Technological Fluency Labs for Math and Science Students, Version 0.1

Len Brin, Joe Fields, Younhee Lee and Ray Mugno

Southern Connecticut State University

Contents

Preface

The goal of this book is to help you learn about a variety of computational tools that will be useful in your career as a mathematician or scientist.

1 Introduction

This book aims to introduce the reader to several computational tools that are useful for people entering careers in Mathematics and the Sciences. We have chosen to emphasize systems that are open-source but we generally indicate other options in the introduction to each chapter.

The book is divided into 8 main chapters.

- This Introduction.
- Writing with embedded mathematical content.
- Computational Algebra.
- The creation and inclusion of Graphics in technical documents.
- Interactive Geometry.
- A chapter on the use of Spreadsheets.
- Statistics
- Miscellaneous

In each of those chapters there will be technological as well as mathematical learning goals. The point of this book is to collect laboratories/activities that will allow one to reach these goals through an active learning strategy: doing stuff, not being told about it!

An important aspect of developing a facility with tech is that you get better at it the more you do it. Many people advocate that "learning how to learn" is one of the biggest benefits of a college education. So we hope that part of the takeaway from doing these labs is that it will be easier for you to figure out other tools that you'll encounter in the future. For example, a great many computer applications allow you to access their functionalities via a menu system. Many of those menu entries have a keyboard shortcut associated with them. Once you see the increased productivity you get from using such shortcuts, you start to look for them in other applications. And, there are fairly consistent conventions about how those shortcuts are selected so knowledge you gain in one app often transfers over directly to others!

In the early days of computing, someone introduced the initialism RTFM which doesn't really stand for "Read that fine manual." That's one piece of advice that you should certainly take away from the TFLabs experience – learn how to access and read the help facility! A couple of other suggestions for getting up to speed quickly in a new computing environment:

- Often, the help for a particular command will include examples. Scrolling past that wall of confusing text until you get to the relevant examples isn't a terrible strategy.
- See if there are so-called "tool-tips" – hovering your mouse over a button often causes a pop-up that displays a hint about what the button does.
- Look for the aforementioned keyboard shortcuts. A lot of programs use Ctrl-Z as a shortcut for "undo" (this can be super useful! Unless the command is Ctrl-U in your particular application.)
- Learn to read error messages. These messages are generated programattically when something goes wrong - this usually makes them a bit cryptic. If you're able to crack the code, you'll probably at least get a hint as to what caused the problem.
- Use your favorite search engine to see if there is relevant help already out there. "Hey Siri, how do I get an ampersand in LaTeX?" (I really don't know if that will work, but putting that question in my browser's search bar sure does!)
- Doing the previous "googling" will often lead you to online help fora like Quora, Reddit and Stack Overflow. Reading threads in a forum can be really helpful. Posting questions in a forum is hit-or-miss – sometimes you'll get a quick useful answer, other times the result is not what one would want...

Let's get started!

2 Writing

2.1 HTML

Have you ever seen a web URL like

`https://www.agnesscott.edu/lriddle/women/love.htm`

and wondered what the `https` or the `htm` meant? In both `https` and `htm`, the `ht` is short for hypertext. The `tp` in `https` is short for transfer protocol and the `s` is for secure. The `m` in `htm` is for markup. All webpages, when it comes down to it, are made of HTML (hypertext markup language) code. The HTML tells the browser what should be displayed and, in general terms, the desired layout. The browser takes care of the fine details depending on the type of device, size of the screen, or personal preferences set by the user.

Markup in HTML is like a note to the browser telling the browser how to display the following content. The markup won't be displayed, but it will modify how the content will be displayed. Take this screenshot from `https://www.southernct.edu/` for instance.

Simplified just a bit, the HTML code that produces the “Back to Campus” announcement above looks like this:

```
<img src="/sites/default/files/back-to-campus-banner.jpg" /><p style
="font-size: 18px;">As the university prepares for the start of the
fall 2021 semester, we remain #SouthernStrong, with <a href="https://
inside.southernct.edu/reopening#services">all of the following academic
offerings and campus services</a> in place for our students in a safe,
engaging campus environment. Our goal is to get you off to a great
start, whether you're returning to campus or joining our community for
the first time!</p>
```

It is just simple text! No bells, no whistles. Parts enclosed by < and > are the markup. These parts will not be displayed themselves but rather describe how something is to be displayed and leaves the rest up to the browser. For example,

```
<img src="/sites/default/files/back-to-campus-banner.jpg"/>}
```

tells the browswer to insert an image (img) and tells the browser where to get the image (src). This part of the code creates

The rest of the code creates the paragraph underneath this image. It starts with `<p style="font-size: 18px;" >` — which tells the browser to start a paragraph (p) using an 18 pixel font—and ends with `</p>`, which tells the browser this is the end of the paragraph. In the middle of the paragraph is what makes webpages webpages—a hyperlink. The text between `<a href="https://inside.southernct.edu/reopening\#services" >` and `</a>` is marked up as a link, it's displayed in blue so the reader can see that these words are where they can click. Clicking them will bring up a new webpage in the browser.

Lab: 1
Working with markup

<HTML>
...
</HTML>

To see markup in action, point your browser to

https://htmledit.squarefree.com/,

where you can write some hypertext markup and see how it looks on your browser. The blue box at the top holds the markup, and it can be edited by you! The box below shows how the browser renders the markup. Do the following exercises on the squarefree webpage and answer the questions.

1. Insert before the word magically and insert after the word magically. What did this accomplish? Note: em is short for *em*phasis!

