Math and science working groups

Release 2024.01.10

Mark Galassi

Dec 09, 2024

CONTENTS:

Date Dec 09, 2024

Author **Mark Galassi** <mark@galassi.org>

CHAPTER

ONE

INTRODUCTION AND NOTES FOR TEACHERS

state of the book - 2024-02-05

At this time the "mostly-developed" working group is for the more advanced "math for research" in *Part II -- Math for research*: the chapter *Approximating functions with series* and the following chapters, aimed at upper grade high school students. These might also be interesting to students who have finished high school and never got that type of math. They might also be interesting to precocious middle school students who want to push ahead.

The more basic "visualizing algebra" part in *Part I -- Visualizing algebra* is coming together as a course: I have taught the working group twice, and have a clear idea of what is important for the students at that level. But much of what I have concluded, and many of the detailed lesson plans I followed, are not yet here in the book. I should have them in by the next time I teach this working group.

At this time "visualizing algebra" section that is best developed is *Review of prerequisites*. This has proven to be the most important chapter.

I also sometimes have a "physics with calculus" working groups, for which we use the OpenStax University Physics book [Ling, 2016]. Since we just work from that book, there is currently no section in this book for "physics with calculus".

1.1 Overall motivation and plan

1.1.1 Visualizing algebra

The visualization of algebra is fun and mentally stimulating. This correspondence between algebra and geometry should be taught as a delightful romp: fun, amazing, and metaphorically rich.

It is also a key part of being able to attack a data set, understand it, and visualize it. The techniques used for that all make heavy use of this visualization of algebra.

Our plan is then to review how to visualize functions, and to show the details of it which make you fluent in understanding visualization of functions, data, and data analysis techniques.

This book will have very brief chapters and sections, deferring much of the specific work to the OpenSTAX book on Algebra and Trigonometry [Abramson, 2021]

A more complete motivation is given in the section *Motivation and plan* of the "Visualizing algebra" part of this book.

1.1.2 Math for research

Almost all math, including the math used in science and scholarship, does not have exact solutions. The quadratic formula, and the harmonic oscillator, are exceptions.

So in solving math problems we need to develop approximation techniques.

These techniques have been considered the domain of upper division college classes, but an effective computer user can get comfortable with many of these techniques a lot sooner.

The purpose of the math approximation working group is to use effective computer visualizations, and maybe a bit of programming, to learn some of the "numerical approximation" techniques.

We start by getting comfortable with visualization: we make sure that every student can use a plotting program, either on their computer (like gnuplot on linux systems) or GeoGebra or Desmos on the web.

And our practice of plotting gives us a review of polynomials, getting comfortable with how many *extrema* (max or min) and *roots* (zero crossings) a polynomial of n^{th} degree has.

Then our real start consists of learning how to approximate transcendental functions (sin, cos, exp, ...) with power series, and show how that can be applied to simplify the equation of the pendulum.

Then we will move on to Fourier analysis: the techniques to approximate functions with sums of sin and cosine functions.

After this we will look at how to approximate the calculation of areas: the field of numerical integration. We will also mention the whole world of *monte carlo* techniques for calculating integrals.

Then a discussion of how to solve *differential equations* that cannot be solved exactly.

More detailed motivation for, and a rambling introduction to, the "math for research" working group is in that part of the book: the sections *Motivation for "math for research"* and *Rambling introduction*.

1.2 Prerequisites

1.2.1 Visualizing algebra prerequisites

This working group has almost no prerequisites: basic algebra should be enough.

My experience is that many students in American schools take algebra, but then follow paths which allow them to not recall the details. This means that we will do a quick review of basic algebra, customized for the students in the working group.

1.2.2 Math for reseaerch prerequisites

The prerequisites vary according to who is in the working group: with a more advanced group of students we might assume that they know trigonometry, or even differential calculus.

The least we can assume is that the students should know something about polynomials. They might not know the *name* polynomial, but they should understand:

1. Visualizing them (this field is called analytic geometry): plotting a function of x versus x.
2. The equation for a straight line $y = mx + b$
3. The equation for a simple parabola $y = x^2$. From here it is a simple matter to quickly introduce a diverse collection of parabolas, like $y = -2x^2 - 3x + 7$.

If some students have never seen those very simple polynomials then they can join the more fundamental math working group on visualizing algebra: *Part I -- Visualizing algebra*.

A good introduction to polynomials can be found in the openstax textbook on algebra and trigonometry. [Abramson, 2021]. To study from this book you can download the PDF file, or view it in a web browser. The chapter of interest is the one on polynomials and rational functions.

1.3 Notes for teachers

This book is not meant to be a text book for self study. It is meant to give an instructor examples for discussion and elaboration. Some of the text is also so that students can copy+paste instructions into a plotting program or web site.

The way I teach the course is driven by a few things:

- The students are highly motivated.
- They usually hear about the working groups because they have a diverse set of interests, and thus they see mathematics together with its links to other subjects. They also see math fitting in a historical tradition.
- This is not something that they are required to do: they do it because they understand it's important to go beyond school curriculum.

These selection effects mean that the students can work on material more advanced than their grade level.

1.4 A case to whet your appetite

There is a stark example of "being able to solve" versus "not being able to solve" as a problem acquires a bit of complexity: finding the roots of polynomials.

Remember that the *roots* of a polynomial $y = p(x)$ are the values of x for which $y = 0$.

1.4.1 Doing it graphically

Graphing programs

Students should get comfortable with a plotting program which can plot functions. There are online platforms like Geogebra and Desmos which students can use and which we will be using during class.

I also recommend that students become quite comfortable with a command-line plotting program which can plot functions, like gnuplot. I will not require the use of the command line for this working group: we will use the graphical ones in class so that all students can follow.

Let's say that you want to find all the places where the function

$$y = 1.2x^2 - 4x - 12$$

crosses the x axis. This is called finding the *roots* of the function.

Let us start by doing it graphically. Paste the following sequence into your plotting program:

```
1.2*x^2 - 4*x - 12
```

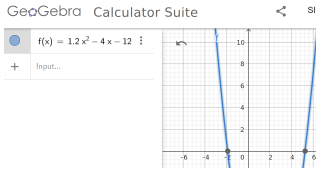

Figure 1.4.1: Overall plot of the polynomial $y = 1.2x^2 - 4x - 12$ - you can spot very rough values of the two roots.

The plot tells you that the parabola has two zero crossings, approximately at "a bit more than -2" and one at "a bit more than 5", as shown in Figure 1.4.1

If you zoom in on the left hand plot as shown in Figure 1.4.2 you can eventually resolve it to 3 digits of accuracy: somwhere between -1.9 and -1.91. You might even hazard the guess of approximately -1.908.

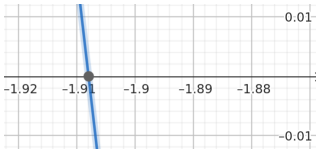

Figure 1.4.2: Same plot, but zooming in on the root to the left you can read off the approximate value of x for which $y = 0$.

If you zoom in on the left hand plot as shown in Figure 1.4.3 you can eventually resolve it to 3 digits of accuracy: somwhere between 5.24 and 5.245. You might even hazard the guess of approximately 5.241.

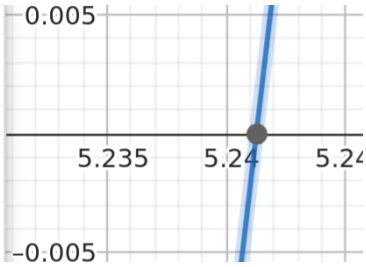

Figure 1.4.3: Same plot, but zooming in on the root to the right you can read off the approximate value of x for which $y = 0$.

1.4.2 Quadratic formula

We have known for a long time how to calculate those solutions *in closed form:* using the general form for the second degree polynomial function $y = ax^2 + bx + c$, the solution sare given by the quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

which in this case is:

$$x = \frac{4 \pm \sqrt{(-4)^2 - 4 \times 1.2 \times (-12)}}{2 \times 1.2} = (5.241268, -1.907935)$$

This is much faster than the graphical approach we used.

So everything is fine, and we can calculate solutions to these problems. In math this is called a closed form solution, and when we find closed form solutions to problems we are very happy.

1.4.3 Let's do a cubic graphically

Let us look at the third degree ("cubic") polynomial:

$$x^3 + x^2 - 10x - 2$$

How do we find its roots? Let us start with the graphical method. Paste this into your online plotting program:

```
x^3 + x^2 - 10*x - 2
```

and you find that there seem to be roots at around $(-3.7, -0.2, 2.8)$ and with a bunch of zooming you could find: $(-3.61, -0.197, 2.81)$.

This gets tiresome…

1.4.4 Cubic formula

So what about a formula for 3rd degree polynomials? Can you find its roots with a formula that might involve square roots or cube roots? The answer is yes: the polynomial function

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

has roots that can be written in closed form – a *cubic formula*.

The actual formula is so long that you will not see it written down in the same way as the quadratic formula – math books always introduce abbreviations to render it. The Wikipedia page on the cubic formula will give you many of the gory details, and this link on quora shows the full gory equation.

I show them here just to point out how they are not really useful:

$$x = -\frac{1}{2}\left(i\sqrt{3}+1\right)\left(\frac{\sqrt{-\frac{1}{3}b^2c^2+\frac{4}{3}ac^3+9a^2d^2+\frac{2}{3}(2b^3-9abc)d}}{6a^2}-\frac{2b^3-9abc+27a^2d}{54a^3}\right)^{\frac{1}{3}}-\frac{b}{3a}-\frac{(b$$

and

$$x = -\frac{1}{2}\left(-i\sqrt{3}+1\right)\left(\frac{\sqrt{-\frac{1}{3}b^2c^2+\frac{4}{3}ac^3+9a^2d^2+\frac{2}{3}(2b^3-9abc)d}}{6a^2}-\frac{2b^3-9abc+27a^2d}{54a^3}\right)^{\frac{1}{3}}-\frac{b}{3a}-\frac{$$

and

$$x = \left(\frac{\sqrt{-\frac{1}{3}b^2c^2+\frac{4}{3}ac^3+9a^2d^2+\frac{2}{3}(2b^3-9abc)d}}{6a^2}-\frac{2b^3-9abc+27a^2d}{54a^3}\right)^{\frac{1}{3}}-\frac{b}{3a}+\frac{b^2-3ac}{9a^2\left(\frac{\sqrt{-\frac{1}{3}b^2c^2+\frac{4}{3}ac^3+9a^2d^2+\frac{2}{3}(2b^3-9a}{6a^2}\right.}$$

Good luck making your browser window wide enough to see the full equations!

1.4.5 Quartic formula

So what about 4th degree polynomials? It turns out that, yes, there is a closed form solution. But this solution is even more complicated than the cubic, so we don't see it written down often. Still, someone has posted it on quora, so here it is in its full majesty:

The 4 solutions to the polynomical equation

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

are given by:

$$x = -\frac{b}{4a} \pm \left(\frac{1}{2} \sqrt{\frac{3b^2 - 8ac}{12a^2} + \frac{1}{3a} \left(\sqrt[3]{\frac{2c^3 - 9bcd + 27b^2e + 27ad^2 - 72ace + \sqrt{\left(2c^3 - 9bcd + 27b^2e + 27ad^2 - 72ace\right)^2 - 4\left(c^2\right.}}{2}} \right.\right.$$

1.4.6 Roots of higher order polynomials

We have learned two things:

1. Closed form solutions exist for 1st, 2nd, 3rd, and 4th degree polynomials. Polynomials and their solutions have a rich history - most of the low-order polynomials were known to ancient Babylonians (20th century BCE), who had tables to help calculate approximate solutions. Exact general algebraic solutions were discoved much later:

   quadratic - 9th century CE General solution by al-Khwarizmi, the founder of Algebra.

   cubic - 16th century CE General solution by Del Ferro and Tartaglia.

   quartic - 16th century CE General solution by Ferrari.

2. They are not very useful beyond 2nd degree.

Now you might ask if closed form solutions exist for a fifth or higher degree polynomial. For a long time people sought out clever tricks, like those used for cubic and quartic, but it turned out to not be possible.

The general formulation of the question was:

> Does there exist a formula for the roots of a fifth (or higher) degree polynomial equation in terms of the coefficients of the polynomial, using only the usual algebraic operations (addition, subtraction, multiplication, division) and application of radicals (square roots, cube roots, etc)?

