Getting started with *Mathematica* 8

The rest of this document is based on Wolfram's old "First five minutes with *Mathematica*" tutorial and still contains very useful information. It is available in PDF and *Mathematica* notebook form at http://facstaff.unca.edu/mcmcclur/-Mathematica/.

Getting Mathematica

Getting a copy of Mathematica is outrageously simple for UNCA students and faculty and the link above contains instructions. In a pinch, you can also get a disk from me. You should create an account using your UNCA email address; you can then download your own copy and use it right away. You'll need to go through the simple activation process, if you want to use it for more than two weeks.

Wolfram Resources

You absolutely *must* become adept at using the Documentation Center available under the Help menu. In addition, the Wolfram Learning Center is a great resource for learning to use *Mathematica*. I particularly recommend Cliff Hasting's Hands on Start to *Mathematica* video:

• http://www.wolfram.com/support/learn/

• http://www.wolfram.com/broadcast/screencasts/handsonstart/

Basic computations

The first thing we should do is try to add 2 and 2. You can do so by typing the following command *exactly as shown* and hitting ENTER, the enter key.

```
2 + 2
```

Please take the phrase "exactly as shown" seriously. Details such as capitalization, brackets [] versus braces {}, and the like tend to be important. Also, ENTER and ↵ (return) are different. On many systems, ENTER is equivalent to SHIFT-↵, shift-return.

The next computation is a bit more impressive.

```
500!
```

The result has over 1000 digits. You can compute the exact number of digits like so.

```
Length[IntegerDigits[%]]
```

Note that `%` refers to the previous output. Previous means most recently generated and might or might not refer to the output directly above. If you re-execute the last command, you should get 4.

Speaking of 1000 digits, here's 1000 digits of π.

```
N[Pi, 1000]
```

Note that there is a big difference between `Pi` and 3.14; one is an *exact* number, the other is a decimal approximation. There is even a big difference between 3 and 3.0. Usually, computations with decimal approximations will run much faster than exact computation. The following command indicates that they are treated differently in computations.

```
{Pi - 3, Pi - 3.0}
```

2D Graphics and lists

Next, let's plot a function.

```
Plot[Sin[x], {x, 0, 10}]
```

Most commands may be input in a variety of formats and have a variety of options. Next, we use a list to plot two functions and we use a couple of options to affect the appearance. Options are specified using rules with arrows: →, which can simply be entered as ->.

```
Plot[{Sin[x], Cos[x]}, {x, 0, 3 Pi},
 Filling → True, Ticks → {
   Table[x, {x, 0, 3 Pi, Pi / 4}], {-1, 1}}]
```

Note that we've set up lists in two different ways. The list of functions is enclosed in braces: this is the sole purpose of braces in *Mathematica*. The ticks on the *x*-axis are also represented as a list, but that list is generated using the **Table** command.

```
Table[x, {x, 0, 3 Pi, Pi / 4}]
```

More generally, the syntax is

```
Table[expr, {x, xMin, xMax, xStep}]
```

Natural language input

Suppose we don't remember or even know the exact **Plot** syntax just yet. Well, type "= plot sinx" and the following amazing thing happens:

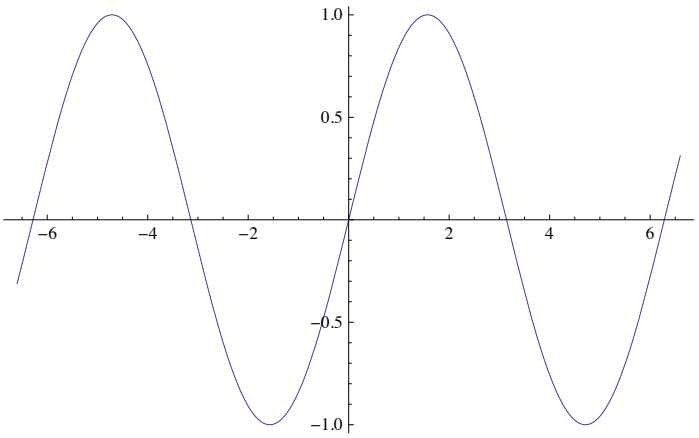
In[79]:=

plot sinx

Plots (1 of 2)

```
Plot[Sin[x], {x, -6.6, 6.6}]
```

Out[79]=



Note that we get both the correct *Mathematica* syntax, as well as the resulting graph. This is an excellent way to learn many of the capabilities. This type of input is more fully explained by Cliff's "Hands-on Start" video.

3D Plots

Here's how to find the syntax for a 3D plot.

In[82]:=

plot sin(x+cos(y)), x=-3..3, y=-3..3

3D plot

```
Plot3D[Sin[x + Cos[y]], {x, -3, 3}, {y, -3, 3}]
```

Let's modify that a bit with options. Here's a darker version of it, restricted to lie in a sphere and named so we can refer to it. Note that the **RegionFunction** is represented using a pure function.

```
restrictedPlot = Plot3D[{Sin[x + Cos[y]] + 1}, {x, -3, 3}, {y, -3, 3},
  RegionFunction → (#1^2 + #2^2 + #3^2 < 9 &),
  PlotStyle → Blue, BoundaryStyle → Thick]
```

Here's a clear image of the containing sphere (represented using graphics primitives).

```
clearSphere = Graphics3D[{Opacity[0.5], Sphere[{0, 0, 0}, 3]}]
```

Here they are together.

```
Show[restrictedPlot, clearSphere,
 PlotRange → {0, 3}]
```

Algebra and calculus

Mathematica can do algebra.

```
Expand[((x + h)^8 - x^8) / h]
```

It can take limits.

```
Limit[%, h → 0]
```

Thus, I guess it can do calculus.

```
D[x^8, x]
```

Even integration. Here is the tutorial's example:

```
Integrate[1 / (x^5 - 1), x]
```

They make it dynamic, too.

```
Manipulate[Integrate[1 / (x^n - 1), x], {n, 2, 10, 1}]
```

■ Solving equations

Mathematica can solve equations symbolically and numerically. Note that equations are represented using double equals signs.

```
Solve[x^2 + 2 x - 1 == 0, x]
```

Even simple polynomial equations can yield complicated symbolic results.

```
Solve[x^3 + 2 x - 1 == 0, x]
```

NSolve can be used as a numerical polynomial solver.

```
NSolve[x^3 + 2 x - 1 == 0, x]
```

Other simple equations are not amenable to the techniques of **Solve** or **NSolve**.

```
NSolve[Cos[x] == x, x]
```

We can see that there is a solution close to 0.8, however.

```
Plot[{Cos[x], x}, {x, 0, 1}]
```

In this case, the **FindRoot** command is the correct numerical tool to use to get an estimate. Note that the command accepts an initial guess, which we have from the plot.

```
FindRoot[Cos[x] == x, {x, 0.8}]
```