

Minimal Mathematica

This handout is about the program called *Mathematica*, which is a high-level symbolic manipulation program that can do a great many interesting things. Precisely because it is so powerful, *Mathematica* isn't easy to use. This handout tries to teach you just enough about the program to get you going, but no more than that. Consult the friendly help file if you need to know more.

Mathematica can be used to write articles and to do mathematics. In fact, this handout is being written inside *Mathematica*. But the main things we want to do with it are going to be (a) plotting and graphing, and (b) simple algebra.

Getting Started

The first thing you need to do is to find the program. *Mathematica* has been installed in several machines on campus. It should be available in the computer lab on the fourth floor of Mudd (Mudd 416) and in the Olin computer lab (Olin 324). These copies work with a "key server," which is a program on the central server that controls how many copies are in use at a given time. Colby only has a license to use a limited number of copies. In practice, this means four things:

-- you can get a copy of *Mathematica* from the Colby server and install it on your machine (see the ITS web pages for instructions),

-- that copy, however, will only run if you are on campus and the keyserver has a key you can use;

-- if you try to use *Mathematica* and get a message saying that there are too many copies in use, you have to wait until fewer people are using the program; and

-- if you are done using the program, shut it down so that others can use it!

When you start up *Mathematica* you get a blank screen called a "notebook." Once you start typing, you will see some weird markings on the right of the screen. *Mathematica* notebooks are composed of "cells," which are indicated by square brackets at the right. Cells can have subcells, and different cells have different functions. This cell, into which I am typing right now, is formatted as a text cell. Since you mostly will be computing stuff, you can probably just use the default, which is that every cell is an input cell. (You don't see cell markings on this printout, but you will if you look at this notebook online.)

So let's just type something into the first cell. Let's ask the program to compute a large power. You do this in the obvious way, except for one detail. So type in $37!$ and **then hit the Enter (not the Return) key**. (If you're on a laptop, Shift+Enter will also work. I'm not sure how it works on the Mac, sorry.)

```
In[1]:= 37^34
```

```
Out[1]= 208381240119593773598371865160915107617069000520604889
```

Notice that *Mathematica* numbers your input as In[1], and numbers the output cell correspondingly. At this point, you can use *Mathematica* as a calculator (but a slightly weird calculator).

```
In[2]:= 3 + (44 * 3^2) / 313
```

```
Out[2]=  $\frac{1335}{313}$ 
```

```
In[3]:= N[%]
```

```
Out[3]= 4.26518
```

Notice that by default *Mathematica* tries to give you exact answers. The N function tells it to give a numerical value; % always refers to the latest output. I could also have written it as N[Out[2]].

How about variables? *Mathematica* is happy to use them. It treats any word (without spaces in it) as a variable, unless it already knows what that word means. So if you write

```
In[4]:= 33 foo^2 + 4 foo - 2
```

```
Out[4]= -2 + 4 foo + 33 foo^2
```

it will happily treat "foo" as a variable. That means that if you want f times o times o, you need to add spaces:

```
In[5]:= f o o
```

```
Out[5]= f o^2
```

Mathematica understands that two juxtaposition means multiplication, but you can use * too if you want:

```
In[6]:= f * o * o
```

```
Out[6]= f o^2
```

Now comes an important thing. In everyday use, we use parentheses with several different meanings, depending on context. When we write $f(x+1)$ there is usually no doubt about whether we mean the variable f times the polynomial $x+1$ or the function f evaluated at $x+1$. Computer programs aren't that smart. So, in *Mathematica*, round parentheses always mean grouping. Square brackets always mean function evaluation. So:

```
In[7]:= f (x + 1)
```

```
Out[7]= f (1 + x)
```

```
In[8]:= Expand[%]
```

```
Out[8]= f + f x
```

```
In[9]:= f[x + 1]
```

```
Out[9]= f[1 + x]
```

```
In[10]:= Expand[%]
```

```
Out[10]= f[1 + x]
```

Of course, *Mathematica* doesn't know which function "f" is, so it can't expand the last expression.