After couple of centuries in which mathematicians hunted for these closed form solutions, French teenager Èvariste Galois developed a theory (Galois Theory) which allowed him to prove that polynomials of degree 5 or higher have no general closed form solutions.

Solutions obviously still exist for special cases, but there is nothing equivalent to the formulae shown above. They have to be calculated using a *numerical* algorithm.

This proof of the *impossibility* of finding a solution is one of the great achievements of mathematics, and it also means that we need to develop what are called numerical approximations to these problems, since we cannot write down exact expressions.

In Section 19 we will explore numerical techniques to solve this particular problem.

1.4.7 To conclude: how about symbolic algebra?

We will learn the rudiments of using the SymPy symbolic algebra system that runs inside Python. Since we have been looking at the roots of cubic polynomials, let us take a peek at how SymPy would find the roots of the one we looked at above:

$$x^3 + x^2 - 10x - 2$$

To do this we can use the convenient web-based evaluator that the SymPy team offers. Go to the SymPy Gamma web site, and punch this into the text entry field:

```
roots(x**3 + x**2 - 10*x - 2)
```

You should get something like what is in Figure 1.4.4

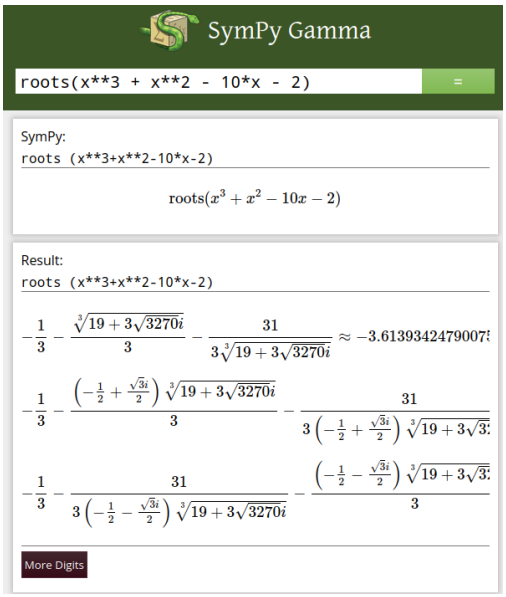

Figure 1.4.4: The roots of our cubic polynomial in closed form.

This is complicated and ugly (although exact); try to change from roots to real_roots:

```
real_roots(x**3 + x**2 - 10*x - 2)
```

and you will get the actual numbers as shown in Figure 1.4.5. Note that you easily get many more digits than with the graphical method, but they are close.

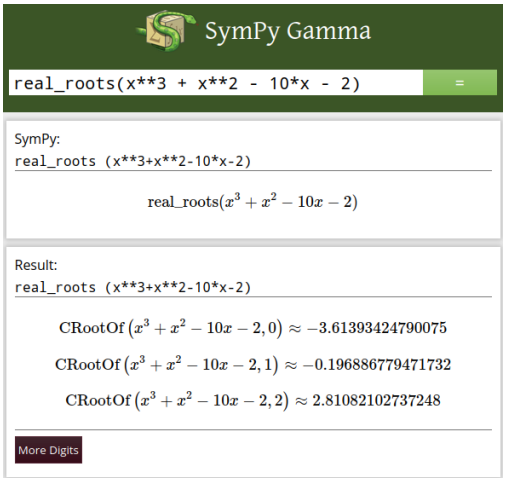

Figure 1.4.5: The roots of our cubic polynomial evaluated numerically. You see that the three roots are all real.

CHAPTER

TWO

LOGISTICS FOR SPECIFIC COURSES

2.1 Fall/winter 2024 - visualizing algebra

First informational meeting:

First meeting:

date: Monday 2024-10-07
time: 7pm US/Mountain (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern)
location: https://meet.jit.si/math-working-group
(note: the first meeting is then repeated on Wednesday Oct. 9th at
8pm and on Sunday October 13th at 10am)

Following meeting times:

New lessons:
Mondays 7:00pm US/Mountain time (6:00pm US/Pacific,
8:00pm US/Central, 9:00pm US/Eastern)

Catch-up lessons: *Wednesday* 8:00pm US/Mountain time and
Sunday 10:00am US/Mountain time (adjust for your time zone)

location: https://meet.jit.si/math-working-group

Initial email to students:

subject-line: math working group starts Oct. 7 - “visualizing algebra”

Dear students,

TL;DR - Join us Monday, Oct. 7, 7pm US/Mountain time (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern) at https://meet.jit.si/math-working-group for the "visualizing algebra" working group kick-off. Let me know if you want to join. The material is described at: https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html

Longer story:

From time to time we have a "math working group" where we explore math differently from how it is done in school, adding elements that are crucial in preparing for advanced academic work and research.

Monday October 7 we will start "visualizing algebra" - the purpose is to build "agility and wide experience in visualizing equations and data". We also do significant review of the algebra you learned a while ago, but we make it fun and memorable.

This is a no-anxiety working group. We get together and do real work, but you are always encouraged to say "hey, I did not get get that at all - can we step back?" You will find that everyone is happy when we slow down or step back! Also: if you ever feel that you are not as good as others, you're wrong. Everyone steps in and out of doing well and being confused, and you will certainly be at both ends, even if you don't notice. That's why it OK to say "please, let's step back".

And most of the work we do will involve visualization to understand the ideas, so at one level you can think of it as looking at pictures of math stuff and discussing the concepts :-)

We will meet on Monday evenings. If you cannot make it that's OK: we will have a catch-up session on most Wednesday evenings, and on the weekend. The catch-up is even more free-form: in catch-up we can discuss any past material.

To see more detail on the material you can look at the "motivation and plan": https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html

Please reply to mark@galassi.org or call +1-505-629-0759 (voice only) if you are interested.

Kick-off meeting and first lesson:

date: Monday 2024-10-07
time: 7pm US/Mountain (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern) - duration: 1 hour and 15min
location: https://meet.jit.si/math-working-group
course name: Visualizing Algebra
link to book: https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html

At our initial meeting we will discuss various logistic matters, and get started doing math!

Expectations of students:

Students need to (a) do this for their own interest, not because of parental pressure; (b) manage their own correspondence entirely; (c) write from a non-school address; (d) Cc: a parent if the student is under 18 years old.

And...

If you cannot make the kick-off meeting, please write or call me and I will confirm the catch-up dates and times.

After this, in the depths of winter, we will kick off the more advanced "math for research" working group.

Initial email to teachers:

subject-line: math working group kick-off Monday Oct. 7 - "visualizing algebra"

Dear faculty,

If you have students who are interested in math or advanced academic work of any kind, we are kicking off this season's math working groups. The first one that is about to start is also very good review for students with lacunae, or who have lingering "math blocks".

The purpose is to give students some extra math tools which are useful when they work on research projects. This is part of the pipeline to prepare students for the Institute for Computing in Research, and internships beyond that.

The first one, "Visualizing algebra", kicks off with an informational meeting and first lesson on Monday, October 7, 2024. This has plenty of review of basic algebra, along with material high school students do not see.

The more advanced one will start later in the winter: approximation techniques (Taylor series, Fourier series, numerical solutions to differential equations).

I hope you can point your students toward this working group. Overall info is at:

https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html

Students, or anyone interested, should email or call me and show up to the informational/kick-off meeting:

date: Monday 2024-10-07
time: 7pm US/Mountain (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern)
videocon-location: https://meet.jit.si/math-working-group
logistics: https://markgalassi.codeberg.page/math-science-working-groups-html/intro-notes.html#fall-winter-2024-visualizing-algebra
email: mark@galassi.org
phone: +1-505-629-0759 (voice only)

2.2 OLD: Winter/spring 2024 - math for research

First meeting:

date: Monday 2024-02-12
time: 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern)
location: https://meet.jit.si/math-working-group

Following meeting times:

New lessons: **Mondays** 8:00pm US/Mountain time (7:00pm US/Pacific,
9:00pm US/Central, 10:00pm US/Eastern).

Catch-up: *Wednesdays* 8:00pm US/Mountain time.
Weekend catch-up: *Sundays* 10am US/Mountain time.

location: https://meet.jit.si/math-working-group

Initial email:

Subject: math working group starts Feb. 12 - "Math for research"

Dear students,

TL;DR - Join us Monday, Feb. 12, 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern) at https://meet.jit.si/math-working-group for the "Math for research" working group kick-off. Let me know if you want to be on the list. The material is described at https://markgalassi.codeberg.page/math-science-working-groups-html/math-for-research/motivation-and-review.html

Longer story:

From time to time we have a "math working group" where we explore areas of math that are crucial in preparing for advanced academic work and research.

Monday Feb. 12 2024 we will start the working group on "math for research". The biggest focus is on the almost-daily techniques used by practicing scientists: approximating functions with Taylor series, Fourier analysis, solving differential equations numerically, and other topics. High schools do not teach these topics - some do a bit, but do not get far.

This is a *no-anxiety* working group. We get together and do serious work on advanced material, but you are always encouraged to say "hey, I did not get get that at all - can we step back?" You will find that everyone is happy when we slow down or step back! Also: if you ever feel that you are not as good as others, you're wrong. Everyone steps in and out of doing well and being confused, and you will certainly be at both ends, even if you don't notice. That's what makes it OK to say "please, let's step back".

And most of the work we do will involve visualization to understand the advanced concepts, so at one level you can think of it as looking at pictures of math stuff and discussing the concepts :-)

We will meet on Monday evenings. If you cannot make it that's OK: we will have a catch-up session on Wednesday evenings, and possibly another one on the weekend. The make-up is even more free-form where we can discuss any past material.

To see more detail on the material you can look at the book we will be using:

https://markgalassi.codeberg.page/math-science-working-groups-html/math-for-research/motivation-and-review.html

and the following chapters, starting with:

https://markgalassi.codeberg.page/math-science-working-groups-html/math-for-research/series.html

---

2.3 OLD: Fall/winter 2023 - visualizing algebra

First informational meeting:

First meeting:

date: Monday 2023-10-30
time: 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern)

location: https://meet.jit.si/math-working-group
(note: the first meeting is the repeated on Wednesday Nov. 2nd at
8pm and on Sunday November 5th at 10am)

Following meeting times:

New lessons:
Mondays 8:00pm US/Mountain time (7:00pm US/Pacific,
9:00pm US/Central, 10:00pm US/Eastern)

Catch-up lessons: *Wednesday* 8:00pm US/Mountain time and
Sunday 10:00am US/Mountain time (adjust for your time zone)

location: https://meet.jit.si/math-working-group

Initial email:

subject-line: math working group info/kick-off Monday October 30, 2023

Dear students and researchers,

In brief - math working groups to prepare for research are coming up: visualizing algebra now (Fall/Winter), and later we will have the advanced working group "math for research" in the winter. Please reply to mark@galassi.org or call +1-505-629-0759 (voice only) if you are interested.

Kick-off meeting:

date: Monday 2023-10-30 time: 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern) location: https://meet.jit.si/math-working-group course name: visualizing algebra link to book: https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html

At our informational meeting we will discuss (a) how often to meet (once or twice/week); (b) if we should meet at 8pm (US/Mountain time zone) or an hour later for our core meeting; (c) what days are best for catch-up sessions.

Then we will start the work!

More detail -

First of all: our format will be very flexible, with the opportunity for missing meetings - we will have make-up sessions, and frequent reviews of previous material.

Motivation:

The math we study in school often does not have the focus and examples that are important for real world research work. These working groups are an approach to solving this problem:

"""What do you do with real world problems, when textbook solutions do not work?"""

This first one, on visualizing algebra, is to build "agility and wide experience in visualizing equations and data." It's also to have fun with visualizations of cool functions.

Specific links:

motivation and plan: https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/motivation-and-review.html#motivation-and-plan

And…

If you cannot make the Monday October 30 kick-off meeting, please send your feedback on what times you can make.

After this, probably in the depths of winter, we will kick off the more advanced "math for research" working group.

2.4 OLD: Winter/spring 2023 - math for research

First meeting:

date: Monday 2023-02-27
time: 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern)
location: https://meet.jit.si/math-working-group

Following meeting times:

New lessons: **Mondays** 8:00pm US/Mountain time (7:00pm US/Pacific,
9:00pm US/Central, 10:00pm US/Eastern).

Catch-up: *Wednesdays* 8:00pm US/Mountain time.
Weekend catch-up: *Sundays* 10am US/Mountain time.

location: https://meet.jit.si/math-working-group

Initial email:

Subject: math working group starts Feb. 27 - "Math for research"

Dear students,

TL;DR - Join us Monday, Feb. 27, 8pm US/Mountain (7pm US/Pacific, 9pm US/Central, 10pm US/Eastern) at https://meet.jit.si/math-working-group for "Math for research" working group. Let me know if you want to be on the list.

