Mathematica for the constantly busy
3rd Edition

Nathan Chapman

January 15, 2017

I dedicate this work to the people who strive for truth about our reality, otherwise known as scientists.

"Math: a lifetime of free entertainment" - Dr. Branko Ćurgus

Preface

On June 23, 1988, *Mathematica* 1.0 was released by Wolfram Research in Champaign, Illinois. Since then *Mathematica* , has gone through many updates and versions to the current *Mathematica* 11.0.1 edition, released on September 28, 2016, and has been implemented in numerous industries as well as academic sciences. These range from financial operations with big business on Wall Street to numerical and symbolic computation for theoretical physics research on Bose-Einstein condensates in Bellingham, Washington. This just goes to show how practical *Mathematica* can be, while still being genuinely beautiful.

Mathematica itself is a beautiful machine that is capable of doing outstanding things for the world. It can evaluate terrible integrals in a fraction of a second, solve differential equations faster than you can say "quidditch", and even make some quite perfect plots. Even not on the math forefront, *Mathematica* can handle and operate with incredible amounts of data, and serve as a dynamic data handler/controller for live experiments. While there are other programs out there that also do math on a very efficient level and are great for certain scenarios, like Matlab, STATA, R, Geometer's sketchpad, I believe *Mathematica* can do just about anything. (Here I'm thinking of things like mathematical analysis, or abstract algebra, or topics along the lines of higher, pure math)

Technically though, *Mathematica* is a combination of a front-end/GUI, notebook/cell-format, coding environment, which sends it commands and input to the kernel, which does the computation, then sends the result back to the notebook and displays it as best it can.

Here we will explore some of the basics of the glory that is *Mathematica* , ranging from doing simple arithmetic to solving partial differential equations and then plotting them in 3D.

This guide is meant to be either a quick reference, or a guide along the journey that is *Mathematica* .

Before we begin, I would like to point out the in-program, interactive documentation can easily be reached by simply pressing the F1 key at any time. If the cursor is on a command, it will automatically go to the page of that command.

Contents

I Syntax **9**

1 Capitalizing Functions **13**

2 Brackets **15**

3 Function Format **17**

3.1 Semicolons . . . 17
3.2 Strings . . . 17
3.3 Comments . . . 17
3.4 The equals symbol . . . 17
3.4.1 Programming Declaration . . . 17
3.4.2 Mathematical equality . . . 18
3.4.3 Delayed definition . . . 18
3.4.4 Clear . . . 18

4 Spaces, Constants, and % **19**

4.1 Spaces . . . 19
4.2 % . . . 19
4.3 Constants . . . 19

5 Exercises **21**

II Math! (Not factorial) **23**

6 Arithmetic **27**

7 Algebra **29**

7.1 Solve . . . 29
7.2 Trigonometry . . . 29
7.3 Sum . . . 29
7.4 Logarithms . . . 30

8 Calculus **31**

8.1 Limits . . . 31
8.2 Derivatives . . . 31

8.3 Integrals . . . 31

9 Linear Algebra **33**
9.1 Making Matricies and vectors . . . 33
9.2 Vectors . . . 33
9.2.1 Vector Products . . . 33
9.2.2 Gradient . . . 34
9.2.3 Vector Analysis . . . 34
9.2.4 Vector Operations . . . 34
9.3 Matrix Arithmetic . . . 34
9.4 Matrix Operations . . . 34
9.4.1 Inverse . . . 35
9.4.2 Transpose . . . 35
9.4.3 Determinant . . . 35
9.4.4 Row Reducing . . . 35
9.5 Eigen-stuff . . . 35
9.6 NullSpace . . . 35
9.7 Norm . . . 36
9.8 Dimension . . . 36
9.9 Solving matrix equations . . . 36
9.10 LeastSquares . . . 36

10 Differential Equations **37**
10.1 Solving ordinary differential equations . . . 37
10.2 Solving partial differential equations . . . 37
10.3 Numerically solving ordinary differential equations . . . 37
10.4 Using the solution . . . 38

11 Exercises **39**

III Data! (Not the Android) **41**

12 Reading in Data **45**

13 Handling Data **47**
13.1 Length . . . 47
13.2 Depth . . . 47
13.3 Select . . . 47
13.4 Indicies . . . 48
13.5 Flatten . . . 48

14 Exercises **49**

IV Important Functions 51

15 Plot 55
15.1 Basic Plotting . . . 55
15.2 "Plot" Options . . . 55
15.2.1 Labeling Axes . . . 55
15.2.2 Coloring your plot . . . 56
15.2.3 Smoothness . . . 56
15.2.4 How much you see . . . 56
15.2.5 Colors! . . . 56
15.3 3D Plotting . . . 57

16 We Make Table 59
16.1 Making simple sets . . . 59
16.2 Plotting your data . . . 60

17 Simplifying 61

18 Manipulate 63

19 Export 65
19.1 Making GIFs . . . 65

20 Exercises 67

V Graphics 69

21 Too many graphics 71
21.1 Hyper-hyper-cube . . . 72
21.2 Waterfall plot . . . 73

Part I

Syntax

Mathematica has some interesting syntax, though, once used enough, it all makes sense on why the initial developers chose to instigate such algorithms. In the subsequent chapters, we will see notation used in very specific ways, that may counter your previous knowledge of how such notation is used in other fields. For example the single square bracket notation used in *Mathematica* is used to denote the argument and options for functions, while in mathematical analysis, if a function encloses its argument with square brackets, it is assumed the argument is a set and the output of the function is the image of that set.

