Chemistry 20262 - Mathematical Methods Spring 2018
Professor J. Daniel Gezelter

In-class Practical 1
Introduction to Mathematica

Today we'll be getting used to a very powerful symbolic mathematics program called Mathematica. You'll be using it to do very difficult integrals, derivatives, and linear algebra later in the semester (and next year), but today we'll just be exploring how to interact with it.

Mathematica's User Interface

1. First, make sure you have downloaded a copy of Mathematica to your laptop. To get the software, go to:

   `https://oit.nd.edu/services/software/software-downloads/mathematica-for-students/`

2. Mathematica documents are often called "electronic notebooks" and consist of a sequence of pairs of input and output cells. A cell is where you enter a Mathematica command or receive output from the program.

   Open the Mathematica program on your computer. You should see an empty notebook without any cells. Click the mouse somewhere on the notebook to activate the cursor. Then type

   ```
   2 + 2
   ```

   and press the **Enter** key on the numeric keypad. If you have no numeric keypad, you can use **Shift-Return** in place of **Enter**. You should see a bracket appear at the right edge of the notebook window. This bracket defines the extent of the input cell you just created. Throughout this exercise the indented lines represent Mathematica input and output. Boldface lines will represent the input you type into a Mathematica notebook. Mathematica automatically puts your output into boldface, so your Mathematica notebook should look very similar to the indented lines in this tutorial.

   Pressing the **Enter** (or **Shift-Return**) key instructs Mathematica to evaluate the input cell that contains the cursor. (Make sure the cursor is in an input cell, or else Mathematica won't do anything when you press **Enter**.) In this case, we are asking Mathematica to evaluate the expression 2 + 2. After some time, an output cell will appear in your notebook with the result:

   ```
   4
   ```

   Notice that the input and output cells are labeled "`In[1]`" and "`Out[1]`". Each has its own bracket, and they are grouped together by an outer bracket.

   It may seem that it took Mathematica a long time to perform this simple calculation. This is because Mathematica loads into your computer in two stages. When you open the Mathematica icon, the user interface portion of the program is loaded. The part of the program that actually

does calculations doesn't get loaded until the first time you press **Enter**.

You can use the mouse to copy text from a previous input cell and paste it into the current input cell. This is useful if you want to repeat a portion of an earlier calculation or if you need to correct a spelling mistake. Just make sure the cursor is in the current input cell before you press **Enter**.

Simple Mathematica Calculations

1. In your notebook, type

   ```
   (1 + 2) * 3
   ```

   and press **Enter** (or **Shift-Return**). You'll get the result:

   ```
   9
   ```

2. You can omit the multiplication sign, and Mathematica is smart enough to know that it is implied:

   ```
   (1 + 2) 3
   9
   ```

3. But Mathematica can do a lot more than simple integer math. Try

   ```
   102 / 9
   ```

   You get the result

   ```
   34 / 3
   ```

   You see that Mathematica reduced the fraction to its simplest form by canceling out common factors in the numerator and denominator. Also, notice that the result is still expressed as a fraction, not as a decimal number. This is because the fraction is more exact than any decimal approximation to it, and Mathematica tries to maintain as much accuracy as possible unless you specify otherwise.

4. To get a numerical approximation to this result, type

   ```
   N[102 / 9]
   ```

   You'll get the result

   ```
   11.3333
   ```

   The **N[...]** command tells Mathematica to evaluate the quantity in brackets **numerically**. If we want more decimal places, we can ask Mathematica to calculate the same thing to 20 digits:

```
N[102 / 9, 20]
11.333333333333333333
```

5. At this point, it's useful to introduce a Mathematica shortcut that lets us take the previous output cell and include it in the current input cell. Enter

   ```
   102 / 9
   ```

   and you'll get the standard Mathematica result:

   ```
   34 / 3
   ```

   Then enter in a new input cell

   ```
   N[%, 20]
   ```

   and you'll get the result

   ```
   11.333333333333333333
   ```

   just as before. The **%** symbol tells Mathematica to insert the output of the previous calculation.

6. Mathematica knows the basic mathematical functions like sine and cosine. Typically these functions have the name you would expect, but with their first letter capitalized. To take the sine of 34/3 radians, enter

   ```
   Sin[%]
   ```

   and you'll get the result

   ```
   -0.94349962701548489718
   ```

   Notice that Mathematica has kept 20 significant figures; it remembers that that was the accuracy of the previous output cell and passes that information along to the current cell.