Longer story:

Every now and then we have a "math working group" where we explore areas of math that are crucial in preparing for advanced academic work and research.

Monday Feb. 27 2023 we will start the working group on "math for research". The biggest focus is on the almost-daily techniques used by practicing scientists: approximating functions with Taylor series, Fourier analysis, solving

differential equations numerically, and other topics. High schools do not teach these topics - some do a bit, but do not get far.

This is a *no-anxiety* working group. We get together and do serious work on advanced material, but you are always encouraged to say "hey, I did not get get that at all - can we step back?" You will find that everyone is happy when we slow down or step back! Also: if you ever feel that you are not as good as others, you're wrong. Everyone steps in and out of doing well and being confused, and you will certainly be at both ends, even if you don't notice. That's what makes it OK to say "please, let's step back".

And most of the work we do will involve visualization to understand the advanced concepts, so at one level you can think of it as looking at pictures of math stuff and discussing the concepts :-)

We will meet on Monday evenings. If you cannot make it that's OK: we will have a catch-up session on Wednesday evenings, and possibly another one on the weekend. The make-up is even more free-form where we can discuss any past material.

To see more detail on the material you can look at the book we will be using:

https://markgalassi.codeberg.page/math-science-working-groups-html/math-for-research/motivation-and-review.html

and the following chapters, starting with:

https://markgalassi.codeberg.page/math-science-working-groups-html/math-for-research/series.html

2.5 OLD: Winter 2022-2023

2.5.1 (completed) visualizing algebra

This working group has been completed; the information is only here for archival purpose.

First meeting:

```
date: Monday 2022-12-01
time: 7pm US/Mountain (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern)
location: https://meet.jit.si/math-working-group
```

Following meeting times:

```
New lessons: **Mondays** 8:00pm US/Mountain time (7:00pm US/Pacific,
9:00pm US/Central, 10:00pm US/Eastern)

Catch-up lessons: *Wednesdays* 8:00pm US/Mountain time and
*Saturdays* 10am US/Mountain time

location: https://meet.jit.si/math-working-group
```

Initial email:

```
Dear students and researchers,

In brief - math working groups to prepare for research are coming
up: visualizing algebra now, math approximations in
February.  Please reply to mark@galassi.org if you are interested.
```

(continues on next page)

(continued from previous page)

```
Kick-off meeting:

date: Monday 2022-12-05
time: 7pm US/Mountain (6pm US/Pacific, 8pm US/Central, 9pm US/Eastern)
location: https://meet.jit.si/math-working-group
course name: visualizing algebra
link to book: https://markgalassi.codeberg.page/math-science-working-groups-html/

At our informational meeting we will discuss (a) how often to meet
(once or twice/week), (b) if we should meet at 7pm (US/Mountain
time zone) or an hour later.

Then we will start the work.

More detail -

First of all: our format will be very flexible, with the
opportunity for missing meetings - we will have make-up sessions,
and frequent reviews of previous material.

Motivation:

The math we study in school often does not have the focus and
examples that are important for real world research work.  These
working groups are an approach to solving this problem:

"""What do you do with real world problems, when textbook solutions
do not work?"""

This first one, on visualizing algebra, is to build "agility
and wide experience in visualizing equations and data."

Specific links:

motivation and plan:
https://markgalassi.codeberg.page/math-science-working-groups-html/visualizing-algebra/
↪motivation-and-review.html

And...

If you cannot make the Monday December 5 kick-off meeting, please
send your feedback on what times you can make.

After this, probably in February, we will kick off the "math
approximations" working group.
```

2.5.2 Math approximations

Probably starting in February 2023.

CHAPTER

THREE

VISUALIZING ALGEBRA: MOTIVATION AND REVIEW OF PREREQUISITES

3.1 Motivation and plan

Purpose: The goal for this working group on *visualizing algebra* is to build:

> *Agility and wide experience in visualizing and solving equations.*

In-sequence: This fits in the broad sequence of skills that you might call "practical math for research".

Computer use: We will use *plotting programs* and *plotting web sites* to draw functions and data. But:

Programming: Almost no programming. Some explorations can include canned python programs which you can execute without knowing the language.

Examples from: Social science, news and current events, science, sports, ...

Auxiliary textbooks: OpenSTAX textbook on Algebra and Trigonometry [Abramson, 2021].

Other interesting books to accompany this material: *"No Bullshit Guide to Math and Physics"* by Ivan Savov [Savov, 2014].

Who should join? This does not fit in school curriculum boundaries, but if I had to narrow it: the material is aimed at students who are taking the US courses Algebra 1, Geometry, and Algebra 2. It supplements and makes real what students will learn in those courses, and might give them the feeling of excitement that the courses sometimes do not provide.

Users taking more advanced courses, like the US pre-caluculus and calculus, might find very useful review and some new material here (approaches to visualization, symbolic algebra, ...). But if they do not need review then most of it will be old-hat and maybe not the best use of their time.

Those more advanced students might be interested in the other math working group we have: "Math for Research", discussed at this link: *Math for research: motivation and review of prerequisites*.

Are there other things we get out of this? A "working group" is very different from a classroom setting: you can ask conceptual questions, advanced questions, super-basic questions, questions that revisit the things you learned way back, and have a discussion with the instructor and other students. Semi-formal conversations with an experienced mathematician can be a new and useful experience.

Now let us start by reviewing some of the prerequisite materials, before we start with straight lines and then move on to polynomials.

Information on signing up for the next working group should be in the chapter *Logistics for specific courses*.

3.2 Review of prerequisites

In this section we will look at some basic material which we have seen before, but probably not in the context of producing visualizations and other practical results that we want to see.

We plan to make quick work of these topics, but we will still treat them thoughtfully. For each topic we will:

- look at the definition, explanation, and exercises in the textbook
- do some visualization and application to an area of interest

---

Note: This might be the most important chapter in the course: it turns out that there is a lot of diversity in what people have studied so far, and *everyone* appreciates the review. Another important part of the review is that we discuss the *reasons* for formulas like a^0, x^{-3}, and $y^{\frac{1}{2}}$, including putting it in the context of how we had have had to introduce new mathematical structures throughout our life (naturals, integers, rationals, reals, complex, ...) Some students really resonate with that approach.

3.2.1 The "why" of messy exponents

This will be an interactive discussion of why we have expressions like a^0, b^{-3}, $x^{\frac{1}{2}}$.

A handy table for once we have understood them:

$$\begin{array}{rll} a^0 & = 1 & \text{for any a != 0} \\ b^{-3} & = \frac{1}{b^3} & \text{for any b != 0} \\ x^{\frac{1}{2}} & = \sqrt[2]{x} & \text{for any x} \\ x^{\frac{m}{n}} & = \sqrt[n]{x^m} & \text{for any x} \end{array} \tag{3.2.1}$$

In class I give a lengthy explanation of why we introduce these messy exponents:

1. Write down the definition of powers, as we learned them in 6th grader or so, as a *typographical* definition (this is my own use of the word, losely based on Douglas Hofstadter's "Typographical Number Theory"): you simply define 3^4 as "write down the number 3 four times, and put multiplication signs in between":

$$3 \times 3 \times 3 \times 3$$

2. Emphasize with lots of gesticulation that this only defines b^n *if n is a positive integer!!!!*
3. Then play around with those rules of adding exponents and show examples. (The students usually need to be reminded of this.)
4. Then show how you can subtract exponents and get the division of powers of the same base, but emphasize with great emphasis that *this only works if the first exponent is strictly greater than the second one!!!!* otherwise it doesn't mean anything.

$$\frac{b^n}{b^k} = b^{n-k} \quad \text{ONLY IF} \quad n > k$$

5. Now we imagine what would happen if k could be bigger than n, and we ask "if we extend the definition of subtracting exponents to allow a negative power, do we get a consistent arithmetic? Since we do, we decide to use the notation *convention* of

$$b^{-k} = \frac{1}{b^k}$$

We note that it does not follow from the definition - it's a notation convention.

We then do similar reasoning for a^0, $x^{1/2}$ and $x^{m/n}$

3.2.2 A nostalgic romp through coordinates and plotting

Quickly read the Linear Equations in One Variable chapter in [Abramson, 2021].

Exercises: 2, 4, 6, 8, 10 in Review Exercises

3.2.3 What are functions?

https://openstax.org/books/algebra-and-trigonometry-2e/pages/3-introduction-to-functions

It is useful to point out that functions have a very general meaning, even though we will mostly work with functions of real numbers. So start by reading:

Quickly read the Functions and Function Notation chapter in the OpenStax algebra book [Abramson, 2021].

and work the examples and "try it" exercises given in the rubrics from "EXAMPLE 5" all the way to "TRY IT #6".

3.2.4 Special powers of binomials

It is crucial to review what these look like: it is taught in schools, but somehow most students miss it and reach high school not knowing these expresions. You can use our glossary to remember what *monomials* and *binomials* are. Let us look at these expressions:

$$(a+b)^2 = a^2 + 2ab + b^2$$
$$(a-b)^2 = a^2 - 2ab + b^2$$
$$(a+b)(a-b) = a^2 - b^2$$

There are many other such basic identities, but these are the ones I always look at.

We will do two things with this: the first is to actually work them out and see why they are correct. Everyone has their way of doing this - I put my two index fingers down and scan each expression until I have multiplied everything out.

The second thing we will do is to get used to *factoring* polynomials using those identities. To do so it's useful to write them flipping the left and right hand side of the = sign:

$$a^2 + 2ab + b^2 = (a+b)^2$$
$$a^2 - 2ab + b^2 = (a-b)^2$$
$$a^2 - b^2 = (a+b)(a-b)$$

One approach to factoring polynomials is a "visual inspection", where your brain does pattern-matching on things that "look like a perfect square". Here are some examples: we will look at them together and discuss how we found the factors.

$$x^2 - 4 = (x+2)(x-2) \tag{3.2.2}$$
$$x^2 - 3 = (x+\sqrt{3})(x-\sqrt{3}) \tag{3.2.3}$$
$$25x^2 + 20x + 4 = (5x+2)(5x+2) = (5x+2)^2 \tag{3.2.4}$$
$$49x^2 - 14x + 1 = (7x-1)(7x-1) = (7x-1)^2 \tag{3.2.5}$$

The first one is the simple recognition that you have a difference of perfect squares. The second one shows that the technique is useful even if the *number* is not a perfect square.

The third involves recognizing that both the coefficient of x^2 (5^2) and the constant term 4 (2^2) are perfect squares, and the coefficient of x looks like the "double product" $2 \times 5 \times 2 = 20$.

The fourth again involves recognizing that both the coefficient of x^2 (7^2) and the constant term 1 (= 1^2) are perfect squares, and the coefficient of x looks like the "*negative* double product" $2 \times 7 \times (-1) = -14$.

More discussion, examples, and exercises on factoring polynomials is in the chapter Factoring Polynomials in the OpenStax algebra book [Abramson, 2021], which also discusses more elaborate factoring techniques than this simple visual inspection.

I recommend becoming quite comfortable with factoring polynomials: it comes up a lot in the life of a researcher. The "visual inspection" approach should is one you should always know, while the more algorithmic technique shown in the book is one that you should understand and be ready to look up.

Finally: try factoring these same polynomials using SymPy! You can put these expressions into the SymPy Gamma web calculator:

```
factor(25* x**2 + 20 *x + 4)
factor(x**2 - 3)
factor(49*x**2 - 14*x + 1)
```

As a final note on the $(a+b)^2$ expansion, it's good to remember the classic "Pascal triangle":

```
0              1
1             1 1
2            1 2 1
3           1 3 3 1
4          1 4 6 4 1
5        1 5 10 10 5 1
```

which allows us to automatically write down, with no effort:

$$
\begin{aligned}
(a+b)^0 &= 1 \\
(a+b)^1 &= a+b \\
(a+b)^2 &= a^2+2ab+b^2 \\
(a+b)^3 &= a^3+3a^2b+3ab^2+b^3 \\
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
(a+b)^5 &= a^5+5a^4b+10a^3b^2+10a^2b^3+5ab^4+b^5
\end{aligned}
$$

One fun thing to note is that the sum of the exponents of a and b in each term is always the power of the binomial we are expanding.