The syntax of *Mathematica* is usually the hardest part of the Wolfram language for beginning users to remember and fully understand. A quick overview for the syntax would be

- For every built-in function in *Mathematica* the first letter of each word in the function needs to be capitalized.
- Functions use single square brackets to enclose the argument and options.
- Parenthesis to denote mathematical multiplication and separation.
- Sets/tensors (scalars, vectors, and matrices) are denoted with curly brackets.
- Semi-colons are used to suppress output.
- Comments are done by enclosing the comment in $(* \cdots *)$.
- The equals symbol can be used to denote mathematical equality, programming declaration, or delayed definition.
- Spaces denote mathematical multiplication.
- The percent symbol, %, is used to denote the previous output.
- Special numbers like Pi, are denoted either by typesetting or capitalizing their name.

Chapter 1

Capitalizing Functions

The first letter of each word in every function in Mathematica is capitalized. e.g. Plot, ListPlot, DSolve, FindSequenceFunction, etc.

This might be the most frustrating thing for beginning users to remember. Sometimes I even forget to capitalize the letters relating to words in the middle of commands.

Regardless of feelings though, this is an important part of the syntax because it then offers users to define their own functions and commands with all lower case characters and not get confused, as well as providing a consistent syntax for Wolfram in naming the functions themselves.

Chapter 2

Brackets

- **Square brackets**

  Square brackets, [. . .], are used to enclose everything that is needed to work a function. e.g. Plot[. . .]

- **Parenthesis**

  Parenthesis are used for multiplication.

  e.g.

$$x(x+1) = x^2 + x$$

- **Curly brackets**

  Curly brackets are to group things, like sets and "function machinery".

  e.g. Let V be a vector space spanned by the set $\{v_1, v_2, v_3\}$ or Table[k, {k, 3, 10}]

These differences are very important because use of one where a different should be used will lead to errors, and your code not running the way you want it.

Chapter 3

Function Format

When using a function that calls for an argument and bounds, it usually take its arguments in the form

function[expression, {variable,lower_bound,upper_bound}]

3.1 Semicolons

When a semicolon, ";", is at the end of a function or line, it suppresses the output of that command.

For example, when you want to have a large set of astronomical data that consists of 5 columns with over 7000 entries in each column stored in memory, you can simply import the data, then put a semicolon after it, and it won't display the entire set of data. (When the set is big enough, *Mathematica* will shorten it down to a little display window)

3.2 Strings

When something is inside quotation marks, " ... ", they make it into a string, otherwise known as just text. This is needed for some options in functions like " ColorFunction → "Temperature" ".

3.3 Comments

Commenting in *Mathematica* can be done by simply enclosing whatever you want in "(* ... *)".

3.4 The equals symbol

There are multiple uses for the equals symbol in *Mathematica* , as well as other coding languages.

3.4.1 Programming Declaration

"=" is used to define the left side as the right side.
For example,

$$x = 2$$

means x is now defined as the value 2;
or

$$P = ax^3 + bx^2 + cx + d$$

means P is now the above polynomial.

3.4.2 Mathematical equality

"==" is used to denote mathematical equality.
For example,

$$2 == 2$$

will return True because 2 does indeed equal 2.

3.4.3 Delayed definition

":=" is used to define a variable on the left with an expression on the right each time it runs.
For example,

$$f[x\_] := \sin[\cos[x]]$$

means "$f[x]$" is now a function of the variable x which will be ran and displayed each time. (More on this later.)

3.4.4 Clear

The "Clear" function is used to clear any function or variable of its declaration, and is used by "Clear[...]".

You can clear multiple things at once by putting them all in the function argument and separated by a comma.

Another way of clear, or unsetting variables is to use "unset", which is just "=.".

For example, if $x = 42$, then we can clear it by using "$x = .$".

Chapter 4

Spaces, Constants, and %

In this chapter their wasn't enough content to warrant three different chapters, so I lumped them into one.

4.1 Spaces

Spaces between 2 things in *Mathematica* imply a multiplication between those 2 things.
For example,

$$x\ y = x * y$$

4.2 %

The percent sign, %, in *Mathematica* is used to represent the last output that was produced. This can be very useful, not only for efficiency sake, but also to make code look cleaner.A double percent, %%, represents the second to last output produced. Finally %n represents the output from input n. e.g. %3 represents the third output in the kernel.

4.3 Constants

Mathematica also knows almost all the special numbers out there, like π, e and ∞ (Is ∞ a number?) , etc. Here we'll see how to get these "exact" values. First, to typeset almost anything in *Mathematica* , you can push the escape key, type the name of the thing you want, then press escape again.

- π can be represented in *Mathematica* by either "Pi" or with escape, "p"
- e can be represented by either "E" or with escape by "ee"
- ∞ can be represented by either "Infinity" or with escape by "inf"
- i can be represented by either "I" or with escape by "ii"

Chapter 5

Exercises

Here will be a few exercises and problems to help with learning *Mathematica* . Most of these will work in any version of *Mathematica* , but some may only work in versions 8-10.

1. Use a function in the *Mathematica* catalog.
2. Make a function that multiplies the variable by a special constant.
3. Clear the function from the previous problem and make a function of two variables. Then make function that take a set as an argument.

Part II

Math! (Not factorial)

Math is an important part of *Mathematica* because if it wasn't, it probably wouldn't have "Math" in the name. *Mathematica* was originally based as computer algebra system (CAS) in order to do symbolic computation. This is the same concept as the TI-89 or TI-NSpire with CAS. Since it was first introduced, *Mathematica* has grown to be much more than just a simple CAS, but they didn't just stop updating that part of it. *Mathematica* now can do simply absurd things with symbolic and numeric computations, and it does it efficiently as well.

