Introduction to Mathematica

Getting Started

You begin by starting Mathematica. If you are working on a personal computer (PC), open the Mathematica program. If you are working on a Unix workstation (like those in the Chemical Engineering Department’s Cody Lab), you will need to open a terminal and then type mathematica on the command line. Note that Version 3.0 for PC and Unix machines are quite similar. (A newer version, Version 4.0, is now available, but will not be loaded on the workstations until Spring Semester 2000 at the earliest.)

A notebook in Mathematica is like a document in Microsoft Word. Whatever you type (Input) is entered into a notebook file, which is designated with the extension .nb. Just as Microsoft Word starts with an open document called Document1, Mathematica starts with an open notebook called Untitled-1.nb.

Your begin your Mathematica session by typing Input into the Mathematica workbook. Input is placed into a cell. Cells are designated with brackets at the far right of the screen. Normally, a cell consists of a single mathematical Input, the Output resulting from an evaluating an Input, or Text. A single cell may consist of several lines of typing. When the Enter key is used, Mathematica begins a new line in the same cell. To start a new cell for the next Input, place the cursor below the current cell and then type your Input. A horizontal line across the Notebook indicates that the cursor is at the top of a cell. It is sometime easier to move to a new line using the up and down arrows on the keyboard.

After you have completed typing an Input, you Evaluate the Input (i.e., make Mathematica execute the mathematical command you entered) by hitting Shift and Enter (carriage return). You can evaluate an entire notebook all at once by choosing Evaluation and then Evaluate Notebook from the Kernel menu. You can also evaluate cells by selecting the brackets at the right and typing Shift and Enter. Typing Enter without the Shift key starts a new line in the current cell.

Normally, the default Style is Input. That is, when you begin to type information into a cell, Mathematica is assumes that you are entering Input. Input is used to enter mathematical equations or Mathematica commands. Mathematica’s response from Evaluating a cell containing Input will be in Output Style. Once a cell has been evaluated, the Input is designated with In[number] = and the Output response by Out[number] =. The number indicates the sequential order of the Evaluations that Mathematica has performed (see discussion under the heading The Mathematica Kernel below). If you want to add comments, you must change the Style of the cell
to *Text* (or to various types of titles). To change the *Style* of a cell, click on the cell bracket, then click on the *Style* command in the *Format* menu. To add text, choose *Text*. You can also choose the *Style* for a cell prior to entering information to a cell. If you want to add a title, change the format of the cell to *Title*. You can put comments within an *Input* line by using the syntax (* put your comments here *).

Mathematica automatically groups cells. For example, the *Output[number]* from evaluating an *Input[number]* is automatically grouped with its *Input[number]*. If you double-click the right-most bracket for this *Input-Output* group, then only the contents of the *Input* cell will show. You can manually group cells together by choosing *Group Cells* and then *Manual Grouping* in the *Cell* menu. Then, after selecting (click left button and drag) the cells that you wish to group, use the command *Cell Grouping* and *Group Cells* in the *Cell* menu. If you double-click a group bracket at the far right of the screen, the group will condense and only the contents of the first cell will show. Note that the right bracket of a condensed group looks different than for cells that are not condensed. If you double-click the bracket again, all contents of all cells in the group will be reappear. To return to *Automatic Grouping*, select *Cell Grouping* and the *Automatic Grouping* in the *Cell* menu.

**Exercise 1.** Examine the Mathematica logistics just described. After starting Mathematica, type \( x = 1 \) into the notebook. Now press *Shift-Enter*. Mathematica should respond by adding an *In[1] =* to your \( x = 1 \) *Input*, followed by *Out[1] = 1*. There will be a horizontal line across the page indicating that your cursor has moved to the top of a new cell. Now type \( y = 2 \) and evaluate this. Next type \( x + y \), and evaluate this. You should now see *In[3] = x+y* and *Out[3] = 3* in response. Note that this was the 3\(^{rd}\) cell that Mathematica evaluated. On the right hand side, you will see several brackets. Each cell will be designated by the left-most bracket. The next bracket to the right will group the *Input* and *Output* cells. Double-click this bracket and see the *Output* cell disappear. Note that the shape of the bracket has changed indicating a collapsed group. Double-click the bracket again, and the *Output* cell should reappear. Change the grouping to *Manual Grouping* by choosing the *Cell* menu and the *Cell Grouping/Manual Grouping*. Now, group the *Input-Output* cells from *Input* 2 and 3 together by selecting their brackets and grouping through the *Cell* menu and selecting *Cell Grouping* and *Group Cells* commands. Note that a new bracket appears to indicate the new grouping. Double-click and see the response and then double-click again.
Move the cursor to the bottom of the cells. A horizontal line will appear below the previous Input-Output cells. Select Style from the Format menu. Choose Text and then type in a comment. Select the right bracket for this cell and then choose other styles and see what happens.

Mathematica is a powerful tool that can help you solve many types of math problems. Once you have developed a method on Mathematica to solve a given problem, it is easy to modify it for a different situation. So, just like you would for a FORTRAN or Basic program, you should add comments and try to make your program easy to follow. For Mathematica calculations that will be submitted as part of homework or projects, comments describing the Mathematica computations are essential.

If the command you are typing is too long to fit on one line, you can continue typing on a new line using the Enter key with no Shift. However, you must end the line in a way that Mathematica recognizes that more follows. Choose to start typing on the next line when you have just entered an addition symbol or a comma, for example. Then Mathematica will automatically go to the next line looking for what comes next.