Built-In Functions

Mathematica comes with several sorts of built-in functions. Some are like the buttons in your calculator: sine, cosine, exponential, etc. Some are like the Simplify function above: they do things to expressions. There are, of course, thousands of these; use the Help pages. One basic thing to know, however: all built-in functions have names that start with a capital letter. So "sin" either is a new variable or a new function; Sin is the sine function.

```
In[11]:= Sin[x]
```

```
Out[11]= Sin[x]
```

```
In[12]:= Sin[2]
```

```
Out[12]= Sin[2]
```

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

```
Out[13]= 0.909297
```

If you write `sin[2]`, *Mathematica* will warn you that it looks very much like a function it already knows:

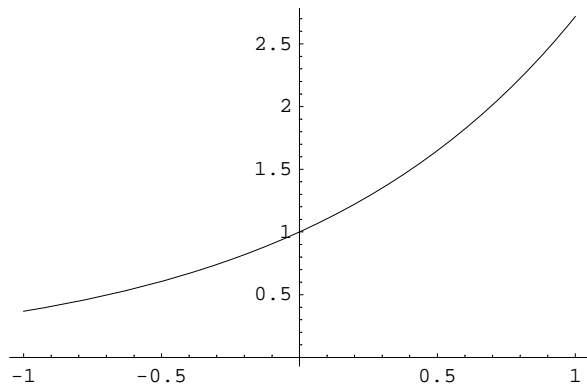
```
In[14]:= sin[2]
```

```
General::spell1 : Possible spelling error: new symbol name "sin" is similar to existing symbol "Sin". More...
```

```
Out[14]= sin[2]
```

The exponential function e to the x is called Exp:

```
In[40]:= Plot[Exp[x], {x, -1, 1}]
```



```
Out[40]= - Graphics -
```

It does have a name for e, which is, of course, input as capital E:

```
In[41]:= Exp[1]
```

```
Out[41]= e
```

```
In[42]:= E^1
```

```
Out[42]= e
```

Don't use lower-case e; you'll confuse the beast.

```
In[43]:= e * Exp[1]
```

```
Out[43]= e e
```

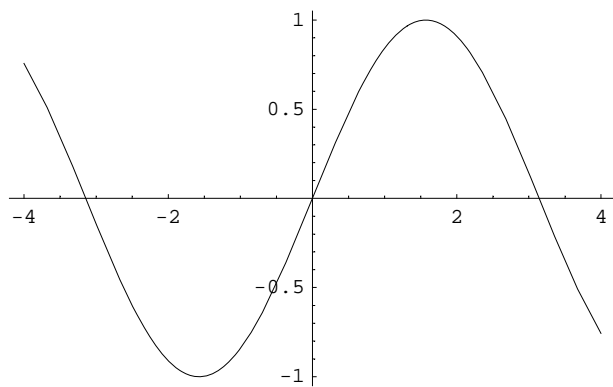
```
In[44]:= Exp[2]
```

```
Out[44]= e2
```

Plotting

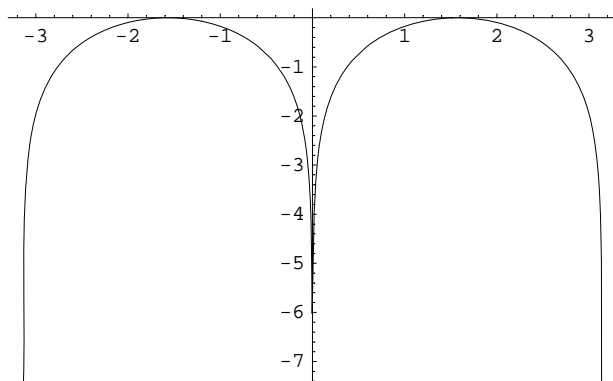
Mathematica has several built-in functions for plotting. In general, the rules are: you must specify the function, and you must specify the range of values to use in the plot. The plot functions have a million other settings that you may use, but these two are the essential ones. Here are the examples to imitate:

```
In[15]:= Plot[Sin[x], {x, -4, 4}]
```



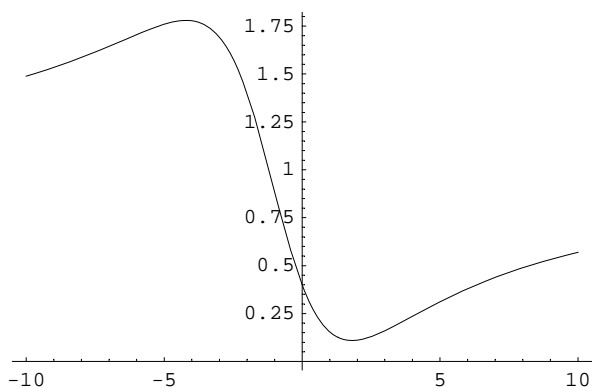
```
Out[15]= - Graphics -
```

```
In[16]:= Plot[Log[Abs[Sin[x]]], {x, -Pi, Pi}]
```



```
Out[16]= - Graphics -
```

```
In[17]:= Plot[(x^2 - 3 x + 4) / (x^2 + 2 x + 10), {x, -10, 10}]
```



```
Out[17]= - Graphics -
```