2. Copy and paste the following code into the webpage.

```
<p>A list of some common HTML markup
<ol>
 <li><tt>p</tt> is short for <b>p</b>aragraph</li>
 <li><tt>a</tt> is short for <b>a</b>nchor (which can
indicate a link or a place to link to)</li>
 <li><tt>ol</tt> is short for <b>o</b>rdered <b>l</b>ist</li>
 <li><tt>li</tt> is short for <b>l</b>ist <b>i</b>tem</li>
</ol>

Notice that markup can be nested -- the b and /b tags above are
inside the li and /li tags, which are between the ol and /ol
tags.</p>
```

3. What happens to text between the and tags?

4. What happens to text between the <tt> and </tt> tags?

5. Now change the <ol> and tags to <ul> and tags. What happens to the displayed page (in the white box)? Note: ul is short for **u**nordered **l**ist.

6. Notice how the b and /b tags in the last sentence are missing the usual angle brackets? What happens if you put them in?

7. This is a problem in all markup languages – some characters have special meaning. In HTML there are so-called escape characters to work around this issue. Google the phrase “html escape characters” and see if you can re-write the last sentence so that things **look like** a tag, but don’t **act like** a tag!

2.2 LaTeX

Donald Knuth is an amazing Zen master of Computer Science and Mathematics. He created a program called TEX(pronounced like "tech") for typesetting – especially typesetting with mathematical content. Knuth wrote TEXmainly as an example of a style of coding that he called "literate programming." Essentially this means that the programmer creates the documentation for their work at the same time they write the actual code. Whether it's because of it being written in the "literate programming" style or simply because of Knuth's amazingness, TEX is renowned as one of the most bug-free systems in existance.

TEX , and a follow-on program known as LATEX , which builds on TEX and extends it, have become the standard(s) for communicating mathematics. We'll be talking only about LATEX from here onward. LATEX is much like HTML – it is a sort of markup language that gets processed and turned into a document suitable for visual display. The appearance of LATEXsource code is very different from that of HTML source code, but in many ways they are incredibly similar! There are two marked distinctions: HTML is meant to describe how to lay out a page so that the content is clear, and do so on many different displays – the web-browser is empowered to alter things to aid in that clarity. On the other hand, LATEXis much more focussed on layout – the first thing one does in a LATEXdocument is to specify the page size. The author of a LATEXdocument can closely control where things will end up on that page[1]. The other big difference between these systems is how well they handle mathematical content. LATEXis very, very good at rendering mathematics and mathish notation from other fields like Chemistry. HTML stinks at that, which is completely weird since the original HTML standard was created by a physicist at CERN, and physicists tend to use a lot of math...

In a nutshell, HTML (the markup language of webpages) is used to control how a webpage appears. LATEX plays the same role for printed documents like books, reports, letters, or labs (like this one!). LATEX is a markup language that excels at creating technical documents, especially those that include mathematical formulas. You might want to read this blog all about why you want to learn LATEX.

Like HTML, LATEX uses tags to mark how text should look, to insert graphics, to create lists, and so on. Markup in LATEX begins with a \ (a backslash) as in `\pi`, which would be used to render our friend, π. Every LATEX document begins with the `\documentclass` tag, which specifies what type of document is to be created (book, article, report, etc.). This is similar to HTML documents, which all begin with the `<html>` tag and end with

[1]There is a strong argument against doing so – the author should really concentrate on content, and let the software control the individual pixels on the page.

the `</html>` tag. After the documentclass tag a LaTeX document must include the tags `\begin{document}` and `\end{document}`.[2] As you can probably guess, these tags mark the beginning and end of the content of the document. One of the simplest LaTeX documents possible is this one:

```
\documentclass{article}
\begin{document}

Hello World!

\end{document}
```

Just like HTML code, it isn't pretty! However, it tells the LaTeX renderer what to do. This code will create an "article" with the sentence "Hello World!" in it.

Let's give it a try!

The next lab will lead you through the process of creating LaTeXdocuments on a web-based service known as Overleaf.

[2]This is similar to HTML documents, in which the beginning of the content is marked by `<body>` and the end by `</body>`.

Lab: 2
Getting started with LaTeX

LaTeX

1. Go to https://www.overleaf.com/register
2. Register for a free account (probably best to use your university email.) You'll have to create a password at this point - make a note of it.[3]
3. It's probably best to skip their "Try the premium version for free" offer.
4. Click the "Create First Project" button – choose a blank project and name it "hello."
5. You'll see two main panels (there's also some junk above and to the left, but ignore all that for now.) The left-hand panel contains the LaTeX source code for your project and the right-hand panel gives a preview of the resulting document.
6. The "blank project" isn't completely blank. The source code panel will be pre-populated with:

```
\documentclass{article}
\usepackage{graphicx} % Required for inserting images

\title{hello}
\author{myemail}
\date{August 2023}

\begin{document}

\maketitle

\section{Introduction}

\end{document}
```

[3]Overleaf isn't exactly a "high stakes" setting, your password needn't be super complicated – just don't re-use a password that protects a more critical account!

You should see your document on the right side. It has a title section (which is what the `\maketitle` command created) and a section heading (this is what the `\section{Introduction}` command did). Scroll down to the bottom of the page and you will see the number 1, the page number. Pages of articles are normally numbered, so LaTeX puts that in for you! The area between the `\documentclass{article}` and the `\begin{document}` tags is known as the **preamble** of the LaTeX document. You should see that the preamble contains some commands that effect how the title looks. The default stuff that Overleaf stuck in there probably isn't quite what you want.

Let's fix that!

