Introduction to Mathematica

Fall 2016

```
In[1]:= Clear["Global`*"]
```

Getting Started

▪ opening *Mathematica*

To open *Mathematica* on the computers in this lab (ECCR 143), click on the window button in the bottom left hand corner, and start typing “Mathematica,” soon the icon will appear in the list above. Simply click on it and then either choose to create a new notebook, or open an existing one.

▪ cell types and styling

In *Mathematica*, the basic formatting structure is a cell. If you hover your mouse on the right side of the notebook, square brackets will appear; each one encloses a cell. To create a new cell, press the down arrow until a horizontal line appears. If you want to input a command to *Mathematica*, just start typing, otherwise go up to the “format” menu, select the style you want and then start typing.

▪ input evaluation

To give *Mathematica* a command, like say compute the value of sin(1) to 7 significant figures, hit the down arrow until you see a horizontal line, type the command and then press “Shift+Enter” (just pressing enter will create a new line). Below, we’ve put in the command to compute the value of sin(1) to 7 significant figures, try evaluating it:

```
In[2]:=  N[Sin[1], 7]

Out[2]=  0.8414710
```

▪ deleting cells

To delete a cell, or group of cells, hover your mouse on the right side of the notebook until you see the square brackets. Select (click on) the set of brackets corresponding to the cells you want to delete, and then hit “delete.”

▪ the documentation center

Any time you want to do something in *Mathematica* and don’t know what function to use, or you know the name of the function but you don’t know how to use it, you should use the Documentation Center. To find it, go to the “help” menu and select “Documentation Center.” Try using the Documentation Center to figure out how to integrate in *Mathematica*.

Using *Mathematica* as a Basic Calculator

Mathematica can be used in many ways - as a means to create a report, as a programming language, to explore and visulaize mathematical ideas from class, and as a calculator. For example, we can carry out basic arithmetic operations:

```
In[3]:= (2 + 2) * 5 / 19
```

Out[3]= $\frac{20}{19}$

Suppose however, that we want a decimal approximation rather than the fraction form:

```
In[4]:= N[%, 13]

Out[4]= 1.052631578947
```

There are two things to note here, one is that “%” sign. This stands for “the last output.” Also note the “13” - this means we want *Mathematica* to give us the decimal form of the fraction to 13 significant figures. The function N[ ] also works on values like π (to get the “pi” symbol, type “<Esc> pi <Esc>”):

```
In[5]:= N[20 π]

Out[5]= 62.8319
```

Notice that the default is to give you six significant figures.

Defining Functions

To define a function in *Mathematica*, we need to make sure to use special notation:

```
In[6]:= f[x_] = x * Sin[x];
```

There are a few important things to note here:

1. we must have an underscore after the independent variable (on the left side only)
2. we cannot just type “xSin[x]” ... there must be either a space or an astrik between the “x” and the “S”
3. *Mathematica* already knows about many functions, like sin(x), but we must be sure to use an upper case letter for the first letter so that *Mathematica* knows we are asking for one of its built-in definitions.
4. The “;” is *suppressing* the output. We still must press “Shift+Enter” to evaluate and store this function as f, but it won’t return anything to us. If we hadn’t put the semi-colon there, it would have done this:

```
In[7]:= f[x_] = x * Sin[x]

Out[7]= x Sin[x]
```

Solving Systems of Equations

There are three main tools we can use to solve a system of equations: **FindRoot[...]**, **Solve[...]**, and **NSolve[...]**. **Solve[...]** will only work on simple systems such as

$$y = x^2$$
$$x^2\ (y-1) = 0$$

If we use **Solve[...]** on this system of equations, it will return to us the solution(s):

```
In[8]:=  Solve[{y == x^2, x^2 (y - 1) == 0}, {x, y}]
Out[8]=  {{x → -1, y → 1}, {x → 0, y → 0}, {x → 1, y → 1}}
```

Notice that we first listed out the equations of the system, using TWO equal signs. Then we give a list of what variables we want to sovle for. If we try to use **Solve[...]** on a more complicated system, it could fail to give us an answer:

```
In[9]:=  Solve[{Tan[x] == x y, x (y - 1) == 0}, {x, y}]

         Solve::nsmet : This system cannot be solved with the methods available to Solve. >>

Out[9]=  Solve[{Tan[x] == x y, x (-1 + y) == 0}, {x, y}]
```