Mathematica sometimes produces output that is long and unnecessary. For example, when you enter a long function, Mathematica will automatically print it again in the output. To suppress output, end the input line with a semicolon (;). (Try this out on the commands you entered in Exercise 1).

If the equations or commands are too long to print on a single line, Mathematica will automatically print on two pages, and one character in between the two pages may be deleted in the printout.

**The Mathematica Kernel**

It is helpful to understand that the Mathematica program consists of two parts: the user interface and the Kernel. The user interface simplifies the process of entering mathematical equations and translates the output into easier to read symbols. The Kernel is the part of Mathematica that performs computations. When you start a Mathematica session, only the user interface is running. The Kernel will be launched the first time you ask Mathematica to evaluate an Input cell. Consequently, it usually takes Mathematica some time to complete the first evaluation since the Kernel must be started before the evaluation can occur. After the Kernel is launched, all further cell evaluations will be faster since the Kernel is already started.

The Kernel remembers each evaluation it has performed and the order in which these were performed. This ordering is indicated by the sequential numbering that Mathematica assigns
to each evaluated Input and the resulting Output. This ordering can be important if you have used the same variable names to designate different calculations. Occasionally, you may have performed an evaluation earlier that is interfering with a subsequent evaluation. When this happens, it is sometimes helpful to use the Quit Kernel command from the Kernel menu and to restart the evaluation process. You can then either use the Start Kernel command (also on the Kernel menu) or just select a cell to evaluate and then type Shift and Enter (which will launch the Kernel). When you have Quit Kernel and then started again you will need to evaluate cells as if they have not been previously evaluated.

**Basic Rules of Mathematica Syntax**

1. The arguments of functions are given in brackets [...] ; parenthesis (...) are used for grouping operations; vectors, matrices, and lists are given in braces {...} ; and double square brackets [[...]] are used for indexing lists and tables.

2. Mathematica is case sensitive. Thus, cos[x] is not the same as Cos[x]. Be careful!

3. Mathematica has several built-in functions (e.g., Cos[x], Sin[x], Exp[x], etc.). The first letter of built-in functions always have the first letter capitalized, and if a name consists of two or more words, the first letter of each word is capitalized and there will be no space between the words (e.g., LaplaceTransform[x]).

4. The palette BasicInput.nb contains symbols for common mathematical manipulations like integration, differentiation, summation, division, and exponentiation. Greek symbols are also included in this notebook. The BasicInput.nb palette is usually opened when Mathematica is started. However, it is sometimes hidden behind the open Notebook. If it is open and hidden, it can be found through the Window menu. The top of the Window menu lists commands for organizing the window. The bottom of the Window menu lists all windows that are currently active. You may see Untitled_1.nb, Messages, in addition to BasicInput.nb. If BasicInput.nb appears, select it and the palette should appear. If it is not listed, then it can be opened from the Palettes command in the File menu. (Note that there are other palettes available in addition to BasicInput.nb that you may find useful as well.) The mathematical and Greek symbols in BasicInput.nb can be selected by clicking on the chosen symbol and then selecting each of the empty boxes and filling them in similar to the procedures used in the equation editor of a word processor. Mathematica understands the Greek letter π to be Pi (i.e., 3.1415 ...).

5. Mathematical manipulations can also be input through a set of standardized notation. Multiplication is represented by either a space or *. In Mathematica, a b means a*b, but ab
means you have a new variable with the name ab. Powers are denoted by a ^ or by using the symbolic representations in BasicInput.nb. Remember that Mathematica is case sensitive. A variable named Ab is not the same variable as ab.

6. If you get no response (e.g., Mathematica seems to be in an infinite loop) or an incorrect response, you may have entered or executed the command incorrectly. You can stop execution of a command by clicking Interrupt Evaluation under the Kernel menu.

7. If you want the output from an evaluated function to be displayed as a numerical value (e.g., in decimal format) rather than as a symbol or fraction, use the command N[...]. For example, N[Pi] or N[π] will return 3.14159. N[Pi,25] will return the number Pi to 25 digits. Typing //N after the input line will achieve the same results (e.g., Pi//N returns 3.14159).

8. If you wish to perform an operation on the last output that Mathematica produced, you can use the symbol %. For example, N[%] will result in the decimal representation of the last output. (For example, Input and then Evaluate Pi. Mathematica will respond with π. Now type N[%.] Mathematical will respond with 3.14159.) You can also refer to specific output using Out[put output number here], or %output number where output number is the number Mathematica has assigned to the cell when it was evaluated. For example, to generate the numerical value of the output designated as Out[17], the command is N[%17] or N[Out[17]]. Keep in mind that a specific output number will change if you re-evaluate an input line.

9. To delete a cell select the right bracket designating the cell and select the Clear command from the Edit menu. Other useful editing commands can be found in the Edit menu, including Cut, Copy, Copy as and Paste. Notice that the Preferences command appears in the Edit menu. You can use the Preferences command to set default preferences for various Mathematica operations. For example, to change the default values of fonts, you would choose formatting options/fonts after selecting Preferences. Other useful “editing” commands are available in the Input menu (Copy Input from Above and Copy Output from Above).

Help

To get on-line help from Mathematica regarding a command, enter ?Command. You can enter Options[Command] to find out what options are available with a command and what the default settings are. This is particularly helpful with plotting commands.