As with syntax, a quick overview of the mathematical computation capabilities *Mathematica* has to offer is

- Faster than lightning arithmetic (Adding, subtracting, multiplying, and dividing)
- Unbelievable algebraic manipulations like solving almost any equation, and even evaluate, to sometimes closed-form expressions, infinite sums.
- Make calculus look like arithmetic, by doing all kinds of derivatives and integrals of all kinds of complexities.
- Awesome power in doing all sorts of linear algebra computations, like finding the determinant, inverse, eigenspace, and solving matrix equations even when the method of least square is required.
- Solving really nasty differential equations of almost all kinds, including boundary eigenvalue problems.

Moral of the story is, *Mathematica* is really, really, really great at doing math.

Chapter 6

Arithmetic

To do arithmetic in *Mathematica* , all you have to do is type the numbers you want to add, subtract, multiply, or divide, in the form you want to compute, and hold "Shift" and press "Enter".

For example,

$$\begin{aligned} \text{input: } & 2+2 \\ & \quad \text{Shift + Enter} \\ \text{output: } & 4 \end{aligned}$$

Now, the real learning part here is the action of the Shift+Enter motion. In *Mathematica* 8 and later versions, commands can be done by also pushing the Enter button on the numeric keypad.

While this part may be a bit trivial, if you need to do arithmetic with lot of numbers at once, you probably don't want to use a simple calculator, or you want to store a bunch of number into a table to do calculations.

Chapter 7

Algebra

7.1 Solve

Mathemaica has a beautiful function that when given an equation and a desired variable in that equation it will solve that equation for that variable. Imaginatively enough, this function is called Solve.

For example

$$\text{input: } \text{Solve}[ax^2 + bx + c == 0, x]$$
$$\text{output: } x \to \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note: There are also variations on Solve to do slightly different things. For instance, there is also "NSolve" which numerically finds solutions to equations.

7.2 Trigonometry

Mathematica of course also has all your trig functions, Sin, Cos, Tan, their reciprocals, and their inverses.

Note: *Mathematica* automatically takes the argument of these function to be in radians. If you want the argument to be in degrees, simply put "Degree" after the argument bu still inside the function. e.g.

$$\sin\left[\frac{\pi}{2}\right] = \sin[90 \text{ Degree}]$$

Remember, these are functions so they need to be capitalized, but they are not here because LATEXuses them in lowercase.

7.3 Sum

Mathematica can certainly do both indefinite and definite sums even to infinity (Series)!

To take a definite sum we simply use

Sum[exp , {variable,lower_ bound,upper_ bound}]

For example

$$\begin{aligned}\text{In: } & \text{Sum}[p^2, \{p, 10\}] \\ \text{Out: } & 385\end{aligned}$$

7.4 Logarithms

Mathematica can use logarithms of any base, but it's standard "Log" function automatically uses base e.

To take a logarithm, we use

Log[b,v]

where "b" is the base of the logarithm, and "v" is the thing you're taking the log of.

Chapter 8

Calculus

8.1 Limits

Mathematica can evaluate even the nastiest of limits with ease.

To take the limit of the function $f(x)$ as x goes to a, $\lim_{x \to a} f(x)$, we use

$$\text{Limit}[f[x], x \to a]$$

8.2 Derivatives

Differentiating in *Mathematica* is actually quite easy. When you want to differentiate an expression you can simply use

D[f,x]

where f is the function you want to differentiate and x is the variable you want to differentiate with respect to.

Note: "D" gives the partial derivative of the function, so "f" doesn't have to be single variable.

"D" has a few variations that include multiple derivatives, and vector derivatives (gradients).

8.3 Integrals

For all those pesky indefinite integrals that would need several by parts, you can do it in a snap with the mathing power of *Mathematica* using "Integrate"!

To integrate something, we use

Integrate[f,x]

where "f" is the integrand, and "x" is the variable with which you are integrating with respect to.

For example

$$\text{input: Integrate}\left[\frac{1}{x^3+1}, x\right]$$

$$\text{output: } -\frac{1}{6}\log\left(x^2-x+1\right)+\frac{1}{3}\log(x+1)+\frac{\tan^{-1}\left(\frac{2x-1}{\sqrt{3}}\right)}{\sqrt{3}}$$

To do a definite integral, you would do the same thing, but put the variable and bounds in curly brackets.

e.g. Integrate[log[x], $\{x, -10, 10\}$]

Chapter 9

Linear Algebra

9.1 Making Matricies and vectors

There are 2 ways to make matricies in *Mathematica* , typesetting or using sets. Using sets is easier to enter, but doesn't look as pretty. I like the sets method.

In *Mathematica* sets are used as vectors, matricies and tensors. It all depends on how many levels there are.

For example, the set

$$\{v_1, v_2, v_3\}$$

is a vector, while the set of sets

$$\{\{u_{1,1}, u_{1,2}, u_{1,3}\}, \{u_{2,1}, u_{2,2}, u_{2,3}\}, \{u_{3,1}, u_{3,2}, u_{3,3}\}\}$$

is a 3 x 3 matrix.

9.2 Vectors

9.2.1 Vector Products

To take the dot product between two vectors simply put a period between them, $\vec{v}.\vec{b}$, or use the "Dot" function

$$\mathrm{Dot}[\vec{v}, \vec{b}]$$

In the instance you are working with abstract vectors in inner product spaces, you would use the "inner" function for the inner product, instead of "Dot".

To take the cross product between two vectors use the "Cross" function

$$\mathrm{Cross}[\vec{v}, \vec{b}]$$

9.2.2 Gradient

In order to find the scalar derivative of a vector, the gradient, you simply need to use

$$\text{Grad}[\vec{v}]$$

where $\vec{v}$ is a vector with scalar elements.