7. If we try to compute the sine of 34/3 radians from scratch, we don't get the same result:

   ```
   Sin[102 / 9]
   Sin[34 / 3]
   ```

8. Here Mathematica has left both the fraction and the sine function unevaluated. To evaluate this numerically, we can use the **N[...]** command:

   ```
   N[%, 20]
   -0.94349962701548489718
   ```

Notice that Mathematica uses square brackets to delimit the argument of a function, instead of the more conventional parentheses. This is because parentheses are used exclusively for grouping in Mathematica, as in the calculation we entered at the beginning of this section:

```
(1 + 2) * 3
```

Complex numbers in Mathematica

1. Mathematica uses the capital letter I to represent the square root of -1. Type

   ```
   Sqrt[-1]
   ```

   and you'll get the answer

   i

2. You can use I in expressions: the complex number 2 + 3i is represented as

   ```
   2 + 3 I
   ```

   in Mathematica. The generic complex number $(x + yi)$ is written as

   ```
   x + y I
   ```

   or, equivalently,

   ```
   x + I y
   ```

3. Mathematica has a tendency to alphabetize things, so it will usually print out $(x + yi)$ in the second form.

4. Mathematica uses the function **Conjugate** to take the complex conjugate of a number. Try it:

   ```
   a = 2 + 3 I
   Conjugate[a]
   2 - 3 i
   ```

5. We know that the complex conjugate of $(x + yi)$ is $(x - yi)$. But Mathematica gives us

   ```
   Conjugate[x + y I]
   Conjugate[x] - i Conjugate[y]
   ```

   which is not particularly informative.

The problem is that we have not specified whether *x* and *y* are real or complex numbers, and Mathematica won't make any assumptions about *x* and *y* without our help. If *x* and *y* are themselves complex numbers, then the conjugate of $(x + yi)$ is not simply $(x - yi)$. To tell Mathematica that *x* and *y* are real numbers, use the **ComplexExpand** command:

```
ComplexExpand[ Conjugate[x + y I] ]
x - i y
```

Now we get the expected result. **ComplexExpand** is particularly helpful when working with complex functions like electron wave functions.

Plotting lists of (x, y) points

1. We've seen that Mathematica uses parentheses for grouping and square brackets in functions. Mathematica uses **curly braces** to delimit **lists** of numbers. For example, a list of the first ten prime numbers would be

   ```
   {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
   ```

   If you enter this as Mathematica input, you'll get the exact same list as output:

   ```
   {2, 3, 5, 7, 11, 13, 17, 19, 23, 29}
   ```

   This is because we haven't asked Mathematica to do anything to the list. Later on we'll see how we can use lists of numbers in calculations.

2. The reason we want to discuss lists now is that Mathematica represents (x, y) points as lists of two numbers. For example the point (x = 1, y = 3.7) is represented in Mathematica as

   ```
   {1, 3.7}
   ```

   Suppose we have a list of three (x, y) points: (1, 2), (2, 3), and (3, 5). These points are represented as lists of two numbers. The list of the three points is then a nested list, or a list of lists:

   ```
   { {1, 2}, {2, 3}, {3, 5} }
   ```

3. To plot a graph of these three points, we use a new Mathematica command:

   ```
   ListPlot[ { {1, 2}, {2, 3}, {3, 5} } ]
   ```

   This tells Mathematica to plot the list of points inside the square brackets. The spaces are optional and are included here mainly for the sake of clarity.

4. To plot ten points representing the first ten prime numbers, we would therefore type

```
ListPlot[ { {1, 2}, {2, 3}, {3, 5}, {4, 7},
          {5, 11}, {6, 13}, {7, 17}, {8, 19},
          {9, 23}, {10, 27} } ]
```

Again, the extra space is optional. If you type the input as it is shown here, however, you'll notice that your input cell grows large enough to accommodate all three lines of the command. If you forget a curly brace (or add an extra one), Mathematica will inform you of this. One of the points may be hard to see; look closely at the origin of the x and y axes to convince yourself that there is a point there.

5. Now let's join the points with straight line segments. Type

```
ListPlot[ { {1, 2}, {2, 3}, {3, 5}, {4, 7},
            {5, 11}, {6, 13}, {7, 17}, {8, 19},
            {9, 23}, {10, 27} }, PlotJoined -> True ]
```

The modifier

```
PlotJoined -> True
```

tells Mathematica to connect the points with lines.