To find all the commands with the string Word in them, enter Names["*Word*"]. You can leave out the first or second asterisk to find the commands that end or begin with the string Word in them.
Occasionally, typing `?Command` will not provide information for Commands that are included through "Add-On" packages (i.e., not the standard Mathematica group of packages). To find information on these commands, choose Help from the Help menu. Select Add-ons, and then choose Standard Packages followed by the name of the package that contains the Command that you want help with. For example, for help with the \texttt{LaplaceTransform[x]} Command, select Add-ons, Standard Packages, Calculus, LaplaceTransform.

**Built-in Functions**

Mathematica has many built-in functions including:

- Trigonometric functions: \texttt{Sin[x]}, \texttt{Cos[x]}, \texttt{Tan[x]}, \texttt{Cot[x]}, \texttt{Sec[x]}, \texttt{Csc[x]}
- Inverse trigonometric functions: \texttt{ArcSin[x]}, \texttt{ArcCos[x]}, \texttt{ArcTan[x]}, \texttt{ArcCot[x]}, \texttt{ArcSec[x]}, \texttt{ArcCsc[x]}
- Hyperbolic functions: \texttt{Sinh[x]}, \texttt{Cosh[x]}, \texttt{Tanh[x]}, \texttt{Coth[x]}, \texttt{Sech[x]}, \texttt{Csch[x]}
- Inverse hyperbolic functions: \texttt{ArcSinh[x]}, \texttt{ArcCosh[x]}, \texttt{ArcTanh[x]}, \texttt{ArcCoth[x]}, \texttt{ArcSech[x]}, \texttt{ArcCsch[x]}
- Error function: \texttt{Erf[x]}
- Complimentary error function: \texttt{Erfc[x]}
- Bessel functions: \texttt{BesselJ[n,x]}, \texttt{BesselY[n,x]}
- Modified Bessel Functions: \texttt{BesselI[n,x]}, \texttt{BesselK[n,x]}
- Gamma function: \texttt{Gamma[x]}
- Legendre functions: \texttt{LegendreP[n,x]}, \texttt{LegendreQ[n,x]}
- Absolute value: \texttt{Abs[x]}
- Natural Log: \texttt{Log[x]} (not the base 10 log)
- Base 10 Log: \texttt{Log[x,10]}
- Exponential: \texttt{Exp[x]}

Mathematica can also work with complex numbers. In Mathematica \( I = \sqrt{-1} \).

**Add-On Packages**

Not all of Mathematica’s capabilities (e.g., Built-In Functions) are routinely available. Some capabilities are stored in packages that are not part of the default capabilities. To gain access to these capabilities, you must open the needed add-on package. It may not always be
clear that an add-on package is needed. In some cases, the basic version of Mathematica (i.e., without loading an additional package) can perform simple operations (e.g., limits of simple functions), but the package is needed to perform more complicated operations (e.g., limits of more complicated functions). In other cases, the package must be loaded to perform certain commands (e.g., the Package named DiracDelta must be loaded to perform the UnitStep command). If Mathematica is having a hard time doing something you think it should be able to do, you may need to load a package. You can view the available packages from the Help option in the Help menu. Select Add-ons and then Standard Package. The list of available add-on Packages are displayed in the third column. The Packages are grouped into the Directories displayed in the second column (e.g., Algebra, Calculus, DiscreteMath ...). For example, the Packages grouped in the Calculus Directory include DsolveIntegrals, DiracDelta, FourierTransform, LaplaceTransform, and so on.

Each Package will provide additional Built-In Functions to Mathematica. To find more information on the Built-In Functions available through a particular Add-On Package, select the package name in the third column and Mathematica will provide a useful description of the added capabilities. Packages that might be useful include (listed as Package`Command`):

- Calculus`DSolve` For solving differential equations
- Calculus`DiracDelta` For representing dirac delta (DiracDelta) and unit step (UnitStep) functions
- Calculus`Limit` For calculating limits
- Algebra`SymbolicSum` For calculating summations of more complicated sums
- Calculus`LaplaceTransform` Required to take a Laplace Transform or to inverse a Laplace Transform
- Calculus`FourierTransform` Required to take a Fourier Transform or to inverse a Fourier Transform

Two different commands can be used to load a package. The first is \<<Directory`Package\> and the second is Needs["Directory`Package"]]. In both cases, there are no spaces between the Directory name and the Package name. The apostrophe typed between the Directory name and Package name is located above the Tab key on most keyboards. Also note that Package names consisting of two or more words (e.g., SymbolicSum) are written with the first letter of each word capitalized and with no space between words. For example, the Package called LaplaceTransform, located in the Directory called Calculus is added using the following command:

\<<Calculus`LaplaceTransform\>
If you try to open a package twice using the << command, it may cause Mathematica to freeze. The Needs command only opens the package if it is not already open. For this reason, the Needs command may be a better approach. (In Version 3 of Mathematica, this problem appears to be fixed.)

A word of caution about loading packages. If a package is loaded after you have attempted to use a command that requires that package, Mathematica will not recognize the command as a built-in operation. For example, if the Package DiracDelta was not loaded before you tried to evaluate the command UnitStep[x], Mathematica will not recognize UnitStep[x] as a command. It will think that UnitStep[x] was a user-defined function. If the Package DiracDelta is then added, Mathematica will still think that UnitStep[x] is a user-defined function and it will issue a warning that UnitStep[x] has multiple definitions (i.e., as part of the DiracDelta package and as a user-defined function). A typical warning message is “Symbol UnitStep appears in multiple contexts {Calculus`DiracDelta`, Global`}; definitions in context Calculus`DiracDelta` may shadow or be shadowed by other definitions.” The solution is to (1) choose the Quit Kernel option from the Kernel menu, (2) enter the command to load the package, and (3) re-evaluate the Command that required the package.