For example, $(a+b)^4$ has terms like $6a^2b^2$ and $4ab^3$, where 2+2 and 1+3 are both 4.

This relates slightly to ideas in physics: if a and b had physical dimensions (like length or mass) you would get inconsistent expressions if the exponents did not add up to the same amount – you cannot add an area to a volume, for example!

3.2.5 Reviewing fractions – what, seriously??

This is another topic where student preparation is uneven, and it is a good chance to show some tricks.

Simplifying fractions

Start with talking through simplifying these fractions:

- $\frac{36}{3}$
- $\frac{128}{1024}$
- $\frac{512}{384}$

A useful tool to have handy is the "prime factorization". You can simply do a web search for "prime factors 384" and you will get $2^7 \times 3$

If you are running linux you can go faster with the command line program `factor`:

```
$ factor 512
2 2 2 2 2 2 2 2 2
$ factor 384
2 2 2 2 2 2 2 3
```

This tells us that:

$$\frac{512}{384} = \frac{2^9}{2^7 \times 3} = \frac{2^7 \times 2^2}{2^7 \times 3} = \frac{\cancel{2^7} \times 2^2}{\cancel{2^7} \times 3} = \frac{4}{3}$$

These kinds of simplifications are very useful, and they become even more important when we *multiply* fractions.

Important mantra vis-a-vis fractions:

Note: Multiplying and dividing fractions is easy. Adding and subtracting fractions is annoying.

Multiplying (and dividing) fractions

Multiplying fractions involves just multiplying the top and bottom:

$$\frac{a}{b} \times \frac{c}{d} = \frac{a \times c}{b \times d}$$

This is easy! Just what you would dream of. (With plus and minus our dream will be crushed…)

So you can do simple things like:

$$\frac{2}{5} \times \frac{3}{7} = \frac{6}{35}$$

But sometimes the numbers get pretty big, and this is where you want to *simplify before you multiply*. Example:

$$\frac{2}{5} \times \frac{30}{7} = \frac{2}{5} \times \frac{6 \times 5}{7} = \frac{2}{\cancel{5}1} \times \frac{6 \times \cancel{5}1}{7} = \frac{2}{1} \times \frac{6}{7} = \frac{12}{7}$$

Let us do together:

- $\frac{12}{5} \times \frac{20}{27}$
- $\frac{1}{3} \times \frac{18}{19}$
- $\frac{49}{3} \times \frac{12}{21}$

Dividing fractions is just one step from multiplying them. You flip the second (which is called taking the *reciprocal*) and then multiply them:

$$\frac{a}{b} \div \frac{x}{y} = \frac{a}{b} \times \frac{y}{x}$$

Note that I will be trying to move students away from ever using the ÷ symbol, in favor of always using fractions. So you might end up with this kind of situation:

$$\frac{\frac{a}{b}}{\frac{x}{y}} = \frac{a}{b} \times \frac{y}{x}$$

Adding (and subtracting) fractions

Adding (and subtracting) fractions is annoying. The famous "freshman's dream" is incorrect:

$$\cancel{\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}}$$ WRONG/VIETATO/VERBOTEN !!!!

and yet many kids write it down thinking it must be true, because life should occasionally give us something that simple.

Instead you have to do the procedure we learned way back of making a *common denominator*.

It's worth the effort of doing a couple of examples of common denominators with the students, and then quickly moving on.

3.2.6 Getting comfortable with visualizing functions

We have two angles here: one is to become comfortable with what various functions look like. The other is to look at how the intersection (simultaneous solution) of functions in higher dimensions gives insight into the solutions to equations.

The functions most of our students will have seen are linear functions and quadratic functions, so we will visualize a few of these.

In geogebra or desmos enter the following *straight line* functions of x, and discuss the slope and intercepts:

```
(1/2) x + 3
x - 2
-2 x + 4
-x - 1
```

Now insert the following quadratic functions:

```
x^2
x^2 - 4
-x^2
-x^2 - 4
-x^2 - x
-x^2 - x - 1
-x^2 - x + 4
```

and discuss issues like how many roots there are, and whether some of those roots are complex.

More on this topic, which will not be review for most people, will be in Section 5.

3.2.7 Quadratic equations

See the OpenStax chapter on Quadratic equations

This will give us several examples and "TRY IT" exercises that we can do by inspection. For example, we can do example 1, TRY IT 1, example 2, TRY IT 2, example 3, TRY IT 3.

As we do each of these exercises we also plot them in our online graphing calculator.

Motivated students could also go on to the "completing the square" section, but in the working group I would recommend skipping it and moving on.

3.2.8 Some heavy emphasis on how functions are *constraints*

The crucial nexus

There is a crucial point that everyone has to get used to, so we will bring out our graphing calculator and keep talking about it until either everyone falls asleep, or everyone gets it: what does it mean to translate the function to a plot?

We will use the words *satisfy* and *constrain* quite a bit, and we will keep using them until people get comfortable.

The phrases we use to drive home how a function defines the set of (x, y) points will be phrases like these:

"When I write $y = 1.2x - 1$, I am defining the collection of all points (x, y) for which that equal sign *is true*"

"When I write $y = 1.2x - 1$, I am defining the collection of all points (x, y) that *satisfy that relationship*."

"When I write $y = 1.2x - 1$, I am defining the collection of all points (x, y) that *satisfy the constraint* of that equation.

"When I write $y = 1.2x - 1$, I am restricting the possible (x, y) quite dramatically. Before I wrote the function, the entire 2-dimensional plane was 'fair game', but now only a single line is part of that."

"When I write $y = f(x)$, I have reduced the *dimensionality* of the space from the 2-dimensional Euclidean plane to a 1-dimensional curve that lives in that plane."

Let's draw a lot of these with our online graphing calculator, and talk about dimensionality.

Finally, let us talk about what it means to satisfy *two* equations at the same time:

Put these two equations in your graphing program:

$$\begin{cases} y &= x^3 - 1 \\ y &= 2x + 1 \end{cases} \tag{3.2.6}$$

Then subtract a bit from the second one to have something like $y = 2x + 1 - 2$.

Here we talk about how having *two* constraints reduces the dimension of the space from *2* to *0* (a point has dimension 0; a discrete collection of points also has dimension 0).

Here we can gab a bit about a detective drama where the pool of suspects is cut dramatically when you talk about the color of their eyes, or if they are left-handed, ...

We will talk more about the dimensions of spaces and subspaces when we solve more complex equations and systems of equations.

3.2.9 Two equations with two variables

This is probably still "review" for most students, so we take a brief look at *systems of two linear equations with two unknowns*.

We will look at the OpenStax chapter Systems of Linear Equations: Two Variables

Our approach here will be to:

1. Graph some pairs of equations like those in Figure 2 in the OpenStax book and discuss visually when solutions exist and when they do not. We will then make similar graphs in geogebra or desmos for each system we look at.
2. Look at some simple exercises that can be solved either *by inspection* or with a small amount of calculation.
3. Remember the two often-used approaches: *substitution* and *gaussian elimination*.

With this in mind, let us do example 1 through TRY IT #5.

CHAPTER

FOUR

INTRODUCING SYMBOLIC ALGEBRA

4.1 Wouldn't it be nice...

You might have struggled with handling some long gnarly mathematical expressions, and you might have wondered either:

Wouldn't it be nice if they just gave me the numbers, so I can punch them into a calculator, and I don't have to figure out the *general* answer with all the x and y variables...

For example, if my boss asks me "find all the values of x for which this equation is true":

$$\begin{aligned}
&x^5 + x^4\left(-3\pi - \sqrt{7} + 0.4\right) + x^3\left(-12.8 - 1.2\pi - 0.4\sqrt{7} + 3\sqrt{7}\pi\right) \\
&+ x^2\left(-19.2 + 1.2\sqrt{7}\pi 12.8\sqrt{7} + 38.4\pi\right) \\
&+ x\left(-38.4\sqrt{7}\pi + 19.2\sqrt{7} + 57.6\pi\right) - 57.6\sqrt{7}\pi \\
&= 0
\end{aligned}$$

I wish I could just say: "sure, boss, the solutions are $(2.64575131107, -2, 4, -2.4, 9.42477796077)$"

Or wouldn't it be nice if there were a computer program which could do all those algebra or calculus manipulations for me *preserving* the variables without plugging numbers in – all those messy expressions they had me do in algebra class...

For example, if my boss asked me: "find me the derivative of $\sin(7x^3)e^{-x^2}$", I wish I could just say (with no effort):

$$\frac{d\left(\sin(7x^3)e^{-x^2}\right)}{dx} = 21x^2e^{-x^2}\cos\left(7x^3\right) - 2xe^{-x^2}\sin\left(7x^3\right)$$

The first of these wishes is the subject of **numerical analysis**, the second is the subject of **symbolic math**.

In this working group we will be working with a lot of algebraic expressions, and one way in which we will enhance this beyond what is done in school will be to use symbolic math.

We will conclude every segment with an interactive use of the *sympy* symbolic algebra system to have the computer redo our hard-work calculations.

4.2 Preparing your computer to use sympy

4.2.1 On the web

I recommend that at some point you learn to use sympy on your computer with a full python environment. But for the purpose of this course we will use the public web installation called *sympygamma* – https://sympygamma.com/ and enter expressions there.

More complex expressions, but with more prep required, can be used at https://live.sympy.org/ - this will look more like what you do on a computer, showing you the python instructions they use to start.

4.2.2 On a desktop or laptop computer

On a debian-based linux distribution you can run:

```
$ sudo apt install python3-sympy
```

Alternatively, on just about any computer you can run:

```
$ pip3 install sympy
```

You should now be able to type `python3` and then in the python interpreter:

```
>>> import sympy
>>> from sympy import *
>>> init_printing()
>>> x, y, z, t = symbols('x y z t')
>>> k, m, n = symbols('k m n', integer=True)
>>> f, g, h = symbols('f g h', cls=Function)
```

This will allow you to use $x, y, z, t, k, m, n, f, g, h$ as symbolic variables.

4.3 Getting started with a tutorial

We will work with the simpler *sympygamma*, which has a collection of example expressions. They are well chosen, so we will work through them, discussing what it has done after we do each expression.

First we can warm up by putting in simple arithmetic, but we won't do more than one or two expressions, like:

```
22 / 7
355 / 113
```

Then we will expand some expressions:

```
(a + b)**2
expand((a + b)**2)
expand((a + b)**3)
expand((a + b)**4)
expand((a + b)**5)
expand((a + b)**6)
expand((a + b)**7)
expand((a + b)**8)
```

Then let us follow their algebra examples:

```
x
(x+2)/((x+3)(x-4))
simplify((x**2 - 4)/((x+3)(x-2)))
```

If you are curious about how that expression simplified one you can factor $x^2 - 4$ in your head, or you can put into sympy:

```
factor(x**2 - 4)
```

Now some polynomial and rational functions. First put the raw polynomials that will come up into desmos:

```
x^4 / 2 + 5 x^3 / 12 - x^2 / 3
x^2 + 4 x y + 4 y^2 = 0
x^2 + 4 x + 181
x^3 + 4 x + 181
```

(Note that the last two have that big y offset, so you will need to hunt for them in the graph.)

And then the expressions to put in to sympygamma:

```
div(x**2 - 4 + x, x-2)
# my comment for div(): polynomial division with remainders is
# not interesting
gcd(2*x**2 + 6*x, 12*x)
lcm(2*x**2 + 6*x, 12*x)
# my comment for gcd() and lcm(): this mostly goes to show that
# polynomials have a lot in common with integers - I make a brief
# passing mention of "rings" as an algebraic structure.
factor(x**4/2 + 5*x**3/12 - x**2/3)
factor(x**2 + 4*x*y + 4*y**2)
solve(x**2 + 4*x*y + 4*y**2)
solve(x**2 + 4*x*y + 4*y**2, y)
solve(x**2 + 4*x + 181, x)
solve(x**3 + 4*x + 181, x)
solve_poly_system([y**2 - x**3 + 1, y*x], x, y)
```

That last expression corresponds to the system of polynomial equations:

$$\begin{cases} y^2 - x^3 + 1 & = 0 \\ xy & = 0 \end{cases} \tag{4.3.1}$$

The sympy system is well documented. You can start from https://sympy.org/ and follow their documentation link to reach the tutorial page at https://docs.sympy.org/latest/tutorials/index.html#tutorials

The class now moves jumps away from this chapter to their tutorial, starting with the examples at:

https://docs.sympy.org/latest/tutorials/intro-tutorial/intro.html

and continue to their examples of simplification and factoring:

https://docs.sympy.org/latest/tutorials/intro-tutorial/simplification.html

We can stop now, since we will return to sympy examples as we cover those topics.