Notice that *Mathematica* automatically picks an appropriate "window" on the y-axis. You can force it to use the window you want, but in general it is fine to let it decide on its own.

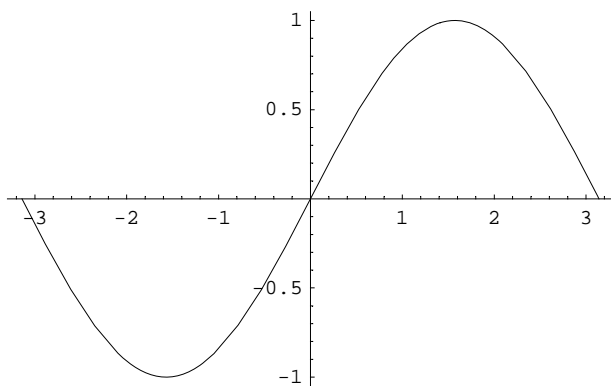
Defining Functions

Sometimes you want to define your own functions. Here's how you do it. To tell *Mathematica* to treat a variable as something you can plug in for, you append an underscore to it. So:

```
In[18]:= f[x_] = Sin[x]
```

```
Out[18]= Sin[x]
```

```
In[19]:= Plot[f[x], {x, -Pi, Pi}]
```



```
Out[19]= - Graphics -
```

Let's make a composite function. First let's define a function g, then compute g on f.

```
In[20]:= g[x_] = x^3
```

```
Out[20]= x3
```

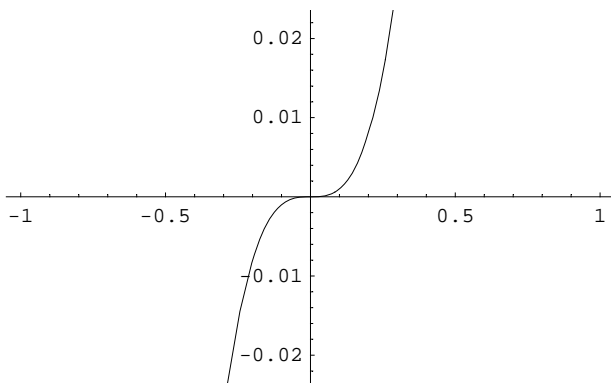
```
In[21]:= f[g[x]]
```

```
Out[21]= Sin[x3]
```

```
In[22]:= f[g[3]]
```

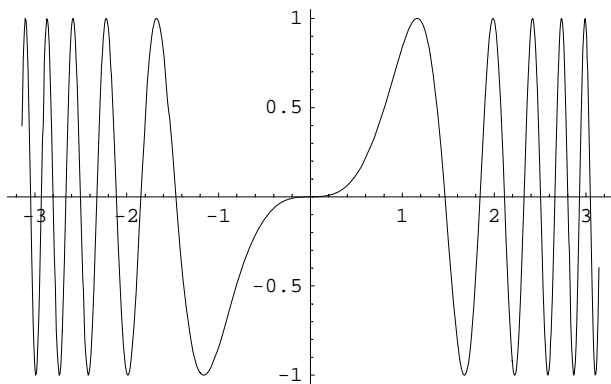
```
Out[22]= Sin[27]
```

```
In[23]:= Plot[f[g[x]], {x, -1, 1}]
```



```
Out[23]= - Graphics -
```

```
In[24]:= Plot[f[g[x]], {x, -Pi, Pi}]
```



```
Out[24]= - Graphics -
```

Derivatives

Mathematica knows how to compute derivatives. In fact, it can do them in two different ways:

```
In[31]:= f[x_] = Sin[x]
```

```
Out[31]= Sin[x]
```

```
In[32]:= f'[x]
```

```
Out[32]= Cos[x]
```

```
In[33]:= D[f[x], x]
```

```
Out[33]= Cos[x]
```