Alternatively, if you know that you have evaluated a command that requires a package before the package has been added, you can remove the command by evaluating the Input Remove[command name] (e.g., Remove[UnitStep]). Then you can load the Package without conflicts. However, this procedure will only work if you Remove the command before you have loaded the Package. Once you have loaded the Package, you will have to use the Quit Kernel approach.

User-Defined Functions

When naming functions, it is a good idea to use all lower cases to avoid potential conflicts with names of Built-In Functions. (Do not forget that Mathematica is case sensitive.) Clear[function name] clears all definitions of a function. (Note however, that it does not remove the list of User-Defined Functions. To remove a function from the list of User-Defined Functions requires the command Remove[function name] as described in the section on Add-On Packages.) Since definitions of functions are frequently modified, perhaps several times in a Mathematica session, it is good practice to perform the Clear command, even when you do not think it is necessary.

The = sign is used for naming functions. There are two ways to name a function. If you are defining a function of x, you can use the commands: f[x_]= expression or f[x_]:= expression. The underscore after the independent variable name is mandatory. f[x_]:=
expression should be used when \( f(x) \) does not make any sense unless \( x \) is a specific variable. The : delays the evaluation of the function until it is needed. In general, if the command \( f[x_] := \) expression produces one or more errors, use \( f[x_] := \) expression.

There are several ways to evaluate a function at a particular value of the independent variable. The first is to type \( f[value] \). The output will be the function evaluated at the value you chose. The next is to enter \( f[x] /. \ x -> value \). The \( f[x] \) refers to the function, /. means “replace with”, and \( x -> value \) instructs the computer to put the value in wherever there is an \( x \) in the function. If you are replacing more than one variable, you need braces around the list (i.e. \( f[a,b] /. \{ a -> value, b -> value2, \} \)).

Piecewise functions can be entered into Mathematica by different methods. For example, if your function was: \( y(x) = 3x^2 \) for \( x < 0 \) and \( y(x) = 0 \) for \( x \geq 0 \), one way to represent the function in Mathematica is:

\[
\begin{align*}
y[x_] & := 3 x^2 /; x < 0 \\
y[x_] & := 1/2 /; x \geq 0
\end{align*}
\]

The symbol := needs to be used because the function does not make sense unless \( x \) is a particular number. Another approach is to use the UnitStep[x-a] command. You will remember that the UnitStep[x-a] is zero if \( x < a \) and is one if \( x > a \). The example described above can then be represented by the following equation:

\[
y[x_] := 3 x^2 (1 - \text{UnitStep}[x]) + (1/2) \text{UnitStep}[x]
\]

Remember that the Package DiracDelta must be loaded for Mathematica to recognize the UnitStep function.

**Exercise 2.** Refer to Number 8 on page 5 for tips on easy reference to output. Add \( a + b \). Multiply your output by \( c \). Add your first output to your second output. Divide your last output by seven. From the last output, replace \( a, b, \) and \( c \) with 2, 6, and 7 respectively. Get the decimal representation of the resulting fraction. (Make sure the numbers following the \( \% \) signs and used in the Out command refer to the correct output!)

| a+b | \( \% c \) | \% number of the a+b Output + \% number of the \( \% c \) Output | \( \% c \) / 7 |
Exercise 3. Clear the function g, enter the function \( g(x,y) = \frac{x^2 + 4xy - 3y^3}{5} \), then evaluate the function two different ways at \( x=2, y=7 \). Mathematica will output a fractional result. Force Mathematica to provide a decimal value by adding \(/N\) to the end of the command to evaluate \( g(x,y) \).

Exercise 4. Clear the function g, enter the function \( g(x) = 3x^2 \) for \( x < 0 \) and \( g(x) = 0 \) for \( x \geq 0 \). Evaluate the function at \( x = -1 \) and at \( x = 1 \).

Plotting

The basic command to plot functions is \( \text{Plot}[f[x],\{x,a,b\}] \) where \( a,b \) is the range of values you want displayed on the x-axis. Mathematica will plot symbolic functions. For this reason, Mathematica can be used to quickly provide information on the shape of a function. (This is very useful for quickly viewing whether the equation you derived in your homework solution is reasonable.)

It is often convenient to name plots (e.g., \( p1 = \text{Plot}[f[x],\{x,a,b\}] \)) gives this plot the name \( p1 \). Then, later, you can use the \( \text{Show} \) command (e.g. \( \text{Show}[p1,p2,p3] \) where plots \( p2 \) and \( p3 \) have been defined by similar commands) to show several plots on the same axis. Also, this allows the user to redefine plotting options (e.g., the axis length, titles etc.) without re-typing the \( \text{Plot} \) command. Several options that can be used with \( \text{Plot} \) or \( \text{Show} \) are listed here for quick reference (see a Mathematica reference book or the Help window for a more comprehensive list):

- \( \text{Frame}\rightarrow\text{True} \) draws a line around the plot (default is false)
- \( \text{AxesLabel}\rightarrow\{\text{“x-axis label”},\text{“y-axis label”}\} \) labels the axes of a plot (the default is no axes labels)
FrameLabel->{“bottom (x-axis) label”, “left (y-axis) label”, “top label”, “right (secondary y-axis) label”} labels the frame of a plot (the default is no frame labels). The last two can just be left out if they are not needed.

