

INTRODUCTION TO MATHEMATICA FOR PHYSICS 103

The following introduction to Mathematica was written by Kirsten Nordstrom '04, updated by Prof. Elizabeth McCormick, and then heavily modified for Physics 103 this year. The original document has been used in physics 214, physics 306, and physics 308, among other places. The writeup is meant to encourage you to *play around* and *explore* Mathematica, while giving only a very limited number of “cookbook” type instructions. This is a fun way to learn, but the downside to this approach is that you may find yourself baffled from time to time when trying to figure out the precise syntax needed to get Mathematica to do something. Make use of Mathematica’s help feature and make use of the web. Finally, make use of other people (your colleagues in the class and your instructor) if you can’t figure something out. If you find yourself bogged down for more than a few minutes on any particular point, *stop* and *get help*.

If you are already familiar with Mathematica, you are free to completely ignore this writeup.

For physics 103, you will need to know how to do plotting, i.e., read this writeup through the second paragraph of the second section. Beyond that is strictly optional.

There are numerous computers around Park that have Mathematica installed. It should be available on the cluster of computers in the science library. It is available on essentially any physics department computer you can get your hands on. The math department computer lab (Park 352) also has it. I will have office hours in Park 352 on Thursday from 9:40 to 11:30 and from 4:10 to 6:00.

You will need to use Mathematica this week and next in Physics 103. I am not yet sure whether or not we will use it more this semester.

If you have your own computer, you can install Mathematica on it for free for use while on campus (or at Haverford or Swarthmore); see <http://www.brynmawr.edu/computing/docs/StudentMathematica.shtml>

An Introduction to *Mathematica*

Welcome to the world of computational physics! This lab will give you the tools you need to start exploring this discipline. Computational physics has developed into an effective method of doing physics. It can be compared and contrasted with experimental and theoretical physics. Essentially, it makes the claim that computer modeling and computer generated numerical solutions are as informative as closed form solutions. What does this mean? If I solve the differential equation $y'(t) = y(t)$, most second-year physics students could rattle off the closed form solution, that is, $y = ce^t$. In contrast, most computerized ways of solving this equation would give numerical values of t and values of y , like so:


$$\{(t,y)\}=\{(0,1), (1, 2.718), (2, 7.389)\dots\}$$

While numerical solutions like this aren't necessarily useful when we have a nice neat equation for the solution, (like $y = ce^t$), numerical solutions are very helpful when the equation is difficult or impossible to solve. Situations like this arise in complex and/or nonlinear systems, and in many problems of advanced theoretical physics. Numerical computations makes values of functions accessible, and the results can be plotted even when the underlying function is not known in analytic form. This is very much "experimental" mathematics.

Mathematica is a powerful tool for both analytic work (formulas) and for numerical work. You may have used previously in calculus or some other course. If you decide to remain a scientist, you'll probably continue to use it throughout your career. It's THAT good. You can do almost anything in *Mathematica*, you can make numerical calculations (as you would on a calculator), solve algebraic equations, solve differential equations, do matrix algebra, integrate, create graphs and animations...the list goes on and on. One wonderful thing about *Mathematica*, as opposed to simpler languages, is that it recognizes common functions (such as e^x), and thus can give you closed form solutions in many, many cases. When it can't do that, it can generally give you a numerical solution. It even handles imaginary numbers with ease. The only thing it doesn't do for you is your homework (except when you are assigned to use it...). This lab will walk you through the basics, helping you get comfortable with *Mathematica* as a tool you can use to help you do physics.

PART I: Getting Started

1. The Basics

The *Mathematica* icon looks like this: . Click on it to get started. When you open the program you might be confronted with several windows. One might say "Welcome to *Mathematica*" and offer to give you a tour. Close it. You can show yourself around. You also might be confronted with a palette, a nice array of symbols so you don't have to enter the text commands. Close the palette. You should learn to do things for yourself.

The big blank window is called the notebook. In this notebook, you will type in commands and then hit "Shift-Return" (hold the "Shift" key while pressing the "Return" key) or just "Enter" on the keyboard. Then *Mathematica*'s kernel (essentially, its brain) will evaluate the cell (as it is called) you have just entered. The output will then be displayed in the notebook. Now you are ready to try a simple example. Try entering "2+2". Type this (without the quotation marks) and then press Shift-Return or Enter. What happens? Can you guess what the commands are for subtraction, multiplication, and division? Exponents are handled by the ^ symbol. Try some things that are way too big for your calculator. How about 3 to the 200th power? How about things that are way too small? Divide 7 by 10 to the 100th power (1 followed by 100 zeros). You should note two things here. What do calculators give you as an answer? Is the *Mathematica* answer useful?