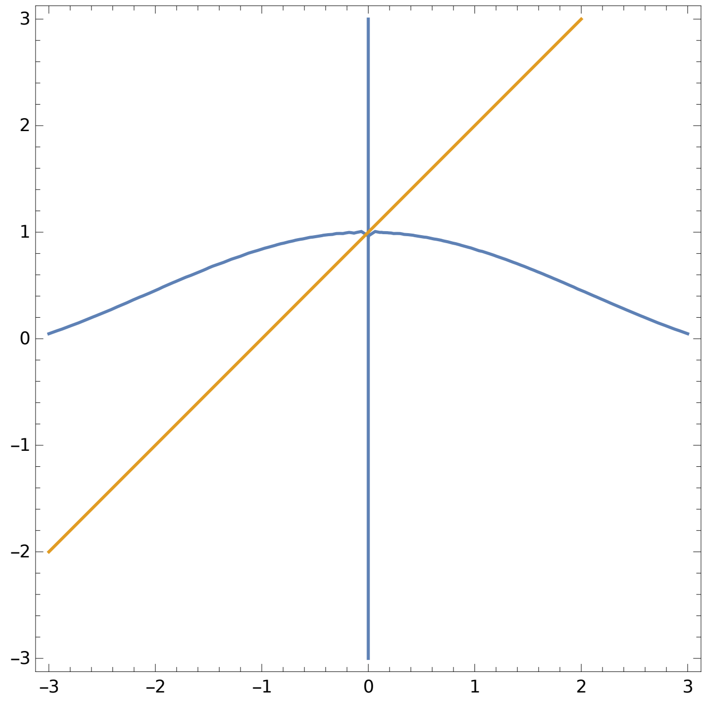
In this case, we need to use either **NSolve[...]**, or **FindRoot[...]**:

```
In[10]:=  NSolve[{Sin[x] == x y, y == x + 1}, {x, y}]

          NSolve::nsmet : This system cannot be solved with the methods available to NSolve. >>

Out[10]=  NSolve[{Sin[x] == x y, y == 1 + x}, {x, y}]

In[11]:=  FindRoot[{Sin[x] == x y, y == x + 1}, {{x, 0}, {y, 0}}]
Out[11]=  {x → 0., y → 1.}
```

Notice that in addition to specifying which variables we want to solve for, we also need to give an initial guess. A good way to select this initial guess is by looking at a plot:

```
In[12]:= ContourPlot[{Sin[x] == x y, y == x + 1}, {x, -3, 3}, {y, -3, 3}]
```

Out[12]=



We’ll talk more about contour plots later this semester.

Taking Limits

We can also have *Mathematica* calculate limits for us:

```
In[13]:= Limit[Sin[x] / x, x → 0]

Out[13]= 1
```

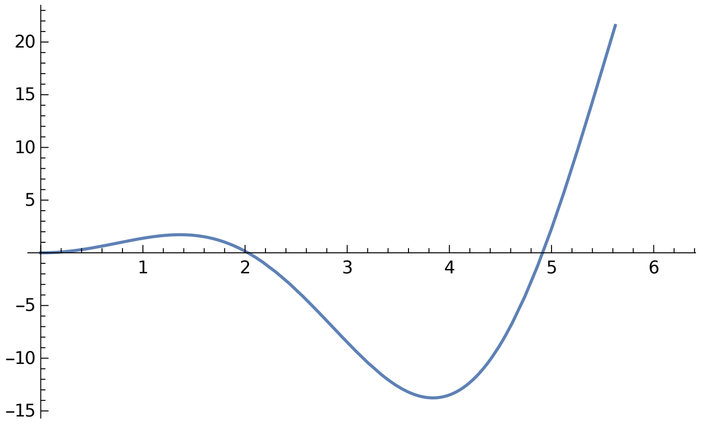
In the line above, the first *argument* is "**Sin[*x*] / *x***" and the second argument is ***x*→0**. The general notation to caluclate $\lim_{x \to x_0} f(x)$ is **Limit[*f*[*x*], *x*→*x*$_0$]**; that is the first argument is the function and the second argument is the independent variable name, then an arrow (get this arrow by typing "-" and then ">") and then finally the point at which you want to calculate the limit.

Making Basic Plots

One of the great things about *Mathematica* is that we can use it to help us visualize things we wouldn’t be able to otherwise. Suppose you are handed some complicated function and you need to know what it looks like:

In[14]:=
```
Clear[f] (* This gray part is a comment... Mathematica won't
 evaluate this part. Because we defined f a few slides earlier,
we should clear it before trying to re-define it. *)
f[x_] = x * Sin[x] + x^2 Cos[x]; (* To get that 2 up in the exponent position,
hold down "Ctrl" and type "6" This puts you in the superscript...
 to get out of it, either use the right or down arrow key. *)
Plot[f[x], {x, 0, 2 π}]
```

Out[16]=



The **Plot[ ]** function has two required arguments. First, you must put the function you want to plot. Second, you must include a list containing (1) the independent variable name, (2) the smallest value to show up on the x-axis, and (3) the largest value to show up on the x-axis.

There are many ways to modify a plot: we can add labels to the axes, we can change the bounds on the y-axis, we can change the color of the line representing the function, and much more. You will need to this on the homework and can find further information and examples in the Documentation Center.

