chapter 2, as follows:
`v/.{a->b,c->Sqrt[e+f]}`

which displays all options, their option names, and their current selections for the function **FunctionName**. Another way to determine these options is to enter **FunctionName** into the *Documentation Center* search box. As an example, we shall see what options are available for **NDSolveValue**, which performs a numerical evaluation of differential equations. Then,

```
Options[NDSolveValue]
```

displays

```
{AccuracyGoal->Automatic, Compiled->Automatic,
DependentVariables->Automatic, EvaluationMonitor->None,
InterpolationOrder->Automatic, MaxStepFraction->1/10,
MaxSteps->10000, MaxStepSize->Automatic, Method->Automatic,
NormFunction->Automatic, PrecisionGoal->Automatic,
SolveDelayed->Automatic, StartingStepSize->Automatic,
StepMonitor->None, WorkingPrecision->MachinePrecision}
```

1.11 Engineering Units and Unit Conversions: Quantity[] and UnitConvert[]

With the introduction of Mathematica 9, the use of units has become an integral part of Mathematica[3]. In this section, we shall introduce its basic use.

The function that attaches units to a numerical or symbolic quantity is

```
xu=Quantity[mag,unit]
```

where **mag** is the magnitude of the quantity with units **unit**, which is the unit designation presented as a string; that is, it is contained within quotation marks. Common engineering units are listed in Table 1.10. In this table, the middle column gives the unambiguous unit designations that are used by Mathematica. These are the ones that are used in this section. The units in the first column are those that are commonly used and are internally converted by **Quantity** to those in the second column. The arguments of **Quantity** can be accessed with

```
mag=QuantityMagnitude[xu]
unit=QuantityUnit[xu]
```

For example, if the length of x is 6.5 m, then

```
xu=Quantity[6.5,"Meters"] (* or Quantity[6.5,"m"] *)
```

displays

```
6.5 m
```

[3] Additional information can be found by entering *tutorial/UnitsOverview* in the *Documentation Center* search window.

Table 1.10 Common engineering units for `Quantity[]`

```
p=QuantityUnit[Quantity[x,str]]
q=Quantity[18.,ToString[p]]
```

str	p	q
Length/Volume		
"m"	Meters	18. m
"cm"	Centimeters	18. cm
"mm"	Millimeters	18. mm
"nm"	Nanometers	18. nm
"Ang"	Angstroms	18. Å
"ft"	Feet	18. ft
"in"	Inches	18. in
"mi"	Miles	18. mi
"L"	Liters	18. L
"gal"	Gallons	18. gal
Force/weight/pressure		
"kg"	Kilograms	18. kg
"N"	Newtons	18. N
"g"	Grams	18. g
"Pa"	Pascals	18. Pa
"lb"	Pounds	18. lb
Temperature		
"K"	KelvinsDifference	18. K
"Kelvins"	Kelvins	18. K
"C"	DegreesCelsiusDifference	18. °C
"Celsius"	Celsius	18. °C
"F"	DegreesFahrenheitDifference	18. °F
"Fahrenheit"	Fahrenheit	18. °F
Energy/power		
"W"	Watts	18. W
"J"	Joules	18. J
"btu"	BritishThermalUnitsIT	18. BTU_{IT}
Electrical		
"A"	Amperes	18. A
"H"	Henries	18. H
"ohm"	Ohms	18. Ω
"farad"	Farads	18. F
"coulomb"	Coulombs	18. C
"V"	Volts	18. V
"T"	Teslas	18. T
Time/frequency		
"s"	Seconds	18. s
"ms"	Milliseconds	18. ms
"micros"	Microseconds	18. μs
"Hz"	Hertz	18. Hz
"GHz"	Gigahertz	18. GHz
"hr"	Hours	18. h