4.4 [unsorted] Some expressions we will do

1. Simple systems of two linear equations with two unknows, starting with two equations in the form $y = mx + b$, rewriting them as $mx - y + b = 0$, and then putting them into sympy. Have the students pick all the coefficients. For example:

$$\begin{cases} y &= \frac{1}{2}x + 4.2 \\ y &= -3x - \frac{1}{2} \end{cases} \tag{4.4.1}$$

2. System of a linear equation and a quadratic equation. Have the students pick all the coefficients. You could keep one of the lines from before. For example:

$$\begin{cases} y &= \frac{1}{3}x^2 - 2x - 1 \\ y &= \frac{1}{2}x + 4.2 \end{cases} \tag{4.4.2}$$

3. The famous gaussian integral:

$$\int_{-\infty}^{\infty} e^{-x^2} dx$$

```
solve([2*x+a*y-z-1, -7*x+y+2*z+1, x-b*y-z+3], x, y, z)
```

CHAPTER

FIVE

VISUALIZING FUNCTIONS

Here we will discuss visualizing functions, rather than data (which is covered in Section 9). We will discuss what programs can do this, and then take a tour of some examples and what insight we get from those examples.

We will have two goals, beyond getting comfortable with plotting software: (a) learning how functions appear visually, and (b) looking at how the intersection of equations gives a lower dimensional space, and what that space looks like.

5.1 Command-line and web

We can either plot on the command line, or use a web-based plotting system.

On the command line I recommend gnuplot for "quickies": it allows simple plotting of functions. On Linux systems it is already there for ready package installation; on other operating systems one will have to download it, or use a third party packaging approach (such as homebrew) to install it.

On the web there are two widely used system: GeoGebra and Desmos. Both work well for simple line plots, but at this time (2022-12-12) only GeoGebra offers surface plots. GeoGebra is also based on Free/Open-Source software, making it an excellent choice overall.

For the first few examples we will give both the instruction in gnuplot and the GeoGebra instruction, and after a while we will switch to simply showing the gnuplot instructions.

Overall I recommend getting used to command line programs rather than web sites, since such approaches can be scripted and are reproducible.

5.2 Reduction in dimension: line plots

Remember that we are always looking at the reduction in dimension that functions bring, so let us start by plotting some simple functions.

```
$ gnuplot
# then at the gnuplot prompt:
plot x**2
replot -2*x + 3
```

In GeoGebra at https://www.geogebra.org/calculator you can type in the two equations:

```
f(x) = x^2
g(x) = -2 x + 5
```

after which you will have to zoom out a bit and shift the graph down to find all the points at which the line and the parabola intersect.

Exercise

in both approaches, learn to zoom in and out. Then, zooming in, find the (x, y) values of the two points at which the line and parabola intersect. Compare them to what you would get with the quadratic formula.

Finally: notice how we start with the plane (all (x, y) points), which has dimension 2. The first equation reduces the (x, y) pairs to just those on the parabola, which has dimension 1. Finally the second equation brings it down to just two points, each of which has dimension zero.

5.3 Reduction in dimension: surface plots

```
$ gnuplot
# then at the gnuplot prompt:
reset
set zrange [-2:32]
set pm3d
set samples 150
set isosamples 60
set hidden3d
splot 30*exp(-(x**2 + y**2) / 10)
replot x + y
replot 2
```

In GeoGebra at https://www.geogebra.org/calculator you can type in the two equations:

```
z = 30 exp(-(x^2+y^2)/10)
z = x + y
z = 10
```

Then zoom out to see the whole figure.

CHAPTER

SIX

THE PANTHEON OF FUNCTIONS

Here we will take a broad tour of the types of functions that come up in typical applications.

6.1 Broad categories

A list of categories with some simple examples.

polynomials

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

rational functions

$$f(x) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n}{b_0 + b_1 x + b_2 x^2 + \cdots + b_k x^k}$$

algebraic functions

$$f(x) = \sqrt[n]{\frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_i x^i}{b_0 + b_1 x + b_2 x^2 + \cdots + b_j x^j}}$$

elementary transcendental functions

exp/log: e^x, a^x, $\log(x)$, ...

trig: $\sin(\theta)$, $\cos(\theta)$, $\tan(\theta)$, ...

inverse trig: arcsin(x), arccos(x), arctan(x)

hyperbolic: $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, ...

inverse hyperbolic: arcsinh(x), arccosh(x), arctanh(x)

special functions Gamma function $\Gamma(x)$, Bessel functions $J_n(x)$, Dirac delta function $\delta(x)$, error function $\mathrm{erf}(x)$

arithmetic functions (argument is an integer) Prime-counting function $\pi(n)$

6.2 Polynomials

We have already seen many examples of polynomials, so here we only list a couple of plotting examples. First for gnuplot locally on your computer:

```
# for gnuplot
gnuplot
# then at the gnuplot> prompt type:
set grid
set xrange [-3:3]
plot x**3 - 3*x
replot x**3 - 3*x**2 - 4*x + 12
```

The same formulas for a web calculator like GeoGebra or Desmos:

```
x^3 - 3 x
x^3 - 3 x^2 - 4 x + 12
```

6.3 Rational functions

Rational functions are *ratios of polynomials*. Let us start plotting an example:

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

```
gnuplot
# then at the gnuplot> prompt type:
reset
set grid
set samples 1000
set yrange [-100:100]
set xrange [-5:5]
f(x) = (x**2 - 4*x + 2) / (x**3 - 3*x)
plot f(x)
```

And in web graphical calculators:

```
(x^2 - 4 x + 2) / (x^3 - 3 x)
```

Let us now unpack what we have been looking at.

First of all, the function is *not defined everywhere!* The denominator is zero in three places $(-\sqrt{3}, 0, \sqrt{3})$ which means that the overall function is not defined (or, as we often say, it "blows up") at those three points. This is clear in the graph.

The function also has *asymptotic behavior*, the behavior when x goes very far in the positive or negative direction (or, as we often say, when x goes to plus or minus infinity: $x \to \pm\infty$).

Asymptotic behavior is a matter of seeing whether the numerator or denominator *dominates,* which depends on which has the higher power.

Now in class we experiment with rational functions that have higher, lower, or equal powers in the numerator versus the denominator.

One way to do that is to change x^2 (in the numerator polynomial) to x^3 and observe the asymptotic behavior. Then go to x^4 and observe it again.

Another example would be:

$$f(x) = \frac{3x^3 - 5x + 1}{-2x^3 + x + 1}$$

```
gnuplot
# then at the gnuplot> prompt type:
reset
set grid
set samples 1000
set yrange [-100:100]
set xrange [-5:5]
f(x) = (3*x**3 - 5*x + 1) / (-2*x**3 + x + 1)
plot f(x)
```

And in web graphical calculators:

```
(3 x^3 - 5 x + 1) / (-2 x^3 + x + 1)
```

I like to state a couple of take-home messages when I teach this material:

1. The "close-by" behavior, around the origin, has two main features: *roots* (where we cross the x axis) and *singularities* (places where the function blows up). The roots are determined by zeros in the numerator, while the singularities are determined by roots in the denominator.
2. The *asymptotic* behavior, as $x \to \pm\infty$, is determined by ignoring all but the highest power of x, both above and below. Our first fraction $f(x) = \frac{x^2-4x+2}{x^3-3x}$ then becomes:

   $$f(x) \approx \frac{x^2}{x^3}$$

   which converges to zero at both plus and minus infinity. The second one $f(x) = \frac{3x^3-5x+1}{-2x^3+x+1}$ becomes:

   $$f(x) \approx \frac{3x^3}{-2x^3} = -\frac{3}{2}$$

   which converges to $-3/2$ in both directions.

6.4 Algebraic functions

These can have roots in addition to all the rational function components.

We can plot $y = \sqrt{x}$ and discuss its behavior, then moving to $\sqrt{x^3 - 3x}$, just to see what they look like. There is no particular insight here, but it is worth mentioning that $\sqrt{x}$ does grow to infinity, but it slows down a lot compared to x.

An interesting plot to examine next is:

$$y = \sqrt{3 + x^2}$$

After discussing what happens when x gets very big (i.e. you can neglect the 3), you can then comment on how at $x = 0$ we seem to have *curvy* behavior, but then the line seems to straighten out. We can graph the following three separately:

```
sqrt(3 + x^2)
x
-x
```

to see a neat effect.

6.5 Elementary transcendental functions

Start with a discussion of why they are called transcendental, possibly mentioning what the distinction is between algebraic and transcendental numbers.

Then get in to examples and pictures!

6.5.1 Exponentials and logarithms

Discussion of bases.

How fast to they grow? Compare polynomial growth with exponential growth:

```
gnuplot
# then at the gnuplot prompt:
reset
set grid
set xrange [0:4]
plot 2**x
plot x**2
```

in a web graphing calculator:

```
2^x
x^2
```

After exploring how these overtake each other, we add x^7 to the plot and see how long it takes for the exponential to overtake that. We can also throw 10^x in for comparison. Those two polynomials and two exponentials should be enough to develop an intuition for exponential versus polynomial growth.

Then we introduce the three bases: 2, 10, e, and explain why they all exist.

We then invert the exponential to get the logarithm function and we draw it.

One use for logarithms is to compress the axis in a plot. To motivate the use of log in plots:

What if you have something that grows fast. Look at internet sites:

https://web.archive.org/web/20110614121350/http://www.isc.org/solutions/survey/history

https://www.isc.org/survey/

https://en.wikipedia.org/wiki/List_of_sovereign_states_by_number_of_Internet_hosts

and possibly from Our World in Data.

Also look at history of human population.

6.6 trigonometric functions

A detour to talk about *radians* versus *degrees*. This is covered in Section 13.5.1.

We first learn to plot sin, cos, and tan. The others are not really important. We plot them over several periods.

Then we discuss frequency, period, amplitude.

We then conclude with the resemblance that parts of sin and cos have with polynomials, and mention that in the future we will learn about Taylor series.

This discussion of approximation by polynomials allows us to point out that radians are the more natural unit of measure for angles.

6.7 Special functions

6.7.1 Gamma function

$$\Gamma(n) = (n-1)!$$

The full definition of the gamma function for all real (and in fact complex) numbers is more advanced. I show it here, but it is a subject for much later on:

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$$

Now discuss gamma function (and factorial) growth compared to polynomial and exponential. Refer to Section 13.4

6.7.2 Other special functions

Unfortunately it is hard to demonstrate these in the web graphing calculators, so we will leave it out for now.

CHAPTER

SEVEN

TOUR OF POLYNOMIAL GEOMETRY

Here we will take a tour of various polynomials. We will do so interactively, using plotting programs (like gnuplot or geogebra), and occasionally also look at their algebraic properties.

For each polynomial we study we will look at:

special points roots, intercept, vertices

factorization factor the polynomial, using both symbolic and numerical approaches

7.1 Second degree polynomials

$$f(x) = 0.1x^2 - 1.5x + 5$$

Plot it by inputing `0.1*x**2 - 1.5*x + 5` in geogebra.

Then look at making those numbers cleaner by myltiplying it by 10.

Now you can factor it by hand, then use GeoGebra to find:

```
Root(0.1*x^2 - 1.5*x + 5)
```

Now let's ask sympy to do it:

```
from sympy import *
init_printing(use_unicode=True)
x, y, z = symbols('x y z')
p2 = 0.1*x**2 - 1.5*x + 5
print(p2)
solve(p2, x)
Poly(p2, x).all_roots()
```

7.2 Third degree polynomials

Start with this polynomial:

$$f(x) = x^3 - 3x^2 - 4x + 12$$

Input `x**3 - 3*x**2 - 4*x + 12` into geogebra. Read the roots, write it in factored form.

7.3 Fourth degree polynomials

$$p(x) = x^4 - 3x^2 + 1$$

Plot it in geogebra with:

```
x ** 4 - 3 * x ** 2 + 1
```

Then try manipulating it in sympy:

```
from sympy import *
init_printing(use_unicode=True)
x, y, z = symbols('x y z')
p4 = x ** 4 - 3 * x ** 2 + 1
p4
factor(p4)
poly4 = Poly(p4, x)
poly4.all_roots()
p4 = x ** 4 - 3 * x ** 2 + 1.7
factor(p4)
```

7.4 Fifth degree polynomials

Let us craft a 5th degree polynomial. In sympy let us write:

```
from sympy import *
init_printing(use_unicode=True)
x, y, z = symbols('x y z')

expr5 = (x - 1) * (x - 2) * (x + 1) * (x + 2) * (x - 7)
expr5
p5 = expand(expr5)
p5
print(p5)
```

CHAPTER

EIGHT

SOLVING DIFFICULT EQUATIONS

The Youtube channel SyberMath has several interesting equations, including rational, irrational, and transcendental. Here we will show some of them so that we can solve them together.