Use the second method when you have several variables in an expression, and you want to tell one which one is really the variable (so that the others are constants). It's happy to do truly nasty ones:

```
In[34]:= f[x_] = Log[Sin[x]^2 + (x + 4)^(2/3)] / (x Exp[x] + 1)
```

```
Out[34]= 
$$\frac{\text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2]}{1 + e^x x}$$

```

```
In[35]:= f'[x]
```

```
Out[35]= 
$$-\frac{(e^x + e^x x) \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2]}{(1 + e^x x)^2} + \frac{\frac{2}{3(4+x)^{1/3}} + 2 \text{Cos}[x] \text{Sin}[x]}{(1 + e^x x) ((4+x)^{2/3} + \text{Sin}[x]^2)}$$

```

```
In[36]:= Simplify[%]
```

```
Out[36]= 
$$\frac{-e^x (1+x) \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2] + \frac{(1+e^x x) (\frac{2}{3(4+x)^{1/3}} + \text{Sin}[2x])}{(4+x)^{2/3} + \text{Sin}[x]^2}}{(1 + e^x x)^2}$$

```

```
In[37]:= Expand[%]
```

$$\text{Out[37]} = -\frac{e^x \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2]}{(1+e^x x)^2} - \frac{e^x x \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2]}{(1+e^x x)^2} +$$

$$\frac{2}{3(4+x)^{1/3}(1+e^x x)^2((4+x)^{2/3} + \text{Sin}[x]^2)} + \frac{2e^x x}{3(4+x)^{1/3}(1+e^x x)^2((4+x)^{2/3} + \text{Sin}[x]^2)} +$$

$$\frac{\text{Sin}[2x]}{(1+e^x x)^2((4+x)^{2/3} + \text{Sin}[x]^2)} + \frac{e^x x \text{Sin}[2x]}{(1+e^x x)^2((4+x)^{2/3} + \text{Sin}[x]^2)}$$

```
In[38]:= Simplify[%]
```

$$\text{Out[38]} = (2 + 2e^x x - 12e^x \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2] - 15e^x x \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2] -$$

$$3e^x x^2 \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2] - 3e^x(1+x)(4+x)^{1/3} \text{Log}[(4+x)^{2/3} + \text{Sin}[x]^2] \text{Sin}[x]^2 +$$

$$3(4+x)^{1/3}(1+e^x x) \text{Sin}[2x]) / (3(4+x)^{1/3}(1+e^x x)^2((4+x)^{2/3} + \text{Sin}[x]^2))$$

Finally, you can do all sorts of other things.

```
In[30]:= Solve[x^3 - 6*x^2 - 12*x - 32 == 0, x]
```

$$\text{Out[30]} = \{\{x \rightarrow 8\}, \{x \rightarrow -1 - i\sqrt{3}\}, \{x \rightarrow -1 + i\sqrt{3}\}\}$$

```
In[39]:= Factor[x^2 - 5x + 6]
```

$$\text{Out[39]} = (-3 + x)(-2 + x)$$

Error Messages

You're going to get them. They're red. *Mathematica* beeps at you. The messages aren't always very clear. Get over it.

The thing to keep in mind is that computers are very literal-minded. What's worse, if you make a mistake but it finds a way to make sense of what you wrote, it won't give you an error message. That's actually worse! If you try writing `Sin(x)`, for example, *Mathematica* will take you at your word: you're defining a variable called `Sin` and multiplying it by `x`. Bad idea, but it doesn't tell you that

```
In[45]:= Sin(x)
```

$$\text{Out[45]} = \text{Sin } x$$

Sometimes you can get help by typing a question mark followed by your function:

```
In[46]:= ?Sin
```

`Sin[z]` gives the sine of `z`. More...

Two question marks gives you more than you probably want:

```
In[47]:= ??Sin
```

`Sin[z]` gives the sine of `z`. More...