To determine the magnitude and the units of x, one uses

```
mag=QuantityMagnitude[xu]
unit=QuantityUnit[xu]
```

which displays, respectively,

```
6.5
Meters
```

To perform a calculation with units, consider the following determination of the average velocity of a device that travels 5 meters in 2 seconds

```
v=Quantity[5.,"Meters"]/Quantity[2,"Seconds"]
```

which displays

```
2.5 m/s
```

To illustrate the usage of **Quantity** further, consider the following quantities: e_{33} (C·m^{-2}), c_{33} (N·m^{-2}), and ε_{33} (F·m^{-1}). We shall show that the following quantity is nondimensional

$$k = \frac{e_{33}^2}{c_{33}\varepsilon_{33}}$$

Thus,

```
k=Quantity[e33,"Coulombs"/"Meters"^2]^2/
  (Quantity[c33,"Newtons"/"Meters"^2]*
  Quantity[Subscript[ε33,"Farads"/"Meters"])
```

which gives

$$\frac{e_{33}^2}{c_{33}\varepsilon_{33}}$$

Since there are no units appearing in this result, k has no dimensions.

The conversion from one system of units to another is done with

```
UnitConversion[xunit,unit]
```

where **xunit** has been created by **Quantity** and **unit** is a string representing the desired unit(s) to which **xunit** is to be converted.

For example, to convert a temperature of, say, 60 degrees Fahrenheit to degrees Celsius, we use

```
UnitConvert[Quantity[60.,"Fahrenheit"],"Celsius"]
```

to obtain

```
15.5556 °C
```

To covert the temperature to degrees Kelvin, we use

```
UnitConvert[Quantity[60.,"Fahrenheit"],"Kelvins"]
```

to obtain

```
288.706 K
```

To convert 55 miles/hr to meters/second, we use

```
UnitConvert[Quantity[55.,"Miles"/"Hours"],
 "Meters"/"Seconds"]
```

to obtain

```
24.5872 m/s
```

To convert 20 lb·ft^{-2} to its SI equivalent, we use

```
UnitConvert[Quantity[20.,"Pounds"/"Feet"^2],"SI"]
```

to obtain

```
97.6486 kg/m^2
```

Other useful unit conversion commands are **`CommonUnits`**, which converts the various units to compatible dimensions, and **`UnitSimplify`**, which attempts to simplify the specified units.

1.12 Creation of CDF Documents and Documents in Other Formats

The notebook environment can also be used to create documents as if the notebook were a word processor. This is done by using the *Writing Assistant* found in the *Palette* menu to create cells that accept text and cells that accept equations in addition to those cells that provide the usual Mathematica computational capabilities. This palette is shown in Figure 1.10. The final document can then be saved in different formats by selecting *Save As* from the *File* menu and then choosing the desired format from the *Format* selections at the bottom of the window.

One format that provides a unique capability is Mathematica's own CDF (computable document format) document creator, which permits one to create electronic documents that can contain interactive graphics. The creation of interactive graphics is performed by **`Manipulate`**, which is discussed in detail in Chapter 7. When the file is opened by the CDF reader, the document has all the characteristics of a regular noneditable file, except that now any graphic entities created by **`Manipulate`** retain their interactivity. There are several ways to convert a notebook to a CDF file. One such method, after the notebook has been completed,

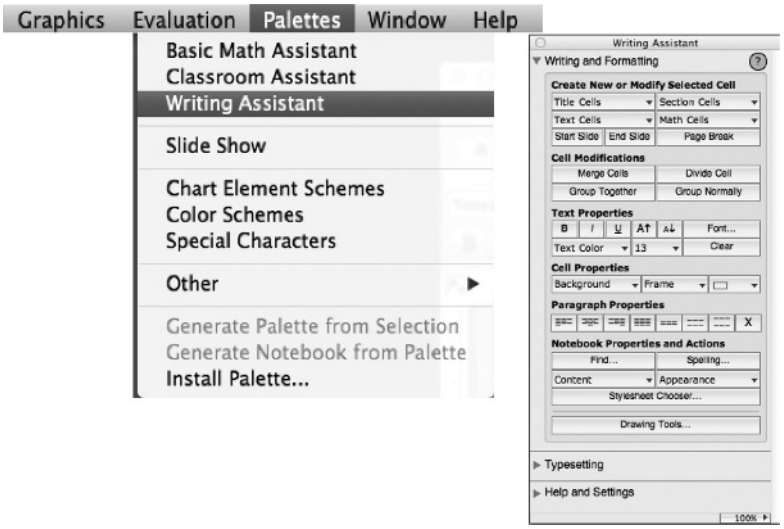


Figure 1.10 Opening the *Writing Assistant* window to access cell formatting templates

is to select *CDF Export* from the *File* menu and then select *Standalone*. This opens a window in which the user selects a file name for the converted notebook and a directory in which it is to reside and then clicks *Continue*. The file is created and the window is exited.