9.2.3 Vector Analysis

When diving into the world of vector analysis, you don't have to leave *Mathematica* behind. But in order to use the vector analysis functions, if you are using a version earlier than 10, you need to import the vector analysis package with

Needs[" VectorAnalysis' "]

If you are using version 10 or later, or after you import the package, you can find the divergence and curl with

$$\text{Div}[\vec{v}, \vec{b}]$$

and

$$\text{Curl}[\vec{v}, \vec{b}]$$

9.2.4 Vector Operations

A few essential vector operations that are worth mentioning are normalizing a vector and orthogonalizing a set of vectors.

To normalize a vector use

$$\text{Normalize}[\vec{v}]$$

To orthogonalize a set of vectors using Graham-Schmidt orthogonalization, use

$$\text{Orthogonalize}[\vec{v_1}, \vec{v_2}, \ldots, \vec{v_n}]$$

9.3 Matrix Arithmetic

Matrix Arithmetic is probably just as you would expect, when you add them, they add as they should, component wise. Same for subtraction. When multiplying matrices, you need to use "." instead of "*", otherwise it will multiply component wise.

9.4 Matrix Operations

Mathematica of course also has the capability to do many matrix operations, like taking the inverse, transpose, determinant, etc.

9.4.1 Inverse

When taking the inverse of a matrix, it's actually pretty simple using "Inverse". (Imagine that!)

To take the inverse of a matrix, you simply use "Inverse[...]". Remember, you can only take the inverse of a square matrix.

For example, let B be the matrix

$$\begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

Then

$$\text{in: Inverse}[B]$$

$$\text{out:} \begin{pmatrix} \frac{b_{2,2}}{-b_{1,2}b_{2,1}+b_{1,1}b_{2,2}} & -\frac{b_{1,2}}{-b_{1,2}b_{2,1}+b_{1,1}b_{2,2}} \\ -\frac{b_{2,1}}{-b_{1,2}b_{2,1}+b_{1,1}b_{2,2}} & \frac{b_{1,1}}{-b_{1,2}b_{2,1}+b_{1,1}b_{2,2}} \end{pmatrix}$$

9.4.2 Transpose

Similar to "Inverse", "Transpose" is as easy as it should be. Simply write "Transpose[...]", where the argument of "Transpose" is a list.

9.4.3 Determinant

Amazingly enough, you can simply find the determinant of a square matrix using "Det[...]".

9.4.4 Row Reducing

To quickly row reduce a matrix, simply use "RowReduce[...]" where the argument is a matrix.

9.5 Eigen-stuff

Another very useful thing *Mathematica* is great for is finding the eigen: values, vectors, and space, of a matrix.

The syntax for the functions "Eigenvalues","Eigenvectors","Eigensystem" is all the same.

"Eigen-thing[...]" will give you the desired result for the input square matrix.

Note, "Eigensystem" will give a set of sets where the first element in each set is an eigenvalue and the second element is the corresponding eigenvector.

Another note, *Mathematica* will give each eigenvalue as if it were distinct. i.e. It will display multiple of the same eigenvalue if that value has multiplicity greater than zero.

9.6 NullSpace

To find the set of vectors that span the null space of a matrix A, we use "NullSpace[A]".

9.7 Norm

To know the magnitude, or Norm, of a number, vector or matrix, you simply use "Norm[. . .]".
Note, this includes complex numbers.

9.8 Dimension

To know the dimension of a matrix we simply do

Dimensions[. . .]

This will give the length 2 set whose first element is the number of rows the matrix has and the second element is the number of columns the matrix has.

9.9 Solving matrix equations

To solve a matrix equation, we can use "LinearSolve". "LinearSolve" take 2 arguments, a $n \times m$ matrix A and a vector $b \in \mathbb{R}^m$, where $A\vec{x} = \vec{b}$. In practice, the function takes the form

LinearSolve[A,b]

9.10 LeastSquares

Just like solving the matrix equation, we can also use the least squares method by using

LeastSquares[A,b]

where A is a matrix and b is the solution to the equation $A\vec{x} = \vec{b}$.

Chapter 10

Differential Equations

Mathematica is also perfect for linear ordinary differential equations with constant coefficients, but also does a pretty good job against other kinds as well!

10.1 Solving ordinary differential equations

When you have a really nasty differential equation and you completely have forgotten how to solve even the simplest form, you can use *Mathematica* s function "DSolve".

To use "DSolve" we write the form

DSolve[eqn,y[x],x]

where *eqn* is the equation we want to solve, $y[x]$ is the function we're solving for, and x is the independent variable of the desired function.

If you want to find the "pure" function solution y you can use

DSolve[eqn,y,x]

10.2 Solving partial differential equations

It doesn't stop at ordinary differnetial equations though, *Mathematica* can even do partial differential equations in a snap!

To solve a partial differential equation in *Mathematica* , we still use "DSolve" but now we use

DSolve[eqn,y[x],$\{x_1, x_2, \dots\}$]

10.3 Numerically solving ordinary differential equations

There is also "NDSolve" which solve differential equations numerically and is used

NDSolve[eqn,y[x], { x ,lower bound ,upper bound }]

It should be noted *Mathematica* does best with linear ordinary differential equations with constant coefficients, but can do most with non-constant coefficients up to second order.

10.4 Using the solution

If you want to use the solution to the differential equation in something else, we can easily use the replace command.

Say we get a pure function solution y to a differential equation, and now we want to use that equation in another equation $y(x) + \sin(y(x))$. To easily substitute the solution $y(x)$ into the equation, we would use

$$y(x) + \sin(y(x))/.\text{DSolve}[eqn, y, x]$$

This will replace every instance of the character y with the solution and simplify.