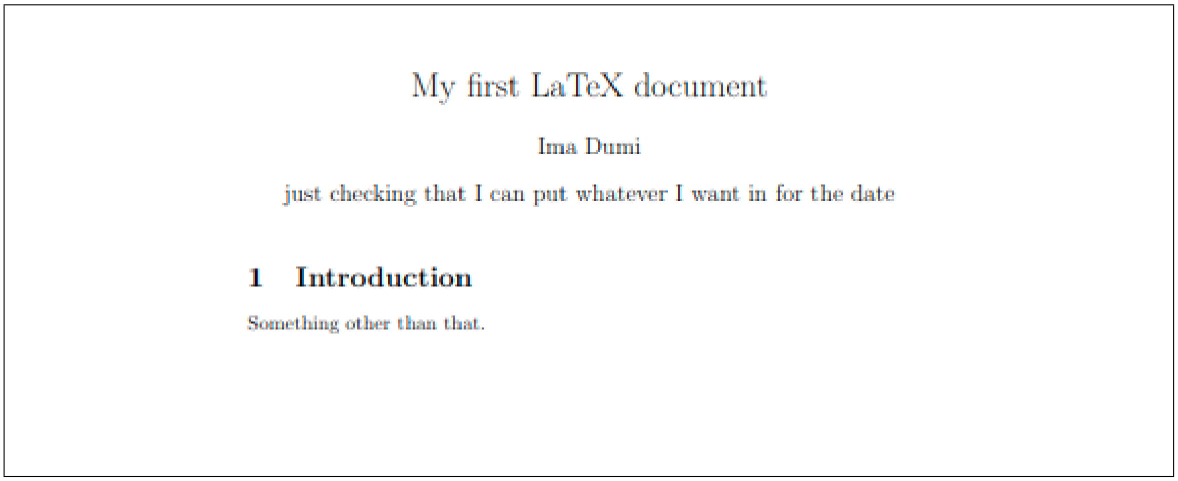
7. Make whatever changes you deem appropriate to the title, author and date commands. (BTW, "date" doesn't necessarily have to literally be the date.) To see what the effect of your changes is, you'll need to press the "Recompile" button.

8. Put some words, introducing yourself after the `\section{Introduction}` command, and recompile. (There is an old and slightly unfunny tradition that the first program you write when learning a new programming language is called "hello world!" Please make your introduction something other than that.)

At this point the source code might look something like:

```
\documentclass{article}
\usepackage{graphicx} % Required for inserting images

\title{My first LaTeX document}
\author{Ima Dumi}
\date{just checking that I can put whatever I want in the date}

\begin{document}

\maketitle

\section{Introduction}

Something other than that.

\end{document}
```

Which should render like so:


My first LaTeX document

Ima Dumi

just checking that I can put whatever I want in for the date

1 Introduction

Something other than that.

Adding a list

Now add a list of your three favorite classes of all time—of course math class is first on your list, so that one has been put in for you. You'll have to supply the next two...

1. Copy and paste the following markup into your document after your greeting, but before `\end{document}`.

```
\par
My three favorite classes of all time are

\begin{enumerate}
  \item Math
\end{enumerate}
```

   Notice that the `\par` tag does not have a begin or end. It only marks where a new paragraph should start. Same with the `\item` tag. It only marks where a new item in the list should start.

2. Add two items to the enumeration (and you can change the first item if by some strange chance math is not your all time favorite class).

3. There are several list-making environments in LaTeX. Try some googling (Maybe "list making latex environments") and you should discover the other list-making environments.

4. Make version of your list of favorite classes that are (1) bulleted rather than numbered, (2) name the class and also give a comment about ~~why math is so awesome~~ why you like it.

Adding an equation

Equations in LaTeX come in two varieties—inline and display. An inline equation is any mathematical expression that appears in the middle of a sentence (like the π right here and earlier in this document). A display equation is any mathematical expression that should appear centered on its own line (because it's super important or just because it's too big to put in the middle of a sentence).

To put an equation in the middle of a sentence, enclose the math between two dollar signs ($). To add a display equation, enclose the math between double dollar signs ($$). Try it!

1. Copy and paste the following markup into your document.

```
My favorite mathematical constant is $\pi$, but I like $e$ too.
Did you know that $$e^{i\pi}=-1?$$ Weird...
```

2. Notice that exponents are typeset using the same notation as used on a calculator! Can you add markup to your document that will produce the following?

   The Pythagorean Theorem states that if a triangle has legs of lengths a and b and hypotenuse of length c, then