Differentiation

There are two ways to have *Mathematica* take a derivative. The first way requires you to have already defined a funtion (we did that on the previous slide):

```
In[17]:= f'[x]
Out[17]= 3 x Cos[x] + Sin[x] - x^2 Sin[x]
```

So we can simply put a prime after the function name like we would when we write it out on paper. Other times though, we may not want to save the funtion under a particular name prior to calculating the derivative. Then we can do this:

```
In[18]:= D[Cos[x] Sin[x], x]
Out[18]= Cos[x]^2 - Sin[x]^2
```

The first argument is the function name and the second argument is the variable that we want to differentiate with respect to.

Integration

```
In[19]:= Integrate[Sin[x], {x, 0, 2 π}]

Out[19]= 0
```

The first argument is the function we want to integrate, the second is a list which has (1) the variable of integration, (2) the lower limit, and (3) the upper limit.

Sometimes, we encounter complicated integrands for which there is no anti-derivative. In this case, we can have *Mathematica* give us a numerical approximation to the definite integral:

```
In[20]:= NIntegrate[Sin[x] / (2 + Sin[x]), {x, 0, 2 π}]

Out[20]= -0.972012
```

( ) vs [ ] vs { }

In *Mathematica* parenthesis, square brakets and curly brackets all have distinct meanings:

( )	for order of operations
[ ]	to go around the arguements and options of *a* function
{ }	to go around elements of *a* list

You can not use them interchangeably!!!

The Kernel

The Blank Kernel

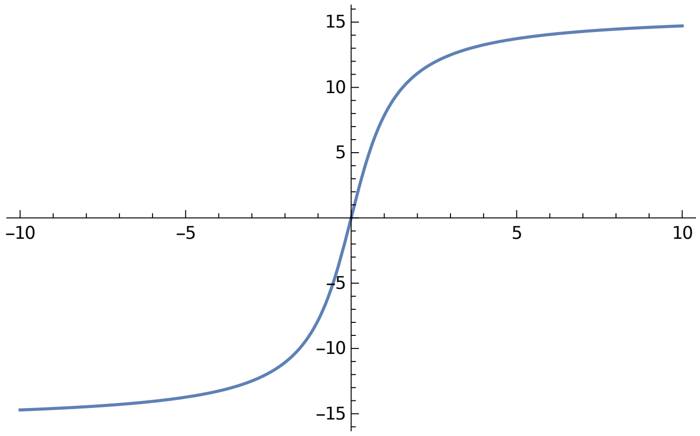
The kernel is the part of *Mathematica* that does all of the computational work and holds all of the function definitions that we've made. Each time we open *Mathematica*, the kernel is “blank” in the sense that it doesn't know about any definitions yet. For example, say we've been working hard on our homework in class, but haven't quite finished up the assignment yet. We save it to out flash drive and come back tomorrow to work on it. Even though the defintions we made yesterday are right there on the notebook, *Mathematica* doesn't know about them yet because we haven't sent them to the kernel. Because the kernel is blank each time *Mathematica* is opened, we will need to re-evaluate our notebook each time we open *Mathematica*.

Accidentally Deleting Commands

Sometimes, as students are working they will accidentally delete an input line. For example, say a student originally has the following in their notebook:

```
In[21]:= Fun[θ_] = 10 ArcTan[θ];
         Plot[Fun[θ], {θ, -10, 10}]
```

Out[22]=



But at some point, they accidentally delete the first line. Things will continue to work fine, because that definition of **Fun[ ]** has already been put in the kernel, but once they close *Mathematica*, they will loose that defintion. As a result, when they save their notebook, and then come back to it later and re-open *Mathematica*, they will have trouble. The kernel is blank each time we open *Mathematica*, so any time they try to use **Fun[θ]**, they will encounter problems because *Mathematica* won't know what they're talking about. To make sure this doesn't happen to you, you should follow the instructions on the next page.

Homework Submission

Each week you will have a notebook like this to read and then a homework assignment to complete. Your homework should be written in *Mathematica* and

- have a title summarizing what topics the homework covers
- have your name
- have `Clear["Global`*"]` as the first input cell in your notebook (note that the symbol after Global is the tick below the tilda in the upper left hand side of your key board, not an apostrophe
- be neatly formatted like this one, with all unecessary cells deleted
- **have no errors**
- **be complete**
- have all output deleted... to do this, go to the Cell menu and select delete all output before saving (you'd have to re-evaluate all of it when you open it back up anyways)

If the above requirements are not satisfied, you may recieve no credit for the assignment. Sometimes, students will define a function and as they're working end up deleting the definition before they save their notebook. Then when the TA opens up the notebook to grade it, the notebook doesn't work properly because the definition isn't there. To avoid this issue, make sure that you follow these steps before submitting your assignment:

1. delete all output as discussed above
2. save your notebook
3. close *Mathematica*
4. re-open *Mathematica* and your homework
5. re-evaluate the entire notebook and make sure everything works properly
6. delete all output and save the file again