PlotLabel->{“name”} places a name above the plot (the default is no plot title)

PlotRange->{y-minimum,y-maximum} changes the range of the y-axis

PlotStyle->{Curve 1 Style}, {Curve 2 Style}, etc. changes the line styles of curves. An example is: PlotStyle->{Thickness[0.02], GrayLevel[0.5], Dashing[{0.05,0.05}]} for a thick, gray, dashed line. This command needs to be used with the Plot command.

Exercise 5. To see how to use the plot and show commands, enter the following example:

\[ g[x_] = \sin(x \pi / 4) \]

\[ p1 = \text{Plot}[\cos(x \pi / 4), \{x, 0, 4 \pi \}, \text{PlotStyle} \rightarrow \{\text{Thickness}[0.02], \text{GrayLevel}[0.5], \text{Dashing}[\{0.05,0.05\}]\}] \]

\[ p2 = \text{Plot}[g[x], \{x, 0, 4 \pi \}] \]

\[ \text{Show}[p1, p2, \text{Frame} \rightarrow \text{True}, \text{FrameLabel} \rightarrow \{\text{x}, \text{g(x)}\}, \text{PlotLabel} \rightarrow \text{“Example 5”}] \]

Note the use of the double quotes to identify a string as text for a label. If you type g(x) without the double quotes, Mathemtica will think that you are saying g multiplied by x; try it and see. The format of the Dashing command is \{line length, spacing length\}. Try putting in other values. Also, try changing the Thickness or GrayLevel.

The command /. (replace with) is useful for plotting. Suppose you want to see the function \( y = 4 x^b + a \) for several different values of a and b. The command could look like:

\[ y[x_] = 4 x^b + a \]

\[ \text{Plot}[[y[x]/\{a \rightarrow 1, b \rightarrow 1\}, y[x]/\{a \rightarrow 2, b \rightarrow 2\}], \{x, -2, 2\}] \]

To label the curves p1 and p2 while plotting them together, you can use the command

\[ p4 = \text{Show}[p1, p2, \text{Graphics}[\text{Text}[	ext{“text for p1”}, \{x\_\text{spot1}, y\_\text{spot1}\}]]],  \]
\[ \text{Graphics}[\text{Text}[	ext{“text for p2”}, \{x\_\text{spot2}, y\_\text{spot2}\}]]]. \]
The text for p1 will be centered at the location (x_spot1, y_spot1) based on the coordinates of the graph.

Mathematica plots can be enlarged by clicking them with the left button on the mouse, clicking and holding with the left mouse button on any corner, and then dragging the corner until the plot is the desired size.

**Exercise 6:** Plot the function, \( y = x^3 - 4 \) for \( x \leq 0 \) and \( y = x^2 \) for \( x > 0 \), over the interval of \(-1 \leq x \leq 1\). Give the plot the title “Problem 2”, name the axes x and y, and make the curve thick, gray, and dashed. (Note: There are two ways to do this problem. You can define y as a piecewise function, or you can instruct Mathematica to plot each function separately over different ranges and then show the functions on the same plot). What is the difference between **FrameLabel** and **AxesLabel**?

**Exercise 7.** Knowing the shape of some of these functions will be useful, so try plotting them. Plot the zero and first-order Bessel functions of the first kind (i.e., \( J_0(x) \) and \( J_1(x) \)) for \( 0 < x < 30 \).

\[
p1 = \text{Plot}[\text{BesselJ}[0,x],\{x,0,30\}] \\
p2 = \text{Plot}[\text{BesselJ}[1,x],\{x,0,30\}] \\
\text{Show}[p1,p2]
\]

**Equations**

To define an equation, use two = signs, i.e. **left hand side = = right hand side**. Equations are not functions. They are equations you might be interested in solving (i.e., finding the values of one or more of the variables to satisfy the equality).

**Analytical Solutions (symbolic function generated)**

The command **Solve** will find exact solutions to linear systems of equations. The command is: \( \text{Solve}[[\text{equation1}(x,y) = = \text{equation2}(x,y), \text{equation3}(x,y) = = \text{equation4}(x,y)],\{x,y\}] \) where the equations are functions of \( x \) and \( y \). Notice that the brackets are needed around the equations to be solved, because there are more than one.
**Numerical Solutions (number value generated)**

The main commands to solve equations numerically are:

- **NRoots[polynomial1(x) = polynomial(x), x]** will find the roots of a polynomial, even if there are more than one root.

- **FindRoot[equation(x),{x, initial guess}]** will work for all equations written in the form that \(\text{equation}(x) = 0\) (note, that the zero is not included in the command), provided that the initial guess is good enough. (Good enough means that \textbf{FindRoot} works; not good enough means that \textbf{FindRoot} does not work and you must try to make a better initial guess.) In choosing an initial guess it is often useful to examine a plot of the function. When only one initial guess is given, \textbf{FindRoot} uses Newton’s method. If you give a low and high range to the initial guess (i.e., \textbf{FindRoot[equation[x],{x, low guess, high guess}]}, \textbf{FindRoot} will use the secant method instead of Newton’s method. Usually, the secant method is more stable than Newton’s method (i.e., the secant method may be less sensitive to poor initial guesses).

- **NSolve[function1(x) = function2(x), x]** will also numerically solve an equation.

**Exercise 8:** Solve for the x-coordinates where the line \(y = 4x + 7\) and the parabola \(y = 17x^2 - 0.5\) intersect. Get both the numerical and the analytical solution, and compare the two.