1.13 Functions Introduced in Chapter 1

In addition to the functions listed in Tables 1.4 to 1.8, the additional functions introduced in this chapter are listed in Table 1.11.

Table 1.11 Summary of additional commands introduced in Chapter 1

Command	Usage
`Clear`	Clears from memory specified user-created definitions
`ClearAll["Global`*"]`	Clears from memory all user-created definitions
`EngineeringForm`	Prints real numbers in engineering notation with specified precision
`N`	Gives the numerical value of an expression with specified precision
`NumberForm`	Prints to a specified precision real numbers in an expression
`Options`	Gives the options available for a specified function and their current values
`Print`	Prints an expression
`Quantity`	Appends units to a number or symbol
`StringJoin`	Concatenates a set of string objects in the order that they appear
`Symbolize`	Convert subscripted variable for internal use as single symbol
`ToString`	Converts an expression to a string
`UnitConvert`	Converts a quantity from one system of units to another

Exercises

Sections 1.5 and 1.8

1.1 For $b = 1.2$, $a = 1.5b$, and $\nu = 0.3$, determine κ when

$$\kappa = \frac{(40 + 37\nu)b^4 + (16 + 10\nu)a^2b^2 + \nu a^4}{12(1 + \nu)b^2(3b^2 + a^2)}$$

1.2 For $\gamma_1 = 0.6$, $\gamma_2 = 0.4$, and $m_o = 0.71$, determine Ω_c when

$$\Omega_c = \frac{1}{\sqrt{\gamma_1^3\gamma_2^3\left(m_o + \alpha + \beta\right)}}$$

and

$$\alpha = \gamma_1\left[\frac{\left(3\gamma_1 + \gamma_2\right)^2}{28\gamma_2^2} + \frac{9}{20\gamma_2^2} - \frac{3\gamma_1 + \gamma_2}{4\gamma_2^2}\right]$$

$$\beta = \gamma_2\left[\frac{\left(3\gamma_2 + \gamma_1\right)^2}{28\gamma_1^2} + \frac{9}{20\gamma_1^2} - \frac{3\gamma_2 + \gamma_1}{4\gamma_1^2}\right]$$

1.3 For $n = 5$, determine the value of c when

$$c = \frac{M\left(1 - M^2\right)\sin\alpha}{\left(1 + M^2 - 2M\cos\alpha\right)^2}$$

and

$$\cos\alpha = \sqrt{\left(\frac{1 + M^2}{4M}\right)^2 + 2} - \left(\frac{1 + M^2}{4M}\right)$$

$$M = \frac{1}{\sin\left(\pi/n\right)}$$

1.4 For $\gamma = 60°$, $\alpha = 35°$, and $n = 4/3$, find the value of d when

$$d = \alpha - \gamma + \sin^{-1}\left[n\sin\left(\gamma - \sin^{-1}\left\{\frac{\sin\alpha}{n}\right\}\right)\right]$$

1.5 For $x = 0.45$, determine the value of k when

$$k = \frac{1.2}{x}\left[\sqrt{16x^2 + 1} + \frac{1}{4x}\ln\left(\sqrt{16x^2 + 1} + 4x\right)\right]^{-2/3}$$

1.6 For $\varepsilon = 0.00025$, $\text{Re} = 8 \times 10^5$, and $D = 0.3$, determine f when

$$f = \left\{ \left(\frac{64}{\text{Re}} \right)^8 + 9.5 \left[\ln \left(\frac{\varepsilon}{3.7D} + \frac{5.74}{\text{Re}^{0.9}} \right) - \left(\frac{2500}{\text{Re}} \right)^6 \right]^{-16} \right\}^{1/8}$$