The *Mathematica* answer might not be useful because in an effort to be as precise as possible it has kept the quotient of these two numbers as a fraction. To get the decimal solution is easy. Enter the operation "N[%]" on a new line. The "N" command tells it to produce a numerical expression. "N" operates on "%", which means the "last output". Try it. That form of "N" is called a "prefix" operation. You can also use a "postfix" operation. It is the same operator "N" but you can type in the operation at the end of the output line. The syntax is "...//N" where "..." is the output you had. Why are these operations called prefix and postfix? For every command there is at least a prefix and postfix form. There is another form, called "infix". What would that be? Try "Plus[2,5]". (The prefix version of the infix operation "+".)

The N operator can give you as many digits as you want. Try `N[1/3,500]`. What about `N[Pi,500]`? (Try it!)

Mathematica is perfectly happy when you write “Pi” for the Greek letter π , but other Greek letters have to be entered by hand, and there is a better way to enter π , too. Here’s how to do it. To enter a Greek letter, say, Delta (Δ), type “`EscDeltaEsc`”, where `Esc` means to press the escape key. In other words, press the escape key, type the name of the letter you want, and press the escape key again. If you want the upper-case Greek letter, capitalize it when you type it in (like “Delta”); if you want lower the lower case greek letter, use lower case when you type it in: “`EscdeltaEsc`” gives δ . To get π , type “`EscpiEsc`”. Many other symbols can be entered in a similar way; for example, “`EscinfEsc`” gives ∞ . (Train your fingers to type these codes! Do not be tempted to use the palette! In the long run, this will save you much time and effort.)

A powerful aspect of *Mathematica* is that it can also do symbolic algebra, not just numerical evaluation. Try entering “`x+x`”. Try “`(x3 + 3x2y + 3xy2 + y3)/(x+y)`”. Hmm... what about “`Simplify[%]`”? Enter in the expression “`x2+2x +1`”. Now try “`Factor[%]`”. Can you guess the command to expand this output to its original form? Play around with some other symbolic examples.

Okay, now we have to go through some more syntax and definitions before we can really get cranking. You might need to refer back to this section to reorient yourself later on, when error messages start popping up.

1. *Mathematica* is case-sensitive. Try entering “`sin[π]`”. Did you get an error message? To *Mathematica* “sin” is just a string of letters, but “Sin” is a command. Now try “`Sin[π]`”. That’s more like it. The same idea applies all over: In general, commands are capitalized.

2. *Mathematica* is sensitive to poor syntax. The most important thing to remember your syntax is when using brackets. Each bracket must have a partner of the same type facing the other way. Enter “`N[4/3]`”. Note the error message. Now close the brackets and run it again. Much better. There are three types of brackets. The first and most familiar are the parenthetical brackets, (). Use these as “separators” in algebraic expressions. So `(3)(4)=12`, just like in old times. (Check it out to make sure.) The next beasts are the square brackets, []. These signify an operation is being done on the stuff inside the brackets. So `f[x]` means that the function `f` operates on the variable `x`. The curly brackets, { }, you may remember if you’ve done any set theory in math. They essentially do the same thing here, they act to group information together.

3. *Mathematica* allows variable names to have as many characters as you want. So you could write “`x=1`” but you could also write “`xy=1`”. This would not mean `x` times equals 1. It would mean that you have a variable called “`xy`” which you have set to 1. If you want to multiply `x` and `y`, you need to leave a space between them: “`x y`”. Otherwise *Mathematica* will think you are talking about a variable called `xy`.

4. There are three “equals” commands. There are major differences between “`=`”, “`==`” and “`:=`”. You will see some examples of this further along in this writeup. Or check out the entries for the Equals commands in the Master Index which you can search via the Help Browser.

5. Each operation needs a specific number of arguments. An example of an operation needing only one argument is the “Expand” operation. Its form is simply “`Expand[x]`”. An example of an operation needing two is the “D” operation for differentiation. You must specify the function and the variable you want to differentiate with respect to. So these are in the form `D[f,x]`. An operation

needing three arguments is “DSolve”. This command solves a differential equation (or system of them or a partial differential equation, if you are comfortable with curly brackets). The arguments for “DSolve” would be the differential equation itself, then the function to solve for, then the independent variable of the function. So the form is “DSolve[Eqn, f, x]” for a differential equation of just $f[x]$. (Things get a little more exciting when we add more variables to make it a partial differential equation or put in boundary conditions.)

If you want to play with more examples before you get started on your own, a good place to go is under the “Help” heading and scroll down to the “Help Browser”. The help browser is wonderful as its pages are all *Mathematica* notebooks. You can change the input and reevaluate the cells. The help browser also has a search tool and connects to the *Mathematica* website. However, don't rely on it too much. It provides many examples but little non-technical explanation. Sometimes it is helpful just go the the web for help. For instance, enter “Mathematica plotting” into a search browser and you're sure to find help on making plots. Finally, if you know a Mathematica command but forget exactly how to use it, enter (using the cosine function as an example) “?Cosine” in the notebook, and You will be given a brief help message and a link to click for further information.