6. Here's another shortcut that can save you a lot of typing. Suppose we wanted to store the list of the first ten primes in memory, so that we don't have to type it in every time we want to use it. Enter

```
primes = { {1, 2}, {2, 3}, {3, 5}, {4, 7},
           {5, 11}, {6, 13}, {7, 17}, {8, 19},
           {9, 23}, {10, 27} }
```

Mathematica will respond with

```
{ {1, 2}, {2, 3}, {3, 5}, {4, 7},
     {5, 11}, {6, 13}, {7, 17}, {8, 19},
     {9, 23}, {10, 27} }
```

7. This list is now stored in the variable “primes”. We can use this variable in a **ListPlot** command as follows:

```
ListPlot[primes, PlotJoined -> True]
```

It's good stylistic practice to give your own variables names that begin with lowercase letters. This prevents us from trying to use the name of a Mathematica command or function as a variable name. Mathematica commands are all **case specific**, which means that uppercase and lowercase letters are distinct.

If you just want to see what is stored in the variable primes, type

```
primes
```

and Mathematica will respond with

```
{ {1, 2}, {2, 3}, {3, 5}, {4, 7},
  {5, 11}, {6, 13}, {7, 17}, {8, 19},
  {9, 23}, {10, 27} }
```

Plotting functions

1. Suppose we want Mathematica to plot the function $y = x^2$ over the range of x values $0 < x < 10$. Enter

   ```
   Plot[x^2, {x, 0, 10}]
   ```

   and you'll get such a plot. (Note that Mathematica uses the **^** character to indicate raising a number to a power.)

2. The Plot command can be used to plot virtually any one-dimensional function:

   ```
   Plot[Sin[ x + 1/x ], {x, 0.5, 5}]
   ```

   Let's take this command apart to see how its components work together.

   The command takes two arguments, which are (in order) the function we want to plot and the range of x values we want to use:

   ```
   Plot[ <function>, <range> ]
   ```

   The range is expressed as a list with three elements: the first element is the variable that will be plotted on the horizontal axis; the second element is the lower limit on this variable; and the third element is the upper limit on this variable.

3. The fact that the independent variable is specified as part of the plotting range means that we can call the independent variable anything we like:

   ```
   Plot[Sin[time], {time, 0, 4 Pi}]
   ```

   (Note that Mathematica has memorized the value of pi. You just have to remember that the letter P is capitalized.)

Combining two or more plots

1. Mathematica lets you store plots in variables so that you can combine several individual plots in a composite figure. Enter

   ```
   splot = Plot[ Sin[x], {x, 0, 2 Pi} ]
   cplot = Plot[ Cos[x], {x, 0, 2 Pi} ]
   ```

   and you will get two individual plots of the sine and cosine function. To plot both functions on the

same set of axes, enter

```
Show[splot, cplot]
```

You can combine different types of plots in this fashion. For example, you might want to combine a plot of experimental data points with a plot of a curve that fits through these points. The data points could be plotted with a **ListPlot** command, and the curve with a **Plot** command. Then the **Show** command would combine these two plots.

2. Sometimes you will generate a plot, and after looking at it, decide that you want to save it in a variable so that you can combine it with another plot. To do this, you can use the **%** shortcut we already mentioned. For example,

```
Plot[ Sin[x], {x, 0, 2 Pi} ]
splot = %
```

will store the plotted sine curve in the variable **splot**.

3. The **Show** command will combine several plots at once:

```
splot = Plot[ Sin[x], {x, 0, 2 Pi} ]
cplot = Plot[ Cos[x], {x, 0, 2 Pi} ]
tplot = Plot[ Tan[x], {x, 0, 2 Pi} ]
Show[splot, cplot, tplot]
```

Or you can combine the plots in stages:

```
splot = Plot[ Sin[x], {x, 0, 2 Pi} ]
cplot = Plot[ Cos[x], {x, 0, 2 Pi} ]
sandc = Show[splot, cplot]
tplot = Plot[ Tan[x], {x, 0, 2 Pi} ]
Show[sandc, tplot]
```

4. When you combine two or more plots, you may want to adjust the limits of the x and y axes to focus attention on a particular region of the plot. The **PlotRange** modifier to the **Show** command lets you do this. Enter

```
Show[splot, cplot, tplot, PlotRange -> { {0, 10}, {-10, 10} }]
```

This tells Mathematica to extend the horizontal axis so that it includes the range $0 < x < 10$ and the vertical axis so that it includes the range $-10 < y < 10$. The general format for the **PlotRange** modifier is

```
PlotRange -> { { <Xmin>, <Xmax> }, { <Ymin>, <Ymax> } }
```

You can see that the range is specified as a nested list, but not as a list of two *(x, y)* points.