When finding roots to an equation, initial guesses are required. When equations have multiple roots it is necessary to find multiple initial guesses. The \textbf{IntervalBisection} command provides two values, one below and one above a root of an equation. If the equation has more than one root, then additional pairs of values are generated. The \textbf{IntervalBisection} command requires that the \texttt{Package NumericalMath`IntervalRoots` be loaded.} The syntax is

\[
\text{IntervalBisection[equation of x, x, Interval[\{small x, large x\}], tolerance, MaxRecursion -> value]}
\]

where pairs of values will be generated for all roots over the interval from small \(x\) to large \(x\) within the designated tolerance. Do not make the tolerance too tight. It may not be necessary to specify the maximum level of recursion allowed If the tolerance value is large enough, the default level of recursion may be adequate. In this case, it will not be necessary to include information for
MaxRecursion. More help is available on Mathematica Help (select Add-Ins, Standard Packages, NumericalMath, IntervalRoots.

**Exercise 9.** Find the roots over an interval [2,8] of $\sin(x)$ and stop recursing when the subintervals have decreased to less than a 0.1.

$$\text{IntervalBisection}[\sin(x), x, \text{Interval}[[2,8]],0.1]$$

Find the roots over an interval [2,8] of $\sin(x)$ and stop recursing when the subintervals have decreased to less than 0.01. This will cause the maximum level of recursion to be exceeded. $MaxRecursion$ can be increased to achieve the closer tolerance as follows:

$$\text{IntervalBisection}[\sin(x), x, \text{Interval}[[2,8]],0.01, \text{MaxRecursion} \to 10]$$

**Exercise 10.** Determine roots of the equation $x^3 + 2x^2 - x - 2 = 0$ using the Solve, Nsolve, Roots, FindRoot, NRoots commands. It is sometimes helpful to plot the function first, to get an idea of where the roots are located. Use the command IntervalBisection to determine initial guesses.

**Lists**

Lists are used frequently in Mathematica, as both input and output. To manipulate a list, it is helpful to first name the list.

**Exercise 11.** First, generate two lists:

- list1={0,1,2,3}
- list2={5,6,7,8}

Create a new list, list3, which contains element 1 from list1 (i.e., 0) and element 3 from list2 (i.e., 7) using the following command:

list3={list1[[1]],list2[[3]]}

Say list1 represents time and list2 represents concentration from an experiment. If you want to pair the data (i.e., you want each time value to be paired with the concentration determined at that time), then you need to transpose the lists:
list4 = Transpose[{list1,list2}]

should give you

{{0,5}, {1,6}, {2,7}, (3,8}}

ListPlot[list4,PlotStyle->PointSize[0.05]]

Note that the PointSize designates the size of the symbols designating the data.

A solution to a set of equations may be returned in the form: \{c1->f[x], c2->g[x]\}. This output is actually three separate, nested lists. The two sets of braces around the list mean that Mathematica sees \{c1->f[x], c2->g[x]\} as a list of one. The expression c1->f[x], c2->g[x] is a list of two. And both c1->f[x] and c2->g[x] are also lists of two. (The arrow is not considered in the numbering of the list, so c1 is item one and the function of x is item two.) Assuming you have named the solution list1, to extract the answer to c1 (i.e., f(x)), you must extract the first item of the first nested list, the first item from the second nested list, and the second item from the third nested list, the command is list1[1,1,2]. Alternatively, you can use the command c1/.list1.

If a list has extra braces, the command Flatten[listname] can be used to remove them.

**Summations**

Mathematica can evaluate infinite sums. It is frequently helpful if the Package Algebra`SymbolicSum` is loaded. (As you will see if you try to load this package, the newest version of Mathematica includes this package as a default.)

**Exercise 12.** Consider the sum:

\[
\frac{2}{\pi^2} \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2}
\]

This summation has the value 1/6. This can be determined as follows:

\[
\text{Sum}[-2/\text{Pi}^2*(-1)^n/n^2,{n,1,\text{Infinity}}]
\]

Note that \infty in Mathematica is represented by the word Infinity. The summation can also be evaluated by using the symbolic summation symbol in the BasicInput.nb toolbar. Try both ways.
**Exercise 13.** Next, consider the sum:

\[-\frac{2}{\pi^2} \sum_{n=1}^{\infty} \frac{(-1)^n}{n^2} \exp(-n^2 \pi^2 t)\]

Enter this summation into Mathematica by using the mathematical symbols available in the BasicInput.nb palette, or by using the following command:

```
g[t_] = -2/Pi^2*Sum[(-1)^n/n^2*Exp[-n^2*Pi^2*t],{n,1,Infinity}]
```

Determine g(t) when t = 0.1. You may need to force Mathematica to respond with a numerical value. If you try to plot g(t) for t between 0 and 1, Mathematica will either take a very long time or simply fail (i.e., it goes off to calculate, but never responds). If this happens to you, you will need to use the *Quit Kernel* command in the *Kernel* menu.