   $$a^2 + b^2 = c^2.$$

2.3 MathJax

2.4 More Markup (Blogs and Wikis)

3 Spreadsheets

3.1 Libré Office Calc

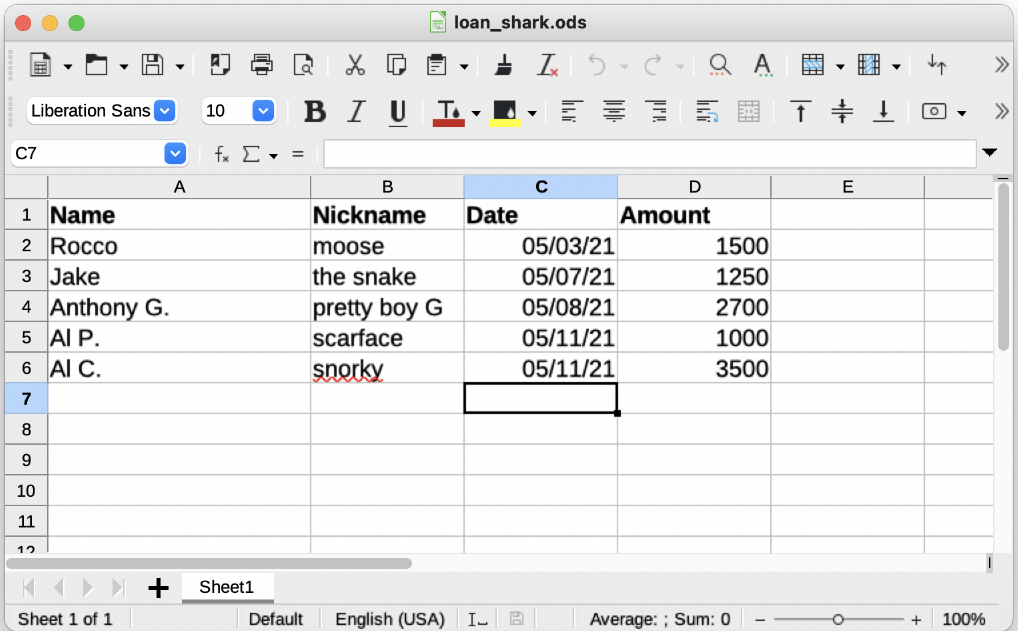
3.1.1 Introduction

tech

Spreadsheets were one of the original “killer apps” when personal computers were first introduced. Spreadsheets allow you to deal with tables of numbers and other data. The standard convention is that the rows of a spreadsheet are indexed by numbers and the columns are indexed with letters. If you need to go past 26 columns, use AA, AB, AC, et cetera for the next several columns. It would be pretty unusual to need to have more than 702 columns but if you needed to, guess what comes after ZZ.

Suppose a businessman was in the completely legitimate business of making loans to people who are regarded as poor credit risks by conventional banks. Of course, he’d want to keep track of the loan amounts and the recipients. A spreadsheet is the perfect tool!

Here’s a screen shot of how the data might be recorded:



	A	B	C	D	E
1	Name	Nickname	Date	Amount	
2	Rocco	moose	05/03/21	1500	
3	Jake	the snake	05/07/21	1250	
4	Anthony G.	pretty boy G	05/08/21	2700	
5	Al P.	scarface	05/11/21	1000	
6	Al C.	snorky	05/11/21	3500	

We're using the free/open-source Libré Office spreadsheet in the above. There are other choices (Excel on Windows and Numbers on Apple computers) but all of them work in a similar way and Libré Office is free. Spreadsheet files created by Libré Office have a .ods suffix.

The stuff that you enter into a cell in a spreadsheet fall into two main categories: a cell can contain data, or a cell can contain a calculated value. To make a "calculation"-type cell you have to put an equals sign up front. When you're doing a calculation you can use the values that are in other cells by referring to them by column (letter) followed by row (number). For example the $3500 that Snorky borrowed is in cell D6.

tasks

Open the loan shark spreadsheet (the file is available on the book website) and make some additions – columns for monthly interest rate, due date and the amount due.

Create a spreadsheet for keeping track of student grades in a math class. Use your favorite actors, sports stars, musicians (or whatever) as students, and just choose random numbers to put in as their grades. Create subtotals for homework, quizzes and exams, also a grand total with homework weighted 20%, quizzes weighted 30% and exams weighted 50%.

3.1.2 Absolute and relative cell references

tech

If you're going to be using the same calculation in a bunch of different places, you can just copy and paste the contents of one cell into another. Get used to using the keyboard shortcuts Ctrl-C and Ctrl-V for copy and paste.

When you copy and paste a formula, the spreadsheet intelligently changes the cell references in the formula. The pasted formula refers to cells that are in the same positions relative to the spot we're pasting into.

For example, if you type `=B3 + C2` into cell `C3` you're telling the system to add the number just above and the number just to the left. If you copy and paste that formula into cell `K7` you'll find that the formula has become `=J7+K6` because those cells are in the same relative positions.

This intelligent pasting usually does the right thing, but occasionally we really just want the thing to stay put! If you want a cell reference to not change when you're cutting and pasting (this is known as an absolute reference) put dollars ($) in front of both the letter and the number. Very occasionally we want a sort of hybrid behavior – we can put the dollar on one but not the other. For instance, if in a formula we refer to a cell using $A3 (with the dollar on the A but not in front of the 3) when we copy and paste the A will stay an A, but the 3 will change appropriately. Some people call this making either the row or the column "sticky."

math

In today's activity we'll be looking at two mathematical concepts: binomial coefficients and difference tables.

Binomial coefficients are sometimes called choice counters. For example, given a set of 5 options how many ways can we select 3 of them? This would be the binomial coefficient $\binom{5}{3}$ which is equal to 10. To pronounce that symbol in English use the word "choose," so the symbol above is read as "five choose three." That notation for binomial coefficients can be a little confusing since many people assume the fraction bar just got left off! So be careful, $\binom{5}{3} = 10$, but $\left(\frac{5}{3}\right) = 1.666\ldots$ so (obviously) these are different – don't imagine fraction bars where they don't actually appear!

There is another notation for the same quantities using a capital letter C. To indicate "five choose three" in this notation write ${}_5C_3$.

So why are these choice counters called "binomial coefficients"? It turns out these numbers also appear when taking powers of a binomial – a polynomial with just two terms. Try computing $(x+1)^0$, $(x+1)^1$, $(x+1)^2$ and $(x+1)^3$. Really, only the last of those is at all difficult! Anything to the 0 power is just 1, anything to the 1 power is itself, and the 2nd power just requires the FOIL rule!

Blaise Pascal – a French mathematician who was one of the founders of the field of probability – was probably the first to notice the pattern when you write these things next to one another.

$$\begin{array}{c} 1 \\ x+1 \\ x^2+2x+1 \\ x^3+3x^2+3x+1 \end{array}$$

The pattern becomes easier to deduce if you remove all the powers of x (and the plus signs) and just concentrate on the coefficients.

$$\begin{array}{ccccccc} & & & 1 & & & \\ & & 1 & & 1 & & \\ & 1 & & 2 & & 1 & \\ 1 & & 3 & & 3 & & 1 \end{array}$$

Numbers on the outside of each row are always 1. Numbers in the middle of a row are just the sum of the two things above them.