2. Differentiation, Integration, Plotting

Now that you have the tools to work *Mathematica*, let's use it to do some basic operations. In the previous section you were told what the differentiation operator was. So take the derivatives of some functions, and see what you get. Take derivatives of functions of several variables. Hmm...helps with multivariable homework. There is also an infix operation that is a little easier to deal with. First define a function $f[x]$. For function definitions you need to input “ $f[x_]:=$ ” followed by what you want f to be. Notice the underline character after the x . You need to make sure that is there. Try making $f[x]$ a polynomial in x . For example: $f[x_]:=x^3+2x^2+2x+1$. When you type this in, the computer will write out the expression. Try writing it out again, “ $f[x]$ ”. Try evaluating f at a few values, for example “ $f[1]$ ”. Here is where *Mathematica* is a great timesaver, as it can evaluate a function at arbitrary values, without having to reinput the function every time, as you would with a calculator.) Now enter “ $f'[x]$ ”. The “'” tells it to take the derivative of f with respect to x . Now enter “ $D[f[x],x]$ ”. And you see where we are going. You can also do derivatives with the D operator: “ $D[f[x],x]$ ” takes the derivative of f with respect to x . “ $D[f[x],\{x,3\}]$ ” takes the third derivative.

You can define a function of more than one variable: “ $f[x_,t]:=Sin[10 x - 5 t]$ ”. Then try “ $f[1,1]$ ”, “ $f[x,t]$ ”, “ $f[0,t]$ ”, etc.

Another powerful aspect of *Mathematica* is its visualization capabilities. Here's how to graph $f[x]$ vs. x using the “Plot” command. The Plot command requires two arguments: The function, and the independent variable. The range of the independent variable must be specified, too. Decide how far you want x to extend. The command ends up looking like: “Plot[$f[x],\{x, xmin, xmax\}]$ ”. For example, “Plot[$f[x],\{x,0,10\}]$ ”. Or “Plot[Cos[x], $\{x,0,2\pi\}]$ ”. Run these and look at the picture. You can plot two functions on the same graph: “Plot[{Sin[x],Cos[x]}, $\{x,0,2\pi\}]$ ”. Want to make them two different colors? Try “Plot[{Sin[x],Cos[x]}, $\{x,0,2\pi\},PlotStyle->\{Red,Blue\}]$ ”. Oooo. Try plotting a more exciting function. There are other ways to change the style of the plot. Check out section 1.9.3 of The *Mathematica* Book via the Help Browser window for these options and some examples. Check out the Help entry on “Graphics`Colors`” for a full list of colors. You need to enter “<<Graphics`Colors`” before using the more exotic colors.

The next step is integration. As you hopefully know, we can take derivatives of many continuous functions with our eyes closed, as long as we remember the chain rule. However, integrals are little harder to obtain and usually require a bunch of guesswork. *Mathematica* can help you a lot, at least in getting the answer. Another nice feature is that it will evaluate both definite and indefinite integrals. Let's try indefinite integrals first. The command is “Integrate” and requires two

arguments, the function and the variable of integration. Integrate something boring as a check. Now integrate something exciting, that you could only attempt by parts, like $x^2\cos[x]$. What about something you couldn't even attempt, like $\cos[x^2]$. Whoa. It can do it! Look up something in an integral table and evaluate it in *Mathematica*. Now try a favorite: $\sin[x]/x$. Hmm...it can't do it. Never fear, there are ways to get around this. We'll get there very soon. First, however, practice a few definite integrals. Instead of just specifying the independent variable, what else will you have to specify? Try it!

Now let's attempt to find a value for the elusive $\sin[x]/x$. First, plot the function. Why doesn't it like it? It doesn't seem too bad. The truth is, it's not, except at one particular point. The graph doesn't show it but it is apparent just looking at the equation. It is hopeless for us to get an indefinite integral out of this. But let's try a definite integral. Definite integrals are easier to obtain as they can be obtained just by counting the pixels under a curve. Try evaluating it in the domain $[-1,1]$. It still hates it, as the problem is still in there. However, we can push the limits of how close we go to zero and break the integral up into pieces. Devise a method for evaluating this integral. How accurate can you get it? How can you tell if it's accurate?

An interesting thing is that in this case, *Mathematica* likes the improper integral a whole lot more. Evaluate the integral of $\sin[x]/x$ from $-\infty$ to ∞ . What is your answer? Does it surprise you that it can do the integral? As an aside, now take the integral of $\sin[3x]/x$ over the same limits. Now replace 3 with a larger integer. Keep going. Plot some of these functions. What happens to the shape and the area under the curve as the integer gets larger? As the integer approaches infinity, you might recognize this as the delta function. What might a delta function describe in the real world?

When you were checking the accuracy before you probably devised a "recursive" method of doing this (what does this mean?). The computer does the same thing when it performs the operation "NIntegrate". This gives an approximation of the area under the curve. You can specify how precise you want it, but we won't do that now. Integrate $\sin[x]/x$ over $[-5,5]$. It still doesn't like it. What about over $[-10,5]$, or even $[0,5]$? -- interesting. Any ideas why the computer would do this?

Prepared by Kerstin Nordstrom, 5/03
Modified by E. McCormack & D. Nice