`Attributes[Sin] = {Listable, NumericFunction, Protected}`

In[48]:= ?? Plot

Plot[f, {x, xmin, xmax}] generates a plot of f as a function of x from xmin to xmax. Plot[{f1, f2, ... }, {x, xmin, xmax}] plots several functions fi. **More...**

Attributes[Plot] = {HoldAll, Protected}

Options[Plot] = {AspectRatio → $\frac{1}{\text{GoldenRatio}}$, Axes → Automatic, AxesLabel → None, AxesOrigin → Automatic, AxesStyle → Automatic, Background → Automatic, ColorOutput → Automatic, Compiled → True, DefaultColor → Automatic, DefaultFont → \$DefaultFont, DisplayFunction → \$DisplayFunction, Epilog → {}, FormatType → \$FormatType, Frame → False, FrameLabel → None, FrameStyle → Automatic, FrameTicks → Automatic, GridLines → None, ImageSize → Automatic, MaxBend → 10., PlotDivision → 30., PlotLabel → None, PlotPoints → 25, PlotRange → Automatic, PlotRegion → Automatic, PlotStyle → Automatic, Prolog → {}, RotateLabel → True, TextStyle → \$TextStyle, Ticks → Automatic}

And sometimes you get nasty error messages.

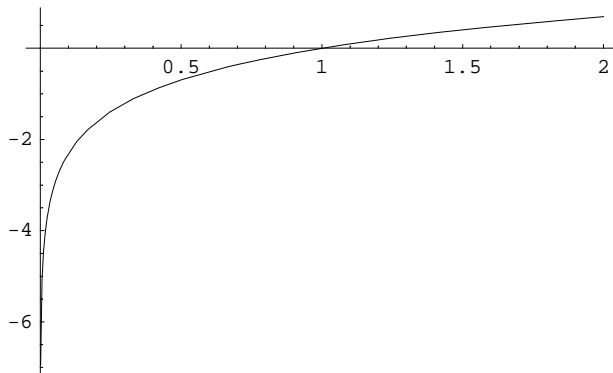
In[25]:= Plot[Log[x], {x, -2, 2}]

Plot::plnr : Log[x] is not a machine-size real number at x = -2.. **More...**

Plot::plnr : Log[x] is not a machine-size real number at x = -1.83773. **More...**

Plot::plnr : Log[x] is not a machine-size real number at x = -1.66076. **More...**

General::stop : Further output of Plot::plnr will be suppressed during this calculation. **More...**



Out[25]= - Graphics -

The problem here is that Log can't be computed for negative values of x.

In[26]:= Plot[Bessel0[x]]

Plot::argmu : Plot called with 1 argument; 2 or more arguments are expected. **More...**

Out[26]= Plot[Bessel0[x]]

We forgot to give the range of x.

Sometimes it's not clear what is wrong:

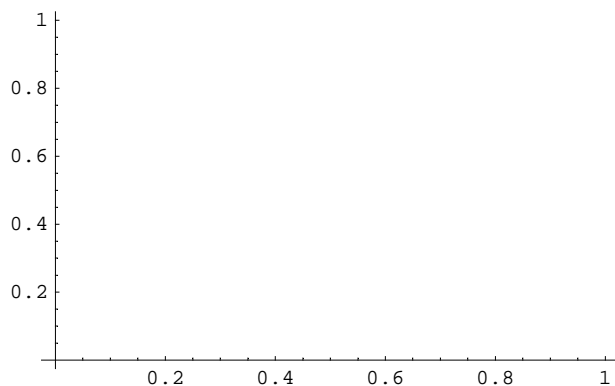
```
In[27]:= Plot[Bessel0[x], {x, -10, 10}]
```

```
Plot::plnr : Bessel0[x] is not a machine-size real number at x = -10.. More...
```

```
Plot::plnr : Bessel0[x] is not a machine-size real number at x = -9.18866. More...
```

```
Plot::plnr : Bessel0[x] is not a machine-size real number at x = -8.30382. More...
```

```
General::stop : Further output of Plot::plnr will be suppressed during this calculation. More...
```



```
Out[27]= - Graphics -
```

Mathematica didn't like `Bessel0[x]`. Maybe it doesn't know that function?

```
In[28]:= ? Bessel0
```

```
Global`Bessel0
```

That's *Mathematica* mumbo-jumbo for "you wrote something called `Bessel0` down, but I don't know what it is. It's just a global variable right now.

```
In[29]:= ? BesselJ
```

```
BesselJ[n, z] gives the Bessel function of the first kind  $J(n, z)$ . More...
```

Aha, so the thing we want is really `BesselJ[0,x]`. Try that.

So...

That's enough. Go and experiment. Don't be afraid of horrible error messages. Experiment further with plotting and derivatives. Play with `Solve`. Find out what `Integrate` does. Try out `ParametricPlot`. Have fun.