The arrangement of binomial coefficients into this triangular array is called Pascal's triangle. Here's the first 5 rows:

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & 1 & & & & \\ & & & 1 & & 2 & & 1 & & & \\ & & 1 & & 3 & & 3 & & 1 & & \\ & 1 & & 4 & & 6 & & 4 & & 1 & \\ 1 & & 5 & & 10 & & 10 & & 5 & & 1 \end{array}$$

Difference tables come from a very common form of analyzing a sequence.

Suppose we asked, "What comes next?" in the following sequence.

$$4, \quad 7, \quad 10, \quad 13, \ldots$$

You probably notice rather quickly that successive terms in the sequence differ by the same number. A difference table is just a formalized way of making the same observation – except that if the differences don't seem to have an obvious pattern we might continue on taking the differences of the differences!

Here's an example. Suppose you're given the following sequence of numbers.

$$2 \quad 5 \quad 12 \quad 23 \quad 38 \quad \ldots$$

A difference table looks like so:

$$\begin{array}{ccccccccc} 2 & & 5 & & 12 & & 23 & & 38 \\ & 3 & & 7 & & 11 & & 15 & \end{array}$$

SInce the differences don't exhibit an obvious pattern we continue on, obtaining

2		5		12		23		38
	3		7		11		15	
		4		4		4		

It looks as though the bottom row – which is known as the second differences is always 4. Can you use that to predict what the next term in the sequence will be?

tasks

Write out the expanded form of $(x+1)^5$.

Create a table of the binomial coefficients $\binom{n}{k} = \frac{n!}{k!(n-k)!}$.

Revisit the grade spreadsheet from the previous section and insert a new row at the top. Put the "weights" for homework, quizzes and exams into cells in this row. (As the teacher you might want to experiment with different weighting schemes.) Are the final grades changed by much if we change the weights to 10, 20 and 70 percent respectively?

Use a spreadsheet to do a finite differences analysis of the following sequence:

$$1 \quad 3 \quad 7 \quad 13 \quad 21 \ldots$$

Find the "back diagonals" for the sequences of squares, cubes, 4th powers etc. (We care about back diagonals because you can use them to generate the entire sequence! (under the assumption that the bottom-most number is a constant) (sorry about all the parentheses)).

Maybe: Given that the back diagonal for x^n contains $s(n,k) * k!$ in the kth row, and the recursion for the Stirling numbers. Use a spreadsheet to create a table of the Stirling numbers.

3.1.3 Builtin functions

Tasks:

Returning to our gradebook example...

Use the MIN function to "drop the lowest quiz."

Use functions to assign letter grades with + and − modifiers.

4 Geometry

4.1 GeoGebra

5 Symbolic Computer Algebra

5.1 Overview

We will investigate the relationship between solutions to equations involving quadratic functions and their graphs by using a computer algebra system. In doing so, we will learn about the following:

1. Making exact computations
2. Simplifying algebraic expressions
3. Solving equations
4. Graphing functions

5.2 Computations using computer algebra

A *computer algebra system* (or *CAS*, for short) is software that is able to perform algebraic procedures similar to what you would normally do "by hand." There are several popular CAS used in mathematics, with the most common being Mathematica, Sage, and Maple. We will be using Sage, as it is available as free open-source software, and it has been largely developed by mathematicians, integrating many specialized packages used by casual and professional mathematics researchers. Much of it uses Python, so if you have any knowledge of coding in Python, you will find Sage to be familiar; but fear not if you do not have any prior experience, as the purpose of this lesson is to get a hang of the basics!

Sage can be installed on your computer or run from the CoCalc servers if you setup an account. For now, we will use a lightweight version of Sage that can be run on a webpage without any installation or setting up an account. Go to https://sagecell.sagemath.org/.

You can enter basic Sage commands into the textbox on the page and click "Evaluate" to get the output. We will begin by testing out some basic arithmetic operations.

First enter the following and click "Evaluate".

```
1+2+3
```

Sage gives the answer of

```
6
```

which is the sum $1 + 2 + 3 = 6$. Note that 6 is interesting in that it is the sum of all the whole numbers smaller than itself that divide into it evenly without leaving a remainder (these are referred to as the *proper divisors* of 6).

We can see that 1, 2, and 3 are proper divisors of 6 without using a calculator, but we can verify this by calculating $6 \div 1$, $6 \div 2$, and $6 \div 3$, respectively. For division in Sage (and indeed in most CAS), we use the "/" sign. Try entering

```
6 / 3
```

in Sage and click "Evaluate". We will receive back the answer of 2, which is of course the value of $6 \div 3$. This is, of course, overkill for something as simple as $6 \div 3$, but it can be useful when working with larger numbers or more complicated expressions. For example, if we want to know whether 9 divides into 6057, we would want to check the value of $6057 \div 9$ and see whether the answer is a whole number. If it is, then 9 divides into 6057 evenly. If it is not a whole number, then that means 9 does not divide into 6057 evenly. Try calculating $6057 \div 9$ in Sage to see what the answer is.

To verify that what Sage returned is in fact the answer to $6057 \div 9$, we can take that number and multiply it by 9 to see if we obtain 6057. To do multiplication, we use the "*" symbol. For example, to calculate 2×3, we would enter

```
2 * 3
```

and click "Evaluate". The result is 6, which is the answer fo 2×3. Take the answer you obtained in Sage for $6057 \div 9$ and multiply that number by 9 to confirm that you indeed obtain 6057.

```
673 * 9
```

Now check whether 9 divides evenly into 70343 by calculating $70343 \div 9$ in Sage. What do you get when you evaluate?