Chapter 11

Exercises

1. Analytically solve an algebraic equation. Ex: $\pi x^2 + ex + \phi = x$
2. Numerically solve a triginometric equation.
3. Add the first 100 natural numbers. Ex: $1 + 2 + 3 + \cdots + 100$
4. Find the closed form solution to the series of a n-th degree polynomial. Ex: $1 + x + x^2 + \ldots$
5. Find the derivative of the solution to the previous problem.
6. Find the derivative of the factorial function.
7. Find the integral of the solution to #6.
8. Create a 2×2 matrix, a vector in $\mathbb{R}^2$. Then multiply them.
9. Find the determinant of the inverse of the transpose of a matrix of your choice.
10. Row reduce a 10×10 matrix of odd numbers.
11. Find the dimension of the nullspace of the space spanned by the eigenvectors of a matrix
12. Solve a matrix equation whose matrix is composed of complex numbers and the solution is composed of real numbers.
13. Solve the differential equation

$$A\frac{d^2u}{dx^2} + B\frac{du}{dx} + Cu(x) = u(x)$$

with the initial conditions of $u(0) = 0$ and $\frac{du}{dx}(0) = \Gamma$.

14. Solve the partial differential equation

$$A\frac{\partial u}{\partial x} + B\frac{\partial u}{\partial t} + Cu(x,t) = u(x,t)$$

with the intial condition $u(x,0) = f(x)$.

15. Using the solutions to problems 16 and 17, evaluate new expressions.

Part III

Data! (Not the Android)

Not only can *Mathematica* do awesome symbolic and numeric computations, it can also handle massive amounts of data!

As with the others, a short overview is in order

- *Mathematica* can easily read in a file either from the computer the on which the notebook is running, or from the internet.
- It's no problem for *Mathematica* to speedily produce the length of the data set.
- If your data has multiple levels, you can easily see how many levels down it goes, or how deep it is.
- If your data has multiple levels, but you don't want it to, you can flatten it to make it all on the first level.
- If you only parts of the data that match some sort of pattern, you can simply select those elements.

Chapter 12

Reading in Data

First, we need to import the data, with guess what, "Import".

To use "Import" we enter

```
Import["file.format"]
```

Now here the quotations are necessary and the file either needs to be in the same folder as your notebook, or you need to specify the entire path in the file name.

I usually like to set my data as the variable "Data" so I can use it easily in later commands.

Import can take many different formats of files, but the main one we'll focus on is CSV (Excel) since many it is usually the format scientific data is taken in for our purposes.

Important Note: If you don't want to see the data remember to put a semicolon after the closing square bracket.

Chapter 13

Handling Data

We don't just want to read in a bunch of data and not do anything with it! Now we can look at the data.

13.1 Length

To find how many elements there are in the first level of your data, we simply use

$$\text{Length}[\ldots]$$

13.2 Depth

To find how many levels there are in the data, we simply use

$$\text{Depth}[\ldots]$$

Now, since *Mathematica* automatically make the data into one big set, the depth is how many indices are needed to specify the location of an element in the set + 1.

13.3 Select

Let's say you only want the data that meet certain criteria, then you would use the "Select" by

$$\text{Select[Data, criteria]}$$

For example, to find the evens out of $Data = \{1, 2, 4, 7, 6, 2\}$, we use.

$$\begin{aligned} \text{in: } & \text{Select}[Data, \text{EvenQ}] \\ \text{out: } & \{2, 4, 6, 2\} \end{aligned}$$

If we want to find the data that match a more abstract criteria we can use pure functions.

For example, to find the data that is greater than 2, we use

$$\begin{aligned}\text{In: } & \text{Select}[Data, \# > 2\ \&] \\ \text{Out: } & \{4, 7, 6\}\end{aligned}$$

In this case, the # is used to denote the element of the set that is being tested and the & is used to make the expression a pure function.

13.4 Indicies

To actually look at an element in the set, or piece of data in the set, we can use double square brackets, "[[" and "]]".

To make good looking opening index brackets we use (Escape) [[(Escape). The same goes for closing brackets.

So, to see the a-th element in the b-th set in Data, we say

$$\text{Data[[b]][[a]]}$$

For example, let $Data = \{\{1, 2\}, \{3, 4\}\}$.

Then

$$\begin{aligned}\text{In: } & \text{Data[[2]][[1]]} \\ \text{Out: } & 3\end{aligned}$$

This can be extended to higher (deeper?) depths by adding more indices with brackets.

13.5 Flatten

Let's say you have a set with multiple levels, but you want the data to be all on the first level. In this case, we can use "Flatten".

To use "Flatten" we simply say

$$\text{Flatten}[\ldots]$$

Chapter 14

Exercises

For these exercises, create a data set with the following code

```
Data = Table[{k,1/k},{k,100}];
```

1. Find the length of Data.
2. Find the depth of Data.
3. Select the elements of Data that are greater than 1.
4. Find the length and depth of the solution to exercise 3.
5. Create a set of data which only has one level. Then repeat exercises 1-3 for this new data set.

Part IV

Important Functions

Out of all the functions *Mathematica* has to offer, I have a few that I think are particularly important and useful because they have been just that in my experience. I'm sure there are a lot of other really amazing functions and commands that *Mathematica* has to offer, but I only have a little bit of experience with it. This is why I encourage you to explore on your own to find the wonders of *Mathematica* for yourself.