Repeated viewing of technique used should get us to the point where we know how to tackle them.

The channel is at https://www.youtube.com/@SyberMath

8.1 Polynomial equations

$$x^6 = (x-1)^6$$
$$(a+b)^5 = a^5 + b^5$$

8.2 Rational equations

Simplify:

$$\frac{2\sqrt{6}}{\sqrt{2}+\sqrt{3}+\sqrt{5}} + \sqrt{5}$$

8.3 Irrational equations

$$\sqrt{x+8} - \sqrt{x} = 2$$

8.4 Transcendental equations

$$(\ln x)^5 = \ln x^5$$
$$2^{x^2} = 3^x$$
$$(x^2 - x - 1)^{x+2} = 1$$

8.5 Functional equations

$$f\left(\frac{2x-1}{x-3}\right) = x^2$$

$$x = \frac{3y-1}{y-2}$$

CHAPTER

NINE

DATA VISUALIZATION

We have focused mostly on visualizing *functions*, but it is important to also know how to visualize *data*.

Since it is not a part of this algebra course I will simply give a pointer to where you can find a lesson I wrote on it for the "Research Skills and Critical Thinking" book:

https://markgalassi.codeberg.page/research-skills-html/data-visualization.html

CHAPTER

TEN

FITTING

10.1 Fitting a straight line

$$mx_1 + b = y_1$$
$$mx_2 + b = y_2$$

which is equivalent to:

$$mx_1 + b - y_1 = 0$$
$$mx_2 + b - y_2 = 0$$

```
from sympy import *
init_printing(use_unicode=True)
m, b = symbols('m b')
x1, x2, y1, y2 = symbols('x1 x2 y1 y2')
solve([m*x1 + b - y1, m*x2 + b - y2], m, b)
# HIT ENTER HERE, then put in the rest
```

```
# long result
x1 = 1
y1 = 1
x2 = 2
y2 = 2
solve([m*x1 + b - y1, m*x2 + b - y2], m, b)
result: {a: 2, b: -5, c: 4}
```

10.2 Fitting a parabola

$$ax_1{}^2 + bx_1 + c = y_1$$
$$ax_2{}^2 + bx_2 + c = y_2$$
$$ax_3{}^2 + bx_3 + c = y_3$$

which is equivalent to:

$$ax_1{}^2 + bx_1 + c - y_1 = 0$$
$$ax_2{}^2 + bx_2 + c - y_2 = 0$$
$$ax_3{}^2 + bx_3 + c - y_3 = 0$$

```
from sympy import *
init_printing(use_unicode=True)
a, b, c = symbols('a b c')
x1, x2, x3, y1, y2, y3 = symbols('x1 x2 x3 y1 y2 y3')
solve([a*x1**2 + b*x1 + c - y1, a*x2**2 + b*x2 + c - y2, a*x3**2 + b*x3 + c - y3], a, b,␣
↪c)
# HIT ENTER HERE
```

```
# long result
x1 = 1
y1 = 1
x2 = 2
y2 = 2
x3 = 3
y3 = 7
solve([a*x1**2 + b*x1 + c - y1, a*x2**2 + b*x2 + c - y2, a*x3**2 + b*x3 + c - y3], a, b,␣
↪c)
result: {a: 2, b: -5, c: 4}
```

10.3 Fitting a cubic

$$
\begin{aligned}
{x_1}^3 + {x_1}^2 + x_1 + d &= y_1 \\
{x_2}^3 + {x_2}^2 + x_2 + d &= y_2 \\
{x_3}^3 + {x_3}^2 + x_3 + d &= y_3 \\
{x_4}^3 + {x_4}^2 + x_4 + d &= y_4
\end{aligned}
$$

which is equivalent to:

$$
\begin{aligned}
{x_1}^3 + {x_1}^2 + x_1 + d - y_1 &= 0 \\
{x_2}^3 + {x_2}^2 + x_2 + d - y_2 &= 0 \\
{x_3}^3 + {x_3}^2 + x_3 + d - y_3 &= 0 \\
{x_4}^3 + {x_4}^2 + x_4 + d - y_4 &= 0
\end{aligned}
$$

```
from sympy import *
init_printing(use_unicode=True)
a, b, c, d = symbols('a b c d')
x1, x2, x3, x4, y1, y2, y3, y4 = symbols('x1 x2 x3 x4 y1 y2 y3 y4')
solve([a*x1**3 + b*x1**2 + c*x1 + d - y1,
       a*x2**3 + b*x2**2 + c*x2 + d - y2,
       a*x3**3 + b*x3**2 + c*x3 + d - y3,
       a*x4**3 + b*x4**2 + c*x4 + d - y4],
       a, b, c, d)
# long result
x1 = -4
y1 = -3
x2 = -2
y2 = 0
x3 = 2
```

(continues on next page)

(continued from previous page)

```
y3 = 0
x4 = 4
y4 = 2
solve([a*x1**3 + b*x1**2 + c*x1 + d - y1,
       a*x2**3 + b*x2**2 + c*x2 + d - y2,
       a*x3**3 + b*x3**2 + c*x3 + d - y3,
       a*x4**3 + b*x4**2 + c*x4 + d - y4],
       a, b, c, d)
```

CHAPTER

ELEVEN

APPENDIX: VISUALIZING ALGEBRA - SAMPLE LESSON PLANS

Teaching the "visualizing algebra" material can clearly be sliced up in many different ways. I have taught it several times, and I always followed a similar logistical format: class length of 1 hour and 15 minutes, and new material once/week, with catch-up sessions twice/week.

But the order in which I teach the various pieces, and the choice of which pieces to teach, depends largely on the makeup of the group of students I work with.

Similarly, the amount of extra background material and the choice of "visions into advanced material" anecdotes depend on the group of students I have.

In this appendix I will start collecting some of the specific lesson plan sequences I have chosen.

11.1 Fall 2024

Opening lesson 1:

- Introduce myself, discuss how researchers see mathematics
- Discussion of logistics and what I expect of students
- Section "The why of messy exponents"
- Section "An example to whet your appetite"

Lesson 2:

- Review from OpenSTAX book: "A nostalgic romp through coordinates and plotting" - do the exercises suggested in the section.
- Review from OpenStax book: "What are functions?" - do the exercises suggested in the section.
- Section "Special powers of binomials"
- If we have time, start with sympy and introduce `expand((a + b) ^ 7)` and so forth.

Lesson 3:

- Start with sympy, using the expressions in Section 4

Lesson 4:

Return to the "review of prerequisites" with the following sections:

- Reviewing fractions - what, seriously??
- Getting comfortable with visualizing functions
- Quadratic equations
- Some heavy emphasis on how functions are *constraints*

Remember to really lay it thick on how each equation reduces the dimensionality of the space.

Lesson 5:

We are now well beyond the review section, and we can spend a brief amount of time on:

- Visualizing functions, using Section 5 – the purpose here is to get used to 2D and 3D plots in Desmos or whatever other online graphing calculator we use.

Then we spend most of our time on:

- The pantheon of functions, using Section 6

CHAPTER

TWELVE

MATH FOR RESEARCH: MOTIVATION AND REVIEW OF PREREQUISITES

12.1 Motivation for "math for research"

Approximating functions and data with sums of carefully chosen functions (like polynomials or trigonometric functions) is an almost daily task for working scientists. And yet I know of no curriculum that puts these topics together before students have reached an advanced calculus level.

Our goal here is to explore series approximations to various functions, and in the process to look at other places in which series come up.

The most important take-home is probably that polynomials can be used to to approximate reasonably well-behaved functions *locally*, and Fourier series can be used to approximate reasonably well-behaved periodic functions *globally*.

In doing so we will also get a tour of many interesting mathematical functions, as well as physical problems that can be solved thanks to these kinds of approximations.

We will not assume a knowledge of calculus at the start, but we will occasionally step out to learn some speficic topics in differentition and integration when we need it.

We will start with some mathematical *experiments* in which we start from known series and see how well they work. This will give us a dissatisfaction which can be expressed as "sure, I see that it works, but how do you get there?"

Then we will move on to learning how to generate the kinds of series we have been experimenting with. This will allow us to work such approximations for many different types of situations.

12.2 Rambling introduction

First of all: this is a no-anxiety working group. We get together and do serious work, but you are always encouraged to say "hey, I did not get get that at all - can we step back?" You will find that everyone is happy when we slow down or step back! Also: if you ever feel that you are not as good as others, you're wrong. Everyone steps in and out of doing well and being confused, and you will certainly be at both ends, even if you don't notice. That's what makes it OK to say "please, let's step back".

As we study how to approximate things with polynomials (and other series), we find that the traditional school model does not completely work. In algebra and trigonometry they have text books which have codified the sequence in which you are supposed to study it, and they have tons of exercises. The books at https://openstax.org/subjects/math for example have tons of very good exercises.

But real work bridges these codified separations, and the textbooks are not as easy and mature. Sometimes one has to patch together many different sources of information, and different sources make different assumptions on what you already know.

This means that you are taken away from that comfort zone of everything being mapped out. In exchange for that discomfort you get to *bridge* areas, and your own thoughts can get profound.

And as we leave the comfort of the schoolkid approach, we replace that with the comfort that in our working group there is no shame in stepping back and re-tackling things from the same or different angles.

One challenge for us will be to find canned exercises for us to work from. Those are helpful, but harder to find as we progress. Our exercises might sometimes involve figuring out how to plot something, or how to write a couple of lines of code about something. We will move slowly on that.

Finally: the progression I have in mind is something like this:

- sequences
- sums of numbers
- taylor polynomials
- taylor series
- fourier series
- numerical solutions to differential equations

but we will happily meander and step back to basic math when we need more material from polymials, or trig, or exponentials, or even an occasional flash-forward to a calculus idea.

12.3 Review of prerequisites

The main prerequisite is what we covered in the "Visualizing algebra" working group.

We will introduce other ideas (such as derivatives and integrals) as the need comes up.

Here are some videos and other preparation tips which give you an interesting review, or just some attractive demonstrations.

- "3 blue 1 brown" video on Taylor series. Note that we are not going to do it the way they do: they assume you know calculus, and we don't. But it is visually quite lovely, so if you have a few minutes take a look.

https://www.youtube.com/watch?v=3d6DsjIBzJ4&t=155s

- Watch the "map of mathematics" video on youtube: https://www.youtube.com/watch?v=OmJ-4B-mS-Y revel in the British accent, and see how all areas of math fits together.
- Watch the "Professor Dave" video on sequences and sums: https://www.youtube.com/watch?v=-DPkqpmm1sI&list=PLybg94GvOJ9FoGQeUMFZ4SWZsr30jlUYK&index=84
- Have ready a way of calculating and plotting. A calculator is OK, as is a computer calculator program. For calculating I personally use "M-x calc", a reverse polish notation (RPN) calculator built in to emacs. If you find RPN too nerdy then the fastest (i.e. no mouse) is probably to start a python3 interpreter and type expressions in to it, but anything you're comfortable with… For plotting I recommend having gnuplot handy because it will allow us to manipulate the expressions we plot.

CHAPTER

THIRTEEN

APPROXIMATING FUNCTIONS WITH SERIES

As I mentioned in the "motivation" section Section 12, apprimationg by series is a technique used almost daily by scientists. Let us dive in.

13.1 Sequences and sums

The classic definition of sequences and sums should be familiar.

For our purposes a sequence is an ordered list of numbers. We can index it with an integer. We can describe these sequences any way we want, for example in English we could say *the sequence of all positive integers* or *the sequence of all positive odd numbers* or even numbers and so forth.