The answer that Sage returns is :

```
70343/9
```

From this, it would be tempting to think that Sage is broken because it did nothing more than return input that you entered as the output. However, remember that a CAS is a tool designed for doing mathematics, and in mathematics, we usually prefer **exact answers** as opposed to decimals, which are not exact for most fractions (e.g. 0.333333333 is *not* the same as $\frac{1}{3}$). However, if we want a decimal approximation for a fraction (or any expression) in Sage, we can put the whole expression inside of "N()". Try evaluating the following in Sage:

```
N(70343/9)
```

Now we obtain an decimal answer that tells us that 9 goes into 70343 times with some remainder. There are several ways to compute the remainder, but the easiest is to use "%". Try inputting and evaluating the following:

```
70343 % 9
```

We obtain an answer of 8, which means that the remainder when dividing 70343 by 9 is 8. We can check that

```
8+9*7815
```

is equal to 70343, which means 9 divides into 70343 a total of 7815 times with an additional remainder of 8. Notice another thing about input and result, which is that unlike a traditional calculator, when we input a particular expression, Sage will follow the correct order of operations. That is exponentials will be calculated first, followed by multiplications and divisions, then additions and subtractions. Expressions that are inside parentheses will precede anything not in parenthesis, with the parts in parentheses following orders of operations as well. If we wanted to do $8 + 9$ first, then multiply the result by 7815, we would need to use parentheses, just like in regular written math

```
(8+9)*7815
```

Finally, the two operations we have not yet discussed are subtraction and exponentiation. Two subtract a number from another number, we use the minus ("–") sign. Try

```
-317-614
```

For exponentiating, we have two options: the caret (“ ˆ ”) or two consecutive asterisks (“**”). The first is common with many CAS and is handy when typing on a computer with a keyboard. The asterisks may be helpful as a shortcut if you are working on a mobile device where special symbols may be hard to access.

We can see that

```
2^3
```

and

```
2**3
```

both provide the answer to 2^3.

5.2.1 Practice Exercises

1. Calculate 2^0, $2^0 + 2^1$, $2^0 + 2^1 + 2^2$, and $2^0 + 2^1 + 2^2 + 2^3$. Continue adding the next largest power of 2 until you notice a pattern in the result. What is the pattern?
2. Use Sage to determine all of the proper divisors of 28. Then verify that the sum of all of the proper divisors of 28 is 28.

5.2.2 Signing up for CoCalc

For more advanced calculations that require multiple steps, it will be beneficial to signup for CoCalc. Follow the following steps to signup and load a Sage notebook for calculations.

1. Go to https://cocalc.com/ in a web browser.
2. At the top right of the screen, click on the “Sign Up” link.
3. Follow the sign up instructions and complete sign up.
4. You should now see a page that says “Signed in as XXXX XXXX” at the top of the page.
5. Click on the “Projects” link at the top left of the page.
6. In the textbox that says, “Project title – you can easily change this at any time!” enter a name for your new project, then click on the “Create New Project” button.

7. On the next page, click the “New” link near the top of the page.

8. Pick a name for your file, select “Sage worksheet,” then click on the “Start project” button at the top of the page.

9. You will now see a blank file with line numbers on the left. By clicking on text area, you will see a blinking cursor where you can enter a calculation you would like for Sage to compute. After you type in your desired command, click the “Run” button at the top of the page or type SHIFT+ENTER to execute that line.

10. CoCalc will run the calculation on the CoCalc server, so give it a few seconds to respond. Once it does, you can continue entering new calculations in the lines below the output. Unlike in an embedded SageMathCell on a webpage, CoCalc will provide output for multiple calculations at once, and it will allow you to keep previous calculations on the webpage so that you can change them or refer back to them.

5.2.3 (optional) Installing Sage

To install Sage on your own computer, go to https://doc.sagemath.org/html/en/installation/binary.html and download the installer appropriate for your operating system. While this requires the most setup as you need to install the software on your computer, calculations will be faster than running on the CoCalc servers.

5.3 Simplifying algebraic expressions

All of the computations we did in the previous section can be done on a regular calculator, but where a CAS like Sage really shines is when working with algebraic expressions including variables. When working with Sage, we will need to declare symbolic variables before we use them.

```
var('x')
```

This tells Sage that we will be using the symbol/letter “x” as a unbound variable in an expression. We wish to treat it as symbol that can represent any possible value and not as a specific value that is fixed.

```
var('x')
3*x+7*x+5
```

This represents the expression $3x + 7x + 5$. Note that the implied multiplication between 3 and x needs to be specified when typing into Sage, as does the implied multiplication between 7 and x. This is because in a CAS, it is also allowable to have names of variables be words or strings (e.g. "apple") and not just single letters. Thus, ab will be interpreted as the variable called "ab" and not the product $a \times b$ of two variables a and b with the multiplication implied by having the two variables next to each other. Similarly, if you just type in

```
var('x')
3x
```

Sage will not interpret this as the product of 3 and x, so it will throw an error message of "invalid syntax" to tell you that this input is not valid Sage input. As an example of a variable having a longer name, consider

```
var('x,speed')
x*speed
```

The first line now declares two variables, one called "x" and another called "speed". Then x*speed represents the product of x with the variable *speed*.

Sage can be helpful when working with algebraic expressions, as it can do things like expand, factor, or simplify expressions.