And here comes the obligatory overview

- One of the functions that makes *Mathematica* one of the best tools ever is easily plotting both 2D and 3D graphs, then customizing your plots an unfathomable amount.
- *Mathematica* is perfect for producing tables of objects that can be defined sequentially, or according to some pattern. Then plotting it.
- Another can't-do-without for *Mathematica* is its ability to simplify almost any expression it's given down down to probably its most simplified form we know.
- *Mathematica* gives us the power to manipulate almost anything we throw at it, that is we can control some variable for an object and see how it changes in real-time.
- *Mathematica* also is perfect for getting your results out into the world in helpful formats by exporting them into almost any format you want.

Chapter 15

Plot

15.1 Basic Plotting

Plotting is a very big deal in the science world. With the help of *Mathematica* and it's gigantic library of options, you can make the best plots.

The basic Plot function is rather simple to deal with since it takes the form

$$\text{Plot}[f[x], \{x, lb, ub\}]$$

where "lb" and "ub" are the lower and upper bounds respectively.

"Plot" in this form will still give you a beautiful plot of whatever function you give it to graph, but it could be better. Let's check out some of the options ol' Stephen gave us.

15.2 "Plot" Options

To implement "Plot" options, they need to be in the function, but after everything the function needs.

For example

$$\text{Plot[} x^2\text{, } \{x, -1, 1\}\text{, "Options go here each separated by a comma"]}$$

15.2.1 Labeling Axes

To label your axes and not just have some random graph, use "AxesLabel".

"AxesLabel" take the form

$$\text{AxesLabel} - > \{x, y\}$$
$$y$$

where x is the label for the "x" axis and "y" is the label for the y axis.

When "AxesLabel" just points to "y", *Mathematica* will just label the "y" axis.

15.2.2 Coloring your plot

To color your plot you can use "ColorFunction". Since there are quite a few implementations for "ColorFunction", I will let you explore these on your own.

15.2.3 Smoothness

Mathematica actually makes its plots by connecting a finite number of points, because it's a computer and they can't deal with infinities. Since this is how plots are made, you can set the number of points *Mathematica* uses to plot using, imaginatively enough, "PlotPoints".

To make your plot as smooth or as jagged as you want, you can use

$$\text{PlotPoints} \rightarrow \#$$

where # is the number of desired plot points.

Having a high number of plot points will certainly make your plot as "smooth", or as accurate to the original function, as possible.

Note: There is a point at which the plot will not change due to the number of plot points. Also, the run time of the code is proportional to the number of plot points.

15.2.4 How much you see

"Plot" intrinsically takes bounds on which to plot the function, but you can set the plot to show more of the viewing window with "PlotRange".

To use "PlotRange" we say

$$\begin{aligned} \text{PlotRange} - > & \{\{-x, x\}, \{-y, y\}\} \\ & \{-y, y\} \\ & y \end{aligned}$$

where "-x" and "x" are the left and right x axis plot bounds respectively, and "-y" and "y" are the bottom and top plot bounds respectively.

15.2.5 Colors!

To make your plot not just a thing blue line, we can use "PlotStyle".

Like "ColorFunction", "PlotStyle" has a lot of options, so I will cover what I think are the most important that I know.

To implement "PlotStyle" we use

$$\text{PlotStyle} \rightarrow \{\text{Thickness[t],Color}\}$$

where "t" is the thickness of the line of your function, (Generally you would want this around .01 for a good solid line, but try it at other values to see what happens.) and "Color" is the color you want your plot to be. *Mathematica* knows almost all colors by name, but can also recognize colors with specific RGB values. RGB valued colors are represented by "RGB[r,g,b]" where "r" is the red portion of the color, "g" is the green portion of the color, and "b" is the blue portion of the color.

15.3 3D Plotting

Mathematica can also make beautiful 3D plots using "Plot3D".

"Plot3D" is almost the exact same as its 2D counterpart, including its options, except for it needing 2 variables to bound the plot.

To use "Plot3D" we enter

Plot3D[f[x,y], {x, xmin, xmax}, {y, ymin, ymax}]

where "f" is the function to be plotted, "x" is one of the function's variables, "xmin" and "xmax" are the lower and upper bound of the x-axis respectively, "y" is the other variable of the function, and "ymin" and "ymax" are the lower and upper bounds of the y-axis respectively.

Chapter 16

We Make Table

"Table" is such a beautiful function because it holds so much power in such a tiny little container. With "Table", we can make enormous sets or sets of sets of sets, or do math with elements of a set with elements of another set, and so on.

16.1 Making simple sets

Like I said before, we can make sets with "Table" by entering

$$\text{Table}[\ expr, \{k, kmin, kmax\}\ ]$$

where "expr" is some expression involving "k" and the expression will be generated $kmax - kmin$ times starting at "kmin" and ending at "kmax", incrementing in steps of 1.

For example, we will make a simple set of the naturals from 1 to 10.

To do this we use

$$\begin{aligned} \text{In: }& \text{Table}[k, \{k, 1, 10\}] \\ \text{Out: }& \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\} \end{aligned}$$

Now if you're starting at 1, you can actually drop it from the function so it reads

$$\text{In: Table}[k, \{k, 10\}]$$

Too simple? How about

$$\begin{aligned} \text{In: }& \text{Table}[Fibbonacci[n], \{n, 3, 10\}] \\ \text{Out: }& \{2, 3, 5, 8, 13, 21, 34, 55\} \end{aligned}$$

We can even have it take different step sizes by simply adding the step size to the end of the bounds!

$$\text{In: } \text{Table}[Fibbonacci[n], \{n, 3, 100, 17\}]$$
$$\text{Out: } \{2, 6765, 24157817, 86267571272, 308061521170129, 1100087778366101931\}$$

Now we get the Fibonacci numbers from 3 to 100 in steps of 17.