To plot an infinite sum, you need to represent it with a finite number of terms. That is, you need to truncate terms that do not contribute significantly. Thus, infinity is replaced by a finite value of \(n\) that is large enough that the solution has converged. For this example \(n = 20\) is close enough to "infinity". Plot the resulting values.

```
g[t_] = -2/Pi^2*Sum[(-1)^n/n^2*Exp[-n^2*Pi^2*t],{n,1,20}];
Plot[g[t],{t,0,1}]
```

If you did not include the semi-colon at the end of the input defining \(g[t]\), Mathematica would print out 20 terms with the numerical values corresponding to \(n = 1, 20\). You could also use mathematical symbols to represent \(g(t)\) with the upper limit of the sum set to 20. Notice that Mathematica chooses what values of \(g(t)\) to plot and it may choose to plot \(g(t)\) in such a way that the value of \(g\) at \(t = 0\) is not plotted. If this happens, you can specify the range of values for \(g(t)\) that you want Mathematica to plot using the *PlotRange* command. For example,

```
Plot[g[t],{t,0,1}, PlotRange{0,0.2}]
```

Examine the effect of changing the number of terms in the calculation for small and large values of time (e.g., \(t = 0.001, 0.1,\) and \(1\)). Calculate the numerical value of the sum for \(n = 1, 2,\)
5, 10, and 20 (or until increasing the number of terms further has no effect). To make this easier you may want to re-define $g(t)$ as $g(t, n_{max})$ using the following command:

$$g[t\_, n_{max\_\_}] = -2/Pi^2*\text{Sum}[((-1)^n/n^2)*\text{Exp}[-n^2*Pi^2*t],\{n, 1, n_{max}\}]$$

Then you can evaluate $g(t)$ at $t = 0.001$ using 2 terms in the summation by entering:

$$g[0.001, 2]$$

**Algebra (Simplification of Answers)**

Sometimes when you use Mathematica to check a solution you reached by hand, the answer Mathematica comes up with looks nothing like your solution. In that case, you may need to change the form of your solution. Some of the basic commands that come in handy to simplify the Mathematica output are: **Factor[...]**, **Expand[...]**, **Together[...]**, **Apart[...]**, **Cancel[...]** and **Simplify[...]**. You can also simplify the numerator and denominator of a fraction separately by extracting just the top or bottom of a fraction. The commands for that are: **Numerator[...]** and **Denominator[...]**. **Factor[...]** factors the argument. **Cancel[...]** is used to reduce a ratio of functions (if possible) to a smaller number of terms. Simplifying solutions in Mathematica is a bit of an art form. You may have to try many different ways before you find the solution form you want!

**Exercise 14.** Define the variable fraction as

$$\text{fraction} = \frac{x^3 + 2x^2 - x - 2}{x^3 + x^2 - 4x - 4}$$

Extract the numerator of fraction and set it equal to the variable num.

$$\text{num} = \text{Numerator[fraction]}$$

Factor the numerator. Evaluate the numerator at $x = 2$.

$$\text{num}/.x->2$$

Extract the denominator of fraction and set it equal to the variable den. Factor the denominator and then evaluate the denominator at $x = 2$. Define newfraction as the ratio of the factored forms of the numerator and the denominator. Confirm that the algebraic form of fraction has not changed.
by evaluating the word fraction. Define a new variable, cancelfraction, as the result from 
\textbf{Cancel[fraction].} Next Expand cancelfraction. Next apply the \textbf{Together[...] command to the} 
result from \textbf{Expand[cancelfraction].} Finally, apply the \textbf{Apart[...] command to fraction to see} 
how Mathematica can be used to find partial fractions.
Practicing with these commands is the only way to get a feel for how they work.

\section*{Calculus}

Mathematica can do several calculus operations, including differentiation, integration, and 
taking limits.

\textit{Differentiation}

\begin{itemize}
  \item $f'[x]$ computes the derivative of $f(x)$ with respect to $x$
  \item $D[f[x],x]$ does the same
  \item $D[f[x],\{x,n\}]$ computes the $n$th derivative of $f(x)$ with respect to $x$
  \item $D[\text{expression, variable}]$ computes the derivative of expression with respect to variable. If 
    the expression is a function of more than one variable, this command will result in the 
    partial derivative of the expression.
  \item $D[\text{expression, \{variable,n\}}]$ computes the $n$th derivative of expression with respect to 
    variable
  \item $\partial x y$ (found on the BasicInput.nb Palette) computes the first derivative of $y$ with respect to 
    $x$.
  \item $\partial x \partial x y$ computes the second derivative of $y$ with respect to $x$.
\end{itemize}

\textbf{Exercise 15.} Take the derivative of the summation from Example 13 with respect to time.
Mathematica can take the derivative of an infinite sum like the function in Example 13. However, 
Mathematica may not be able to plot the result of the derivative of an infinite sum. If this is a 
problem, the function can be estimated by summing over a finite number of terms. For this 
function 20 terms is almost as good as infinity for typical values of time. Plot the derivative of the 
function and the function for $t$ between $[0,1]$. 

**Integration**

Mathematica can evaluate definite and indefinite integrals. It can also calculate numerical values for integrals exactly or numerically. The command for symbolic evaluation of integral is: \( \text{Integrate}[f[x],x] \) for an indefinite integral, \( \text{Integrate}[f[x],[x, \text{lower bound}, \text{upper bound}]] \) for a definite integral. The numerical command is \( \text{NIntegrate}[f[x],[x, \text{lower bound}, \text{upper bound}]] \).

**Exercise 16:** Integrate \( \frac{1}{x} \) with respect to \( x \).

Integrate \((x \sin[x])\) with respect to \( x \). Also, integration \((x \sin[x])\) for limits of \( x = 0 \) and \( x = \pi \).

Integrate \( \exp[-x^2] \) with respect to \( x \). This function cannot be evaluated exactly and Mathematica responds with a function named \( \text{Erf}[x] \) (error function) which is defined to be this integral integral expression.