If we want to factor an expression like $x^2 + 4x + 3$, we can use the **factor** command.

```
var('x')
factor(x^2+4*x+3)
```

We can also expand out an expression using the **expand** command.

```
var('x')
expand( (x+1)*(x+3) )
```

It is also often useful to define a function, such as $f(x)$, much like we do in a regular math class.

```
var('x')
f(x)=x^2+4*x+3
```

We can then evaluate $f(x)$ at different values of x using the standard notation. For example, to evaluate at $x = 1$, we can then enter

```
f(1)
```

which will give us the output of

```
8
```

which is the same as the value of $(1)^2 + 4(1) + 3$, i.e. the value of substituting $x = 1$ into $f(x) = x^2 + 4x + 3$.

We can also easily apply the **expand** and **factor** commands to a function we have already defined.

```
f.factor()
```

will give the output

```
x |--> (x+3)*(x+1)
```

which is the same factorization as when we typed

```
factor(x^2+4*x+3)
```

Similarly,

```
f.expand()
```

can be used to **expand** the function $f(x)$. In addition, a function has the **full_simplify** option.

```
var('x')
f(x)=(x^2+4*x+3)/x+(x+3)*(x-1)/(x+1)
f.full_simplify()
```

Will simplify the very complicated expression of

$$\frac{x^2+4x+3}{x} + \frac{(x+3)(x-1)}{x+1}$$

into a single rational function.

Finally, sometimes Sage will give us an exact expression for something for which we would like a decimal approximation. For example,

```
var('x')
f(x)=(x^2+4*x+3)/x+(x+3)*(x-1)/(x+1)
f(2)
```

gives the output

```
55/6
```

Because Sage is mathematical software, and mathematicians usually want exact answers, that is what it will return, when possible. In order to force it to give a decimal expression, we can use the **n()** command.

```
n(55/6)
```

gives the decimal approximation of

```
9.16666666666667
```

Alternative, we could have entered

```
n(f(2))
```

instead of

```
f(2)
```

which would have given us the same result.

5.4 Solving equations

We will investigate functions of the form $f(x) = ax^2 + bx + c$. To begin with, we will consider when $a = 1$, $b = 0$, and $c = 0$, i.e. $f(x) = x^2$.

Start by defining $f(x) = x^2$ in Sage.

```
var('x')
f(x)=x^2
```

To solve the equation $f(x) = 1$ (i.e. $x^2 = 1$) in Sage, we can use the **solve** command:

```
solve(f(x)==1, x)
```

This will solve the equation $f(x) = 1$, solving for the variable x. Note that when solving, we need to put two equals signs to denote equality. This is because Sage will interpret a single equals sign as an assignment (i.e. we would be defining $f(x)$ to be the function that is always equal to 1).

The output we obtain is

```
[x == -1, x == 1]
```

These are the two solutions that we expect of $x = -1$ and $x = 1$.

5.4.1 Practice Exercises

1. Try to solve the equation $f(x) = 4$ using Sage.
2. Solve the equation $f(x) = 3$ using Sage.
3. Solve the equation $f(x) = 0$ using Sage.
4. Do you expect for there to be any solutions for $f(x) = -9$? Try it in Sage and see what happens.
5. For what values of h does $f(x) = h$ have two real solutions? Only 1 solution? Two imaginary solutions?

5.5 Graphing functions

Another useful way to visualize functions is by plotting the graph of functions. Let's begin by plotting the function $f(x) = x^2$:

```
var('x')
f(x)=x^2
plot(f(x))
```

The first two lines define the function $f(x) = x^2$, then the third line will plot the function. Note that since we did not specify the range for the axes, the plot will automatically pick a range. If we want to see more of the graph, we can try adjusting the **xmin**, **xmax**, **ymin**, and **ymax** values, which correspond to the minimum and maximum values of the x- and y- axes that we desire. For example, changing the plot command to

```
plot(f(x),xmin=-3,xmax=3,ymin=-2,ymax=9)
```

will gives us an x-axis that ranges from -3 to 3 and a y-axis that ranges from -2 to 9.

We can also plot multiple functions on the same plot, in case we want to compare them together. Let's try overlaying the graph of the function $g(x) = 1$ onto the same set of axes:

```
var('x')
f(x)=x^2
g(x)=1
plot([f(x), g(x)],xmin=-3,xmax=3,ymin=-2,ymax=9)
```

We have defined two functions now, $f(x) = x^2$ and $g(x) = 1$. To plot both, we put them both into a list that is enclosed in square brackets, with the two functions separated by a comma. This produces graphs of both $f(x)$ and $g(x)$, in two different colors.

How many times do the graphs of $f(x)$ and $g(x)$ intersect?

5.5.1 Practice Exercises

1. Graph the functions $f(x) = x^2$ and $g(x) = 4$ together on the same set of axes. How many times do the graphs of $f(x)$ and $g(x)$ intersect?

2. Repeat with $f(x) = x^2$ and $g(x) = 3$. How many times do the graphs of $f(x)$ and $g(x)$ intersect?

3. Repeat with $f(x) = x^2$ and $g(x) = 0$. How many times do the graphs of $f(x)$ and $g(x)$ intersect?

4. Repeat with $f(x) = x^2$ and $g(x) = -9$. How many times do the graphs of $f(x)$ and $g(x)$ intersect?

5. What is the relationship between the number of times that the graphs of $f(x) = x^2$ and $g(x) = k$ intersect and the number of solutions to $x^2 = k$?

5.6 Numerical Approximations to Solutions of Equations

Some equations are difficult to solve exactly even with the assistant of a computer and computer software, and Sage is no exception to this.

Use the **solve** function to have Sage solve the equation $x^5 - 3x^4 + x^3 + 2 = 0$.