Now let's see what happens when we put curly brackets around the bounds.

$$\text{In: } \text{Table}[Fibbonacci[n], \{n, \{3, 100, 17\}\}]$$
$$\text{Out: } \{2, 354224848179261915075, 1597\}$$

Oh! It made a table of the 3rd, 100th, and 17th Fibonacci numbers! So this tells you can make "Table" iterate on values denoted in the given set.

16.2 Plotting your data

- **1 to 2 elements**

  To plot the data, or elements of a set, as points with elements of your data being the coordinates of each point, we use "ListPlot".

  To use "ListPlot", we enter

  ListPlot[Data]

  If the data is a set of individual elements, those elements will be plotted in order with indices corresponding to succesive naturals 1, 2, ...

  If the data is a set of sets with 2 elements, the first element in each set will be the x-coordinate and the second element will be the y-coordinate.

- **More than 2 elements**

  When dealing with data that has more than 2 elements in each set, or has more 2 levels, we need to make a table of the just elements we want. To do this, we need to use the indices of the elements in the data, Data[[a]][[b]].

  e.g. If we want to make a plot of the data from the 1st and 3rd elements in each set within "Data", we would say

  ListPlot[Table[{Data[[t]][[1]],Data[[t]][[3]]}, {t, Length[Data]}]

Chapter 17

Simplifying

One of the best functions in *Mathematica* is "FullSimplify". With "FullSimplify", you can input an expression or equation and it will go through every identity and standard reduction it has and apply it to the expression until it has reached the simplest form of the expression.

For example,

$$\begin{aligned} \text{input: } & \text{FullSimplify}\left[i\sin^3[\theta] + \cos^3[\theta] + i\sin[\theta]\cos^2[\theta] + \sin^2[\theta]\cos[\theta]\right] \\ \text{output: } & e^{i\theta} \end{aligned}$$

Another version of this function is just "Simplify". This version goes through much fewer possible reductions, but is a lot faster.

Chapter 18

Manipulate

Another key function in *Mathematica* is "Manipulate" which allows you to, well, manipulate anything you want. This applies to plots, graphics, and almost anything you can think of.

To use "Manipulate", we can enter

$$\text{Manipulate[} stuff, \{var, varmin, varmax\} \text{]}$$

where "stuff" is an expression which has the variable "var" acting on it in some way, and "varmin" and "varmax" are the minimum and maximum of "var" respectively.

For example

$$\text{Manipulate[Plot[} \gamma x^2, \{x, -1, 1\}, \text{ PlotRange -> } \{-1, 1\} \text{], } \{\gamma, -1, 1\} \text{]}$$

gives us the power to manipulate the curvature of the produced parabola from "Plot".

Chapter 19

Export

The “Export" function is amazing if you want to turn an output into another file. For this instance I will focus on how to make GIFs with *Mathematica* .

First, to use “Export" in the simplest way, enter

Export[“file.format",expr]

where *file* is the name of the exported file, *format* is the format of the exported file, and *expr* is the code whose output will be exported.

Important: *Mathematica* will export all files to the default directory. To fix this, save the notebook, then enter

SetDirectory[NotebookDirectory[]]

at the top of the entire notebook to export your files to the same folder the notebook is in.

19.1 Making GIFs

An excellent use of *Mathematica* is to make GIFs of manipulated plots.

To do this, make a manipulatable plot, but instead of using the “Manipulate" function, use “Table". Make sure to adjust the bounds of manipulation and step size accordingly. Then use the output as the expression in the “Export” function.

When it’s finished running, go to the file in the system explorer and open it to see your GIF!

Important: If the result is a lot of images, make sure to suppress the output of the table with “;".

Chapter 20

Exercises

1. Create a plot of your favorite function with the axes labeled appropriately, the line your favorite color, the graph very jagged, and blank space on either side of the graph.
2. Create a 3D plot of a sphere with similar options as above.
3. Create a table of odd numbers between 3 and 171.
4. Create a set of pairs of real numbers, such that when plotted they form a line.
5. Simplify and then fully simplify the solutions to exercises 6-8, 14-16 from section 13.
6. For the solutions to each of the parts to the previous problem, create a scaling coefficient on some part of the solution. Then create a manipulatable plot for each solution and explore the different results.
7. Make a GIF of a parabola that changes its curvature.