Or we can describe them with examples of numbers where you can spot the pattern, or with math operations that map from integers to the numbers in our sequence. We often use curly braces to show that an index like i or k should be understood to cover all integers (or non-negative or positive integers). For example:

$$\text{Natural numbers} : \{i\} : 1, 2, 3, 4, 5, \ldots$$
$$\text{Even numbers} : \{2i\} : 2, 4, 6, 8, 10, \ldots$$
$$\text{Odd numbers} : \{2i - 1\} : 1, 3, 5, 7, 9, \ldots$$

The *sum* of a sequence is simply the addition of all numbers in that sequence. We use the classice uppercase Σ notation. For example:

$$\sum_{k=1}^{100} k = 1 + 2 + 3 + 4 + \cdots + 100$$

In this notation the famous Gauss summation formula is:

$$\sum_{k=1}^{N} k = 1 + 2 + 3 + 4 + \cdots + N = \frac{N(N+1)}{2}$$

An example of a function defined by a sum is the Riemann zeta function, defined as:

$$\zeta(s) = \sum_{k=1}^{\infty} \frac{1}{k^s}$$

We are not really going to study the Riemann zeta function here, but we are interested in one value, $\zeta(2)$:

$$\zeta(2) = \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$$

This will give us an interesting playground to experiment with calculating values of π.

Notice that a sum can be a *finite* sum or it can be a *series:* a sum of infinite terms where they get smaller and smaller so that the sum converges.

13.2 Do sums converge?

Convergence of series is a beautiful mathematical topic, but we will stick to our practical mode of operation here and mostly look at some examples to develop an intuition about it.

For example, the harmonic series diverges:

$$\sum_{k=1}^{\infty} \frac{1}{k}$$

which is interesting: it means that even though each number $1/k$ is getting smaller and smaller, eventually tending to zero, the sum keeps growing without bounds.

Exercise: write a brief Python program which shows that if you pick any number, you can pick a number of terms of the harmonic series that gets bigger than that. Try it for a thousand, a million, then a billion, and see how many terms of the harmonic series you need to get bigger than that.

On the other hand, the *alternating* harmonic series:

$$\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} \cdots = \log(2)$$

converges to the natural logarithm of 2. A variant of the alternating harmonic series with just the odd terms is called the Leibniz series:

$$\sum_{k=0}^{\infty} \frac{(-1)^{k}}{2k+1} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \cdots = \frac{\pi}{4}$$

13.3 Approximating pi with series

Calculate an approximation to π using the Leibniz series:

```
import math
sum = 0
N = 16
for i in range(0, N+1):
    term = (-1.0)**i / (2*i+1)
    sum += term
    pi_approx = 4.0 * sum
    print(i, '   ', term, '   ', sum, '   ', pi_approx)
```

And with the Riemann zeta function:

```
import math
sum = 0
N = 16
for i in range(1, N+1):
    term = 1.0 / (i*i)
    sum += term
    pi_approx = math.sqrt(6.0 * sum)
    print(i, '   ', term, '   ', sum, '   ', pi_approx)
```

13.4 A digression on the factorial

$$
\begin{aligned}
0! &= 1 \\
1! &= 1 \\
2! &= 2 \times 1 \\
3! &= 3 \times 2 \times 1 = 3 \times 2! \\
4! &= 4 \times 3 \times 2 \times 1 = 4 \times 3! \\
5! &= 5 \times 4 \times 3 \times 2 \times 1 = 5 \times 4! \\
&\dots \\
n! &= n \times (n-1)!
\end{aligned}
$$

There is a function for real numbers called the gamma function which has the same value as the factorial (of one less) when it hits the nonnegative integers:

$$\Gamma(n) = (n-1)!$$

The full definition of the gamma function for all real (and in fact complex) numbers is more advanced. I show it here, but it is a subject for much later on:

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx$$

How fast does the factorial grow? Linear? Polynomial? Exponential? Something else?

We can study that question a bit with the following:

specific values First in your head, then using a calculator, calculate 0!, 1!, 2!, 3!, …, 10!

the function Then use the following plot commands to see the gamma function:

```
$ gnuplot
# then at the gnuplot> prompt:
set grid
set xrange [0:2]
plot gamma(1+x)    # this is equivalent to x! for integers
replot x**2
replot x**5
replot 2**x
replot exp(x)
# pause and examine
set xrange [0:3]
replot
# pause and examine
set xrange [0:4]
replot
# pause and examine
set xrange [0:5]
replot
# pause and examine
set xrange [0:6]
replot
# pause and examine
set xrange [0:7]
replot
```

(continues on next page)

(continued from previous page)

```
# pause and examine
set xrange [0:8]
replot
# pause and examine
set xrange [0:9]
replot
```

Once you have tried to pit $n!$ (or $\Gamma(n+1)$) against various exponential functions, take a look at *Stirling's approximation* to the factorial:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

which is just the first term of a more elaborate series:

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \frac{1}{12n} + \frac{1}{288n^2} - \frac{139}{51840n^3} - \frac{571}{2488320n^4} + \dots\right)$$

Stirling's formula is often used to calculate logarithms of $n!$ like this:

$$\log n! = n\log n - n + O(\log n)$$

Looking closely at these formulae shows us that $n!$ is indeed "super-exponential" in how it grows for large values of n.

13.5 Experiments with series for sin and cos

We will start by doing experiments with the polynomials that give us approximations to $\sin(x)$ and $\cos(x)$.

13.5.1 Getting comfortable with radians

Remember: if a sensible angle feels like it's in the range of 30 or 45 of 88 then it's probably in degrees. If it's expressed as $\pi/4$, or some other multiple or fraction of π, or a number that's between 0 and 7, then there's a good chance it's measured in radians.

The conversion factor is $\frac{\pi}{180}$ or its inverse $\frac{180}{\pi}$:

$$\text{angle\_radians} = \text{angle\_degrees} \times \tfrac{\pi}{180} \tag{13.5.1}$$

$$\text{angle\_degrees} = \text{angle\_radians} \times \tfrac{180}{\pi} \tag{13.5.2}$$

You should get comfortable with the everyday angles we use and what they are in radians: 90deg = pi/2, 60deg = pi/3, 45deg = pi/4, 30deg = pi/6.

A chart is at:

https://en.wikipedia.org/wiki/File:Degree-Radian_Conversion.svg

13.5.2 Making plots of polynomials that approximate sin and cos

Let us first generate plots that approximate sin and cos for very small angles:

$$\sin(x) \approx x$$
$$\cos(x) \approx 1$$

Use your favorite plotting program to show the following. I give the examples for gnuplot and for desmos and geogebra:

```
$ gnuplot
# then at the gnuplot> prompt:
set grid
plot [-1.5*pi:1.5*pi] [-1.5:1.5] sin(x)
replot x
replot cos(x)
replot 1
```

```
sin(x)
x
cos(x)
y = 1
```

Study this and make a guess as to where x is a good approximation to sin(x), and where 1 is a good approximation to cos(x). Set your calculator to radians and calculate how far off the approximation is for those values of x you come up with.

The next terms in the sin and cos series are:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} \cdots$$
$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} \cdots$$

Continue approximating the plots for sin and cos with higher degree polynomials, for example with the instructions below, and every time make an estimate (and then calculate it) for where they start diverging.

```
$ gnuplot
## then the following lines have the prompt gnuplot> and we type:
set grid
plot [-6:6] [-1.5:1.5] sin(x) lw 3
replot x
replot x - x**3 / 3!
replot x - x**3 / 3! + x**5 / 5!
replot x - x**3 / 3! + x**5 / 5! - x**7 / 7!
replot x - x**3 / 3! + x**5 / 5! - x**7 / 7! + x**9 / 9!
```

```
sin x
x
x - x^3 / 3!
x - x^3 / 3! + x^5 / 5!
x - x^3 / 3! + x^5 / 5! - x^7 / 7!
x - x^3 / 3! + x^5 / 5! - x^7 / 7! + x^9 / 9!
```

```
$ gnuplot
## then the following lines have the prompt gnuplot> and we type:
set grid
plot [-6:6] [-1.5:1.5] cos(x) lw 3
replot 1
replot 1 - x**2 / 2!
replot 1 - x**2 / 2! + x**4 / 4!
replot 1 - x**2 / 2! + x**4 / 4! - x**6 / 6!
replot 1 - x**2 / 2! + x**4 / 4! - x**6 / 6! + x**8 / 8!
replot 1 - x**2 / 2! + x**4 / 4! - x**6 / 6! + x**8 / 8! - x**10 / 10!
```

```
(cos x)
1
1 - x^2 / 2!
1 - x^2 / 2! + x^4 / 4!
1 - x^2 / 2! + x^4 / 4! - x^6 / 6!
1 - x^2 / 2! + x^4 / 4! - x^6 / 6! + x^8 / 8!
1 - x^2 / 2! + x^4 / 4! - x^6 / 6! + x^8 / 8! - x^10 / 10!
```

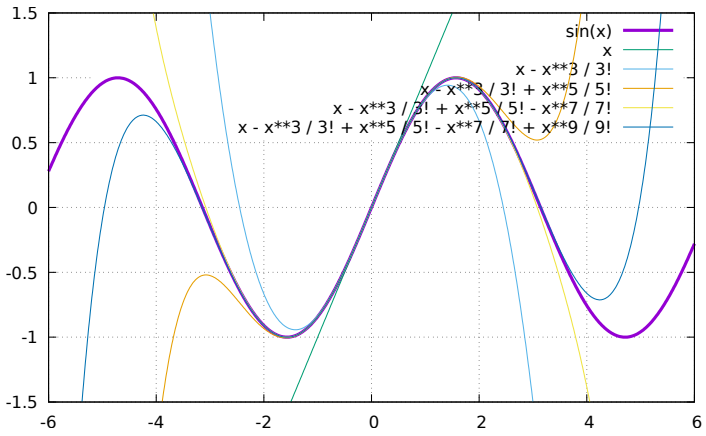

Figure 13.5.1: The first few polynomial approximations for the sin functions. The thicker line is the sin function, and the thinner ones are the ever-improving polynomial approximations.

What is the take-home from these two figures? What they show us is that you can approximate $sin(x) \approx x$ for small values of x. You can also approximate $cos(x) \approx 1 - x^2/2!$ for small values of x.

13.6 Starting to study exponentials

13.6.1 Plotting some exponentials

We first get used to exponentials by looking at 10^x and comparing it to polynomial growth. In gnuplot:

```
$ gnuplot
# then at the gnuplot prompt:
reset
set grid
```

(continues on next page)

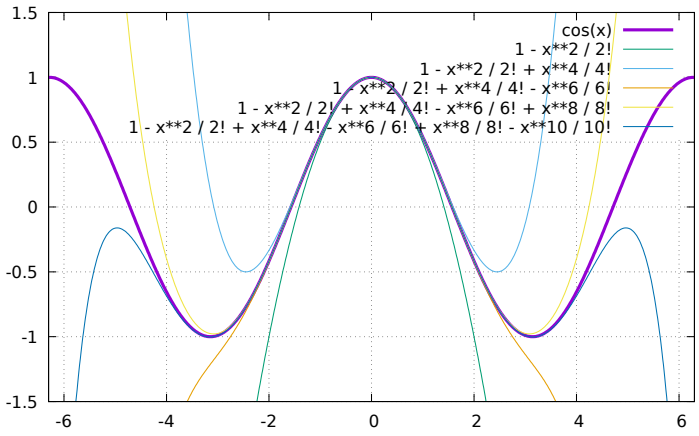

Figure 13.5.2: The first few polynomial approximations for the cos functions. The thicker line is the cos function, and the thinner ones are the ever-improving polynomial approximations.

(continued from previous page)

```
set xrange [0:4]
plot 10**x
replot x
replot x**2
replot x**3
replot x**4
replot x**5
replot x**6
replot x**7
```

And in desmos/geogebra:

```
10^x
x
x^2
x^3
x^4
x^5
x^6
x^7
```

Aha! x^7 grew faster than 10^x in the region from 0 to 4. But wait: we have been told that exponential functions will always outpace polynomials eventually. So how do we resolve that?

Experiment: how far out do you have to go in x before $10^x > x^7$? How would you explore that?

(Here we pause while everyone tries it out on their own notepad, and then we compare what we came up with.)

Now let us explore smaller bases for the exponential function:

```
$ gnuplot
# in gnuplot:
reset
set grid
set xrange [-4:4]
```

(continues on next page)

(continued from previous page)

```
plot 10**x
replot 2**x
# 2**x was tiny!  how does it compare to polynomials?
plot 2**x
replot x**2
# zoom in closer:
set xrange [0:5]
plot 2**x
replot x**2
```

```
10^x
2^x
x^2
```

With a lot of zooming in, this shows that 2^x eventually outgrows x^2. Exercise: explore how long it takes for 2^x to outgrow x^7?