The output that we get is

```
[0 == x^5 - 3*x^4 + x^3 + 2]
```

which is another way of Sage telling us that it could not find a solution. However, plotting the function $x^5 - 3x^4 + x^3 + 2$ tells us another story. Plot the graph of $x^5 - 3x^4 + x^3 + 2$ in Sage to see whether it has any roots. How many are there, and what are the approximate values from the graph? You may want to play around with the range of the x-axis (using xmin and xmax) to get a clearer picture.

(3 roots, roughly -0.8, 1.2, and 2.5)

We see that there are 3 roots, at roughly $x = -0.8$, 1.2, and 2.5. Although Sage cannot get the exact values for them by solving the equation $x^5 - 3x^4 + x^3 + 2 = 0$, it can get approximate decimal values for the roots by using the **find_root** function. In short, a computer algebra system such as Sage uses a sophisticated version of "find where the graph crosses the x-axis, and zoom in repeatedly around that point to get a more precise estimate of the x-coordinate of where the graph crosses the x-axis." To find a root x where $-2 < x < 0$, we would use the command

```
find_root(x^5-3*x^4+x^3+2==0, -2, 0)
```

The output of this is the root

```
-0.7931397744702121
```

If we want to find a different root, we can change the interval on which we instruct Sage to look for a root. For example, if we want to find the root near 1.2, we could try something like

```
find_root(x^5-3*x^4+x^3+2==0, 1, 2)
```

which gives us the value

```
1.199258801379252
```

Try to find the approximate value of the root of $x^5 - 3x^4 + x^3 + 2$ near $x = 2.5$.

(value is 2.563623765649018)

Note that we said that these values are approximate. Let's verify what we mean by that. Recall that a root of a function $f(x)$ is a value of x that makes $f(x)$ equal to exactly 0, i.e. $f(x) = 0$. Use Sage to define the function $f(x) = x^5 - 3x^4 + x^3 + 2$, then plug in the values of the approximate roots from above, e.g. find $f(-0.7931397744702121)$. If -0.7931397744702121 is truly a solution to $x^5-3x^4+x^3+2 = 0$, then $f(-0.7931397744702121)$ should equal exactly 0.

However, the value that we get from Sage is

```
1.35419453428653e-13
```

The e here is used for engineering notation, where 1.35419453428653e-13 means

$$1.35419453428653 \times 10^{-13}$$

In other words, the part before the "e" is a decimal number, but then the part after the "e" is the exponent that we should raise 10 to then multiply the decimal. Another way to think of this is that the "e-13" means we should start with the 1.35419453428653 then move the decimal to the left 13 times, making our number smaller. In contrast, if we had seen

```
1.35419453428653e5
```

that would mean move the decimal to the right 5 places, resulting in the number 135419.453428653.

The number $1.35419453428653 \times 10^{-13}$ is a really small number, but it's not exactly equal to 0. This demonstrates that the decimal "solution" we obtained using **find_root** is only approximate and not an exact solution.

Another thing to be cautious about when using **find_root** is that it will quit as soon as it finds one approximate root. In other words, even if there is another root in the interval that you specify, **find_root** will only tell you the value of one of them. If we try

```
find_root(x^5-3*x^4+x^3+2==0, 1, 3)
```

in hopes of finding both roots that are between 1 and 3, it will not give us both. Try this out and see what you get!

(only the solution 2.563623765649033 is found)

5.7 Iterating functions and chaos

We now consider what happens when applying the function $g(x) = -rx^2 + rx$ to a number repeatedly.

We will start when $r = 0.5$, so that $g(x) = -0.5x^2 + 0.5x$. We will start with the value $x = 2$. What is the value of $g(2)$? $g(g(2))$? $g(g(g(2)))$? If we continue to apply g more and more times to the result, what eventually happens to the output? Use Sage to make your calculations. You may want to apply **n()** to your output to get a decimal value to more easily see the pattern.

Now change x to any number of your choice and repeat the experiment of applying g repeatedly. What is the eventual outcome?

(all initial values converge to 0)

Now consider when $r = 1.2$, i.e. $g(x) = -1.2x^2 + 1.2x$. Start with an initial value of your choice and repeat the experiment of applying g repeatedly. What is the eventual outcome?

(all initial values converge to $\frac{r-1}{r} = 0.1\bar{6}$)

To track the results of iteratively applying the function $g(x)$ to an initial value of $x = a$, we can create a list of values.

```
var('x')
g(x)=-0.5*x^2+0.5*x
```

```
a=0.5

points = [a, g(a), g(g(a)), g(g(g(a))), g(g(g(g(a)))), g(g(g(g(g(a)))))]
```

We can then plot this using the **list_plots** function.

```
list_plot(points)
```

If we want to change the value of a, we can easily do so, then re-run the commands that come after it. In addition, if we want to see what happens when we apply $g(x)$ more times, we can also add those extra iterates to the list to get a better visualization of what happens to the output as we apply $g(x)$ more and more times.

Play around with different values of r between 2 and 4 and any choice of the initial value x. What patterns do you notice?

Now let's fix a value of $r = 3.5$, so we are looking at the function $g(x) = -3.5x^2+3.5x$. Pick two different values of x that are close to each other (within 0.1 of each other) and plot repeated iterates of $g(x)$ on separate graphs. What do you notice? What happens as the two values of x get closer and closer together?

We note that the values seem to be clustered around 4 different values, and that the iterates seem to cycle through the values systematically. This is called a *periodic orbit*, and in particular, this is a periodic orbit of length 4 because the values (approximately) repeat every 4 times that we apply g. To find the periodic points, we can solve for when $x = g(g(g(g(x))))$. Find the points in this periodic orbit by solving $x = g(g(g(g(x))))$ using Sage.

(Solutions are $x = 0, \frac{3}{7}, \frac{5}{7}, \frac{6}{7}$)