5. Note that when we changed the axis limits, Mathematica did not extend the plots across the entire x axis. If you want the curves to extend all the way to $x = 10$, you need to specify this in the original **Plot** commands:

```
splot = Plot[ Sin[x], {x, 0, 10} ]
cplot = Plot[ Cos[x], {x, 0, 10} ]
tplot = Plot[ Tan[x], {x, 0, 10} ]
Show[splot, cplot, tplot, PlotRange -> { {0, 10}, {-10, 10} }]
```

Commonly Used Functions

1. Here is a partial list of various functions that Mathematica has already defined for you. The list of mathematical functions it knows about is ***very*** large, and you'll meet more of them as the semester continues (and as you get into Physical Chemistry)

   Trigonometric functions:

   - **Sin[x]** gives the sine of x in radians.
   - **Cos[x]** gives the cosine of x in radians.
   - **Tan[x]** gives the tangent of x in radians.

   Exponentiation and logarithms:

   - **Exp[x]** gives e to the power x.
   - **Log[x]** gives the natural logarithm of x.

   Square roots:

   - **Sqrt[x]** gives the square root of x.

Derivatives and Integrals in Mathematica

1. Mathematica uses the command **D[...]** to take derivatives of variables and functions. As an example, type

   ```
   D[x^3, x]
   3 x^2
   ```

2. The general format of the **D[...]** command is

   ```
   D[ <function>, <variable> ]
   ```

   which tells Mathematica to take the derivative of <function> with respect to <variable>. Mathematica treats all derivatives as partial derivatives, so we have

   ```
   D[x y^2, x]
   y^2
   D[x y^2, y]
   2 x y
   ```

3. To take the second derivative, we can just use the **D[...]** command twice in a row:

```
D[ D[x^3, x], x ]
6 x
```

4. The command to compute integrals is **Integrate**. It works much like **D[...]** in that we specify an integrand and a variable of integration:

```
Integrate[x^2, x]
x^3 / 3
Integrate[x y, y]
x y^2 / 2
```

Note that Mathematica omits the constant of integration that is common in calculus textbooks.

5. We can also use **Integrate** to compute definite integrals by specifying a lower and upper limit of integration:

```
Integrate[x^2, {x, 1, 2}]
7 / 3
```

computes the integral of *x^2* from *x = 1* to *x = 2*. The special keyword **Infinity** can be used to specify a lower or upper limit of integration:

```
Integrate[(1/x)^2, {x, 1/2, Infinity}]
2
```

6. Some definite integrals are too hard for Mathematica to compute analytically. In this case, we can estimate the integral numerically using **NIntegrate**. (Note the double capital letter at the beginning of this command.) As an example, try

```
Integrate[Sin[x^4], {x, 0, 1}]
1/8 i (ExpIntegralE[3/4, -i] - ExpIntegralE[3/4, i]) + Gamma[5/4] Sin[π/8]
```

You see that when an integral is too hard for Mathematica, it simply spits out a related integral in a partially digested form. But we can compute the integral numerically:

```
NIntegrate[Sin[x^4], {x, 0, 1}]
0.18757
```

7. Another way to integrate numerically is to nest the **Integrate** command within the **N[...]** command:

```
N[ Integrate[Sin[x^4], {x, 0, 1}] ]
0.18757 + 0. i
```

This has the advantage that we can ask for more significant figures:

```
N[ Integrate[Sin[x^4], {x, 0, 1}], 20]
0.18756954468467107035 + 0.*10^-21 i
```

Homework Questions (to be turned in with the problem set on Friday)

1. Consider the Taylor series for sin(x): $\sin x = x - \frac{x^3}{6} + \frac{x^5}{120} - \cdots$

   If we just take the first two terms, we can get a feel for how good the Taylor series does at approximating the function:

```
taylorSinX = x - x^3 / 3!
tsplot = Plot[taylorSinX, {x, -2 Pi, 2 Pi}]
splot = Plot[Sin[x], {x, -2 Pi, 2 Pi}]
Show[tsplot, splot, PlotRange -> {{-2 Pi, 2 Pi}, {-2, 2}}]
```

   Plot sin(x) along with its truncated Taylor expansion. Make different plots for truncation after x^3, x^5, and x^7. You will want to adjust the range displayed so that you can see what's going on—where the Taylor expansion is a good fit to sin x, and where it starts to be a bad fit. Print out the plot for the last of these (all terms up to x^7), and hand it in with your problem set.

2. Find the complex conjugate, square modulus, absolute value and the argument of the following three complex numbers. Use these values to write these complex numbers in their polar forms:

   $-1-7i$
   $3+4i$
   $8-37i$

   Note that **`Conjugate[x], Abs[x]`** and **`Arg[x]`** are helpful commands here.