Integrate \( \exp(-x^2) \) with respect to \( x \) over the interval from zero to infinity. This definite integral has a finite value, which Mathematica should recognize. Also, integrate over the interval \([0,1]\).

Numerically, integrate \((2/\sqrt{\pi}) \exp(-x^2)\) with respect to \( x \) over the interval \([0,1]\). Determine the numerical value of this result by finding \( \text{Erf}[1] \). (Remember that \( \log[e] = 1 \).)

**Limits**

Limits calculated by Mathematica can depend on the form of the input. Mathematica may tell you a limit is zero, or does not exist, even when a finite limit does exist. Be sure to check your answer and see if it is reliable. You may want to try several different forms for the input, and see if they return different answers.

The command is, \( \text{Limit}[\text{expression},x\rightarrow a] \). Remember, infinity is \textbf{Infinity} and negative infinity is \textbf{-Infinity}. You can also use the infinite symbol from the BasicInput.nb palette. Remember, there is a package called \textbf{Calculus`Limit} that contains special limit capabilities. These may required if the expression is not very simple. If there is a chance that the additional package capabilities may be required, it is best to load \textbf{Calculus`Limit} before trying to evaluate the limit of your expression. If Mathematica cannot properly evaluate the limit without the package, loading it after you try to find the limit may cause problems. If this occurs, select Quit.
*Kernel* from the *Kernel* menu, and then re-evaluate your limit after you have loaded
Calculus`Limit`.

**Exercise 17.** Calculate the limit of \( \frac{\sin x}{x} \) as \( x \) approaches zero.

## Differential Equations

Mathematica can be used to solve differential equations. Two examples/exercises are provided below. The first example illustrates the solution to a second-order, constant coefficient, ordinary differential equation. The second example illustrates the simultaneous solution of two ordinary differential equations. Unless instructed otherwise, students should only work Exercise 18.

**Exercise 18.** Solve the ordinary differential equation

\[
y'' - y' = 0 \quad \text{for} \quad y(1) = 1 \quad y'(0) = 0
\]

Use the command

\[
\text{newsoln = DSolve}\{y''[x] + y[x] == 0, y'[0] == 0, y[1] == 1\}, y[x], x\}
\]

Extract \( y \) (that is, the functional expression for \( y(x) \)) from the resulting output using the following command and assign it to the function \( y[x] \):

\[
y[x_] = y[x] / . \text{newsoln}
\]

Create a plot of \( y \) as a function of \( x \) for \( x = 0 \) to \( 1 \).

Mathematica can find the exact solution to systems of differential equations. Here is an example to illustrate the use of the basic command. Consider a model that treats the human body as two well-stirred tanks in series as illustrated below. This type of representation is called a pharmacokinetic model.
Assume that the concentrations of chemical in both stirred tanks are initially zero. At time zero, a mass of chemical called siv (small amount delivered by intravenous injection) is introduced into the first tank. From the first tank, the chemical can go into the second tank or be eliminated (i.e., the kidneys are working). From the second tank, the chemical can only go back to the first tank. All of the transfers between or from the tanks are assumed to be described by a first-order rate constants. Based on mass balance in each tank individually, the following system of two equations apply:

\[ V_1 \frac{dC_1}{dt} = -k_{el} C_1 V_1 - k_{12} C_1 V_1 + k_{21} C_2 V_2 \]

\[ V_2 \frac{dC_2}{dt} = k_{12} C_1 V_1 - k_{21} C_2 V_2 \]

\[ C_1(0) = \frac{siv}{V_1} \quad \text{and} \quad C_2(0) = 0. \]

**Alternate Exercise 18.** Solve this system of ordinary differential equations using the command:

\[
\text{Soln1}=\text{DSolve}\left\{v1*c1'[t] = -1*c1[t]*(kel + k12)*v1+k21*c2[t]*v2, \quad v2*c2'[t] = = k12*c1[t]*v1-k21*c2[t]*v2, \quad c1[0] = = siv/v1, \quad c2[0] = = 0,\{c1[t],c2[t]\},t\right\}
\]

Simplify the result further by typing:

\[
\text{Simplify}\left[\%\right]
\]

Now, remove extra braces and replace the rate constants with numerical values:

\[
\text{Soln2}=\text{Flatten}\left[\text{Soln1}/.\{v1->1,v2->2,\text{kel}->1,k12->1,k21->.1,siv->1\}\right]
\]

Extract \( c1 \) (that is, the functional expression for the concentration in tank 1) from the list and assign it to the function \( c1[t] \). This can be done in two ways:

\[
c1[t_\_]=c1[t]/\text{Soln2}
\]

or
c1[t_] = Soln2[[1, 2]]

Note that this last command specifies that the second piece of the first element should be extracted. This operation may be needed if you wish to then create a plot of c1 as a function of time.

Create a plot of c1[t] for t = 0 to 1.

Using Mathematica in Assignments

There will be many times when you will find it convenient to use Mathematica to perform operations (e.g., plotting) or to check your work (e.g., check algebra, integration or differentiation or limits). Never turn in output that is not clearly labeled. This can be done by entering comments describing the operation you have performed using the appropriate text format, or you can label the operation by hand. Whichever approach you choose to use, please include enough information so that anyone looking at the output knows what he or she is looking at. All plots should have titles describing exactly what is plotted, labeled axes and labeled curves. Remember that it is not always easy to follow a list of Mathematica commands that someone else has assembled. If you have used Mathematica to perform a mathematical operation, please include the Mathematica output as proof of the result.