1.7 For $\lambda = 4.73$, $r = 3$, $\Omega = 300$, and $k = 450$, determine ω_c when

$$\omega_c = \frac{1}{1-\beta^2} \left[-\Omega\beta^2 + \sqrt{\Omega^2\beta^4 + \left(1-\beta^2\right)\omega_o^2} \right]$$

and

$$\beta = \lambda r$$
$$\omega_o = kr\lambda^2$$

1.8 For $x = 0.4$ and $y = 0.6$, determine u when

$$u = -\frac{1}{x} \left\{ \ln \left(1 - \sqrt{1-y} \right) - \sqrt{1-y} \right\}$$

1.9 For $g = 9.81$, $\theta = 17°$, $x = 45$, and $v_o = 14$, determine z when

$$z = -\frac{gx^2}{2v_o^2 \cos^2\theta} + x \tan\theta$$

1.10 For $M = 0.75$ and $k = 1.4$, determine p/p_o when

$$\frac{p}{p_o} = \left[1 + \{(k-1)/2)\} M^2 \right]^{\frac{-k}{k-1}}$$

1.11 For $r_1 = 0.01$, $\varepsilon_1 = 0.02$, $T_1 = 77$, $r_2 = 0.05$, $\varepsilon_2 = 0.05$, $T_2 = 300$, and $\sigma = 5.67 \times 10^{-8}$, determine q_{12} when

$$q_{12} = \frac{4\pi\sigma r_1^2 \left(T_1^4 - T_2^4 \right)}{\frac{1}{\varepsilon_1} + \frac{1-\varepsilon_2}{\varepsilon_2} \left(\frac{r_1}{r_2} \right)^2}$$

1.12 For $\varphi = \pi/10.0$, determine the value of χ when

$$\chi = 2\tan^{-1} \left\{ \tan \left(\frac{\pi}{4} + \frac{\varphi}{2} \right) \left[\frac{1 - e\sin\varphi}{1 + e\sin\varphi} \right]^{e/2} \right\} - \frac{\pi}{2}$$

1.13 For $B = 2$ (an integer), determine the value of w when

$$w = \frac{1}{2\sqrt{2}}\sqrt{3B^2 + \sqrt{9B^4 + 16B^2}}$$

Convert the result to a decimal number.

1.14 For $\theta_o = 27°$, determine the value of p when

$$p = 2\sin^2\left(\frac{\theta_o}{2}\right)\left\{1 + \frac{2}{\pi}\cot\left(\frac{\theta_o}{2}\right)\ln\left[\tan\left(\frac{\pi}{4} + \frac{\theta_o}{2}\right)\right]\right\}$$

Sections 1.6 and 1.8

1.15 Show numerically using 50 digits of precision that

$$\sin(\pi/15) = \frac{1}{4}\sqrt{7 - \sqrt{5} - \sqrt{30 - 6\sqrt{5}}}$$

Section 1.11

1.16 The Reynolds number is given by

$$\mathrm{Re} = \frac{\rho v L}{\mu}$$

where L (m) is a characteristic length, μ (Pa·s) is the dynamic viscosity, v (m·s^{-1}) is the mean velocity, and ρ (kg·m^{-3}) is the density. Show that Re is a nondimensional quantity.

1.17 A similarity coordinate transform is given by

$$\eta = y\sqrt{\frac{U\rho}{\mu x}}$$

where x (m) and y (m) are coordinates, U (m/s) is a velocity, μ (Pa·s) is the dynamic viscosity, and ρ (kg·m^{-3}) is the density. Show that η is a nondimensional quantity.

1.18 The natural frequency coefficient Ω for a beam is defined as

$$\Omega^4 = \frac{\omega^2 \rho A L^4}{EI}$$

where E (N·m^{-2}) is the Young's modulus, I (m^4) is the moment of inertia, L (m) is the length, A (m^2) is the area, ρ (kg·m^{-3}) is the density, and ω (rad·s^{-1}) is the radian frequency. Show that Ω is a nondimensional quantity.