Part V

Graphics

Chapter 21

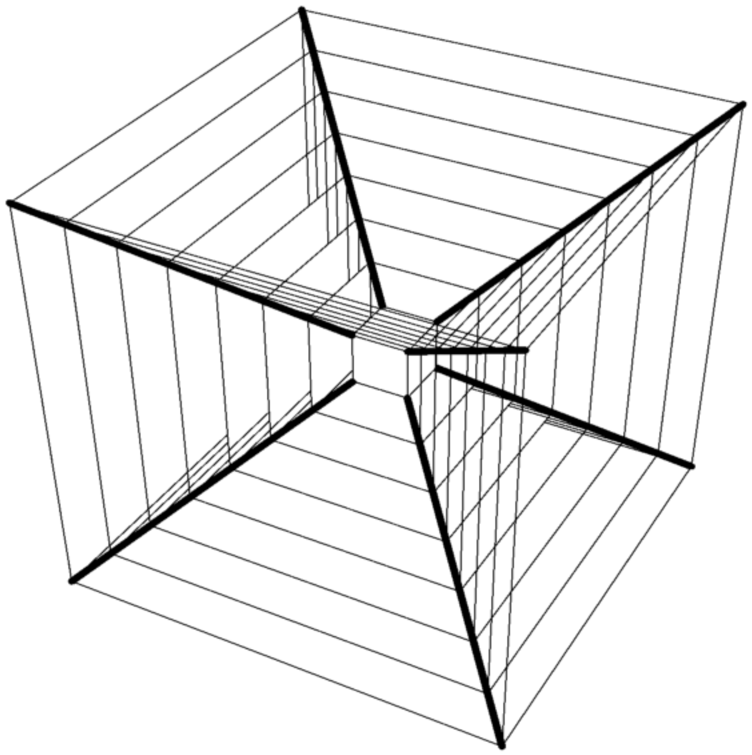
Too many graphics

Mathematica is not only a number cruncher, but also an amazing graphical representation device. Using "Graphics" and "Graphics3D" we can construct almost any shape or object possible with as many different options to make our productions as specific as we want them. For example, with the introduction of chemical data, we can produce accurate, detailed, 3D representations of molecules. Both fortunately and unfortunately, there are way too many different possible graphics and options that I wouldn't even know where to begin in providing examples.

Since there are so many different options and examples that could be shown, I suggest looking at the “Graphics" help page in the documentation. Under “More Details" will be examples of what can be done with “Graphics". But I don't want to cheat you, so here is one and a half examples of some of my own work that creates a 2D picture of a 3D representation of a 4D hypercube with multiple layers, and a waterfall plot of the first few nestings of the tent function. The base code for the waterfall plot was taken from Mr. Wizard on the *Mathematica* stackexchange, but I tweaked the rest to my needs; basically I did not come up with the code from scratch.

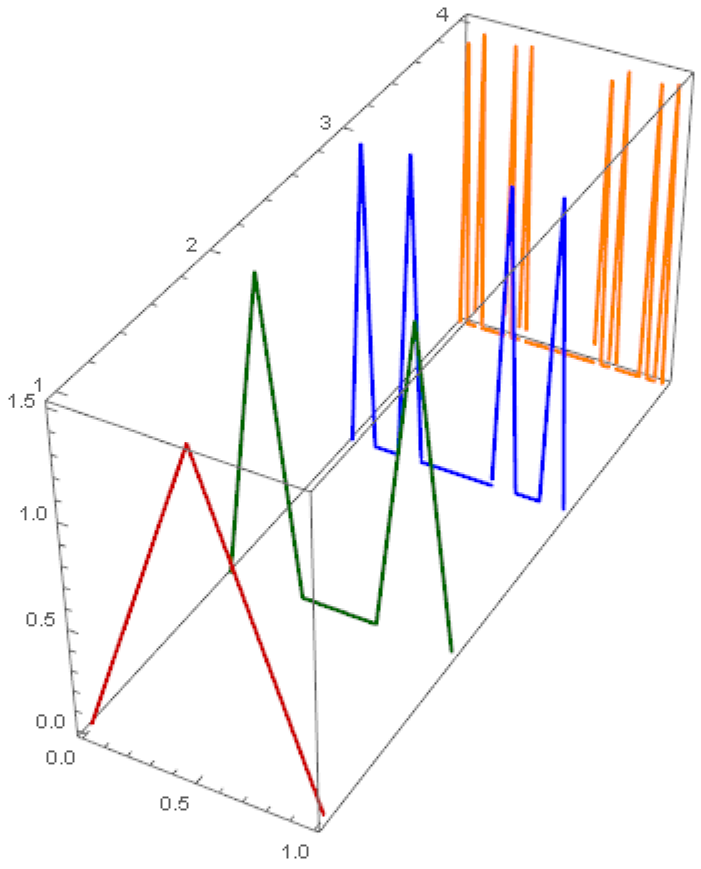
21.1 Hyper-hyper-cube

```
Graphics3D[
{
{Opacity[0],(*Cubes*) Table[{Cuboid[\mu{-1,-1,-1},\mu{1,1,1}]},{\mu,s,2,d}]},
{Thickness[.01],
(*Lines*) Table[{Line[\gamma]},{\gamma,Table[{ones[[k]],2/s ones[[k]]},{k,Length[ones]}]}]}
},
ImageSize->Large,
Boxed->False,
PlotRange->All,
AspectRatio->1
]
```



21.2 Waterfall plot

```
ParametricPlot3D[
Evaluate[
Table[
{x,\mu,Nest[Tent,x,\mu]]},
{\mu,4}
]
],
{x,0,1},
PlotPoints->600,
MaxRecursion->1,
ViewPoint->{1.3,-2.4,2},
PlotStyle->Table[{Thickness[.007],c},{c,{Darker[Red,1/5],Darker[Green,3/5],Blue,Orange}}],
ImageSize->Medium
]
```



Conclusion

Mathematica certainly can do a lot, but unfortunately, like I said in the introduction, it can't do absolutely everything. Sometimes good-ol'-fashion elbow grease and actual math skills are required to do what you need. Sometimes, you just need to get to a certain point, then you can use *Mathematica* to help. For instance, when solving a partial differential equation with boundary and initial conditions that *Mathematica* doesn't want to evaluate as is, you need to use separation of variables first to get system of ordinary differential equations to solve with the boundary conditions. When this time comes, and it will, don't fret, remember math is fun! I can assume you like math because if you didn't like math, you probably wouldn't be reading this.

In addition to *Mathematica* being used all over the world numerous different purposes, *Mathematica* is also used here in our very own back pocket by some faculty in both physics and math in their research.

This is just the tip of the iceberg that is the power of *Mathematica* and almost all of the functions have more to them than presented here. More information on each of these can be found by searching them in the *Mathematica* dialog or from "reference.wolfram.com/language" where all the past and newest features of *Mathematica* are.

Have fun!

Nate Chapman
Physics Sophomore
Math Senior
Western Washington University
January 15, 2017