And with very very small bases:

```
$ gnuplot
# in gnuplot:
set xrange [0:20]
plot 1.1**x
plot x**7
```

```
1.1^x
x^7
```

show how long it takes for 1.1^x to outgrow x^7.

13.6.2 The number e: base for natural exponentials and logarithms

Now let us start looking at base e and why it is such a special number. We will mostly shift to working out of the OpenSTAX Algebra and Trigonometry book, the chapter on exponential and logarithmic functions. But we will mention here that $\exp(x) = e^x$ uses a special number e as a base, and we experiment with calculating e like this:

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \cdots = \sum_{k=0}^{\infty} \frac{1}{k!} \approx 2.71828182846$$

But we take a break from these notes as we get to use a proper text book to explore exponentials and logarithms in more detail.

13.6.3 The Taylor series for e^x

```
$ gnuplot
## then the following lines have the prompt gnuplot> and we type:
reset
set grid
set xrange [-1:3]
set terminal qt linewidth 3
```

(continues on next page)

(continued from previous page)

```
plot exp(x) lw 2
replot 1
replot 1 + x
replot 1 + x + x**2 / 2!
replot 1 + x + x**2 / 2! + x**3 / 3!
replot 1 + x + x**2 / 2! + x**3 / 3! + x**4 / 4!
replot 1 + x + x**2 / 2! + x**3 / 3! + x**4 / 4! + x**5 / 5!
replot 1 + x + x**2 / 2! + x**3 / 3! + x**4 / 4! + x**5 / 5! + x**6 / 6!
```

Discuss what you see here, then expand the x range and plot again:

```
# then try to expand the x range and replot
set xrange [-1:8]
replot
```

13.7 Miscellaneous Taylor expansions

Logarithms:

$$log(1-x) = -\sum_{k=1}^{\infty} \frac{x^k}{k}$$

$$log(1+x) = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{x^k}{k}$$

the first when $|x| < 1$, the second when $-1 < x \leq 1$

Note that when you plot these logarithmic functions you will need to double check that your plotting program uses $log()$ for *natural* logarithms. Some of the use $\ln()$

Geometric series:

$$\frac{1}{1-x} = \sum_{k=0}^{\infty} x^k \tag{13.7.1}$$

$$\frac{1}{(1-x)^2} = \sum_{k=1}^{\infty} kx^{k-1} \tag{13.7.2}$$

$$\frac{1}{(1-x)^3} = \sum_{k=2}^{\infty} \frac{(k-1)k}{2} x^{k-2} \tag{13.7.3}$$

when $|x| < 1$

13.8 Some square root expansions

Square root functions can get complicated. For example, the relativistic formula for the rest *plus* kinetic energy of an object with mass m_0 is

$$E_{\text{total}} = \frac{m_0 c^2}{\sqrt{1 - \frac{v^2}{c^2}}}$$

This has the famous Lorenz gamma factor:

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

We sometimes use a shorthand $\beta = v/c$, where β is the velocity expressed as a *fraction of the speed of light*, and get:

$$\gamma = \frac{1}{\sqrt{1 - \beta^2}}$$

The first few terms in the taylor series expansion in β are (see the Cupcake Physics link in the resources chapter for details):

$$\gamma = 1 \quad + \tfrac{1}{2}\beta^2 + \tfrac{3}{8}\beta^4 + \tfrac{5}{16}\beta^6 + \dots \tag{13.8.1}$$

$$= \qquad 1 \qquad\qquad + \frac{1}{2}\frac{v^2}{c^2} + \frac{3}{8}\frac{v^4}{c^4} + \frac{5}{16}\frac{v^6}{c^6} + \dots \tag{13.8.2}$$

Putting this back into the formula for energy we get:

$$E_{\text{kinetic}} = \frac{m_0 c^2}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{13.8.3}$$

$$= m_0 c^2 + \frac{1}{2} m_0 v^2 + \dots \tag{13.8.4}$$

For low values of v^2/c^2 (i.e. v much slower than the speed of light) we have:

$$E_{\text{total}} = m_0 c^2 + \frac{1}{2} m_0 v^2 + \dots$$

We can read off the terms and realize that the total energy is equal to the famous rest mass $E_{\text{rest}} = m_0 c^2$ plus the kinetic energy $\frac{1}{2} m_0 v^2 + \dots$:

$$E_{\text{total}} = E_{\text{rest}} + E_{\text{kinetic}} = m_0 c^2 + \frac{1}{2} m_0 v^2 + \frac{3}{8} m_0 \frac{v^4}{c^2} \dots$$

Let us explore the Lorenz gamma factor for values of v in the whole range from 0 to c:

```
$ gnuplot
## then the following lines have the prompt gnuplot> and we type:
reset
set grid
set ylabel '\gamma'
set xlabel '\beta (v^2/c^2)'
set xrange [0:1]
set terminal qt linewidth 3
plot 1 / (1 - x**2)
```

What insight does this give us on the energy of an object as it approaches the speed of light? Note that the formulae for length and time are:

$$L = \frac{1}{\gamma} L_0$$

$$\Delta t' = \gamma \Delta t$$

so the behavior of γ as a function of β (and thus v) also affects length and time.

Now let us look at the polynomial approximates in β:

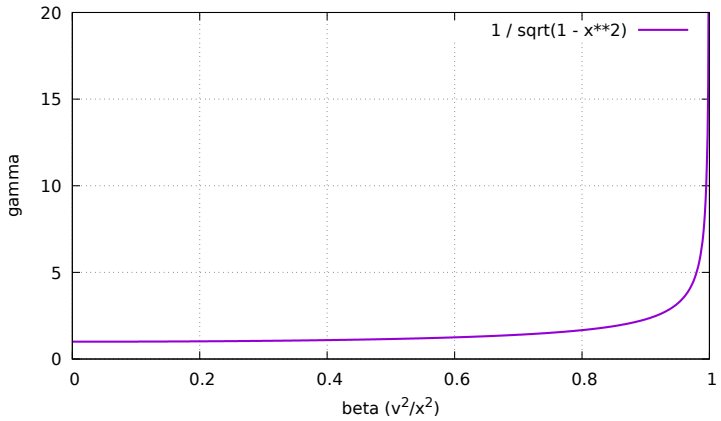

Figure 13.8.1: The lorenz factor as a function of $\beta = v^2/c^2$. Note how it is close to 1 for most of the run, but grows out of control when v approaches the speed of light c.

```
$ gnuplot
## then the following lines have the prompt gnuplot> and we type:
reset
set grid
set ylabel '\gamma'
set xlabel '\beta (v^2/c^2)'
set xrange [0:0.0001]
set terminal qt linewidth 3
plot 1 / (1 - x**2)
replot 1
replot 1 + (1.0/2) * x**2
replot 1 + (1.0/2) * x**2 + (3.0/8) * x**4
replot 1 + (1.0/2) * x**2 + (3.0/8) * x**4 + (5.0/16) * x**6
```

13.9 The pendulum: the equation and how to simplify it

The "simple pendulum" is a classic physics setup shown in Figure 13.9.1.

Here is how to think about these diagrams: the quantity θ is a function of time – we could write it fully as $\theta(t)$, since it will change with time as the pendulum swings.

Our scientific question then becomes: can you "solve" this equation, writing an expression:

$$\theta(t) = \mathrm{SomeFunctionExpression(t)}$$

The terminology used in physics is that we need to "solve Newton's second equation" to find θ.

Looking at the force diagram in the picture, we see focus on a very short bit of the *arc* of the circle that the pendulum's mass is constrained to travel. That arc leads from the current position.

From geometry we know that the length of a bit of arc is:

$$\Delta\mathrm{ArcLength} = l\Delta(\theta)$$

where l is the length of the string. That expression $l\theta$ is what will be used as a displacement in the classical physics equations.

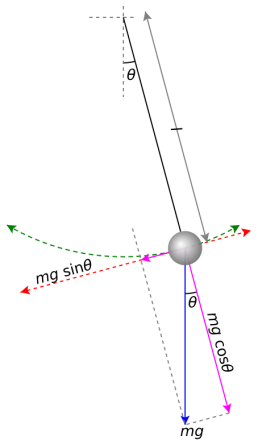

Figure 13.9.1: A force diagram of a simple pendulum. Because of the constraint of the string, the force of gravity acting on the mass *in the direction of montion* is $mg\sin(\theta)$

(Figure credit: wikipedia https://commons.wikimedia.org/wiki/File:Pendulum_gravity.svg licenced under the CC BY-SA 3.0 license.)

Some simple trigonometry will tell us that for this system Newton's 2nd law ($F = m\frac{d^2(l\theta)}{dt^2}$), combined with the force of gravity for a falling body ($F_{\text{gravity}} = -mg\sin(\theta)$) will give us (after we simplify for m which appears on both sides):

$$l\frac{d^2\theta}{dt^2} = -g\sin(\theta)$$
$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0$$

We use the name ω_0 (angular frequency) to refer to $\sqrt{l/g}$, and we get:

$$\frac{d^2\theta}{dt^2} + \omega_0^2\sin(\theta) = 0$$

At this time we are not yet looking at differential equations in detail, so we will simply mention (for those who have already studied them) that the *general* solution to this is very very difficult to find: it involves some advanced and subtle mathematical techniques, and the calculation of what are called *elliptical integrals*.

For a discussion of the general solution you can follow this video:

https://www.youtube.com/watch?v=efvT2iUSjaA

But the important thing to say here is that if θ is a *small* angle, then we can approximate it with: $\sin(\theta) \approx \theta$ and our equation becomes:

$$\frac{d^2\theta}{dt^2} + \omega_0^2\theta = 0$$

or

$$\frac{d^2\theta}{dt^2} = -\omega_0^2\theta$$

Now we can do some experiments to say: "hey, if you have a function where the slope of the slope of that function is equal to minus the function itself, what does that look like?

We will save the full study of differential equation (even this simpler one) for later on in the working group, but we will give ourselves an idea with some plots.

First of all: let us look at the plot of an exponential function. How does the slope of that plot change as we move out on the function?

Then let us plot the $\sin(x)$ and $\cos(x)$ functions one above the other. we will notice that the slope of one looks a lot like the other one.

And the slope of the other one looks a lot like the first one, but negative.

And our mind that loves to make connections will notice that: "the slope of the slope of $\sin(x)$ is…!"

13.10 Taylor series, and an intuition on why they work

13.10.1 Nomenclature

Remember: we always want to demistify terminology, so let's see what names mathematicians use to talk about these series we have experimented with.

The kinds of series we work with most of the time are called *power series*. They have the form:

$$\sum_{k=0}^{N} c_k x^k$$

where c_k are constant *coefficients*. The name "power series" comes from the fact we have increasing powers of x.

There is a particular type of power series called the *Taylor* series. The Taylor series is a wonderful tool which allows you to approximate a function near a certain point, let's call it a. It looks like:

$$S(x) = \sum_{k=1}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{13.10.1}$$

This formula is dense, so let's unpack the two parts of it.

There are the coefficients, which are constants (they do not depend on x): $\frac{f^{(k)}(a)}{k!}$.

And there is the power term $(x-a)^k$, which does depend on x.

So this looks like a polynomial of very high degree (you could almost say inifinite degree).

The series we saw above for $sin(x)$, $cos(x)$, and e^x are all examples of Taylor series. They are all centered at zero, and the coefficients are the *derivatives* of the function, evaluated at zero. In class we can work out what all those derivatives are, and check that the formula we have been using is indeed the Taylor series.

You can understand this formula at two levels: you can either say "sure, when I made my plots I noticed that they approximate the sin, cos, and exponential functions nicely."

Or you can say "hey that's really cool: I wonder how those high order derivatives come in to it".

13.10.2 Intuition on the Taylor Series derivatives

This is a good topic to develop with the first class. By looking at Figure 13.5.1 we can see how the various higher derivatives in the sin function in Equation (13.10.1) nudge our series to get closer and closer to the actual value of the function.

[This writeup will continue when the working group has come up with a good way of describing that intuition.]

13.11 A historical diversion: Bhaskara I's formula

https://en.wikipedia.org/wiki/Bhaskara_I%27s_sine_approximation_formula

CHAPTER

FOURTEEN

FOURIER SERIES: THE "BONES" OF A FUNCTION

There is often more to a plot than immediately meets the eye, and in the life of a scientist one of the great joys is to glean more information than what is on the surface.

Here we will show one of my favorite tools for extracting further information from data: the *Fourier series*. The result of this transformation is referred to as the *Fourier spectrum* of a signal.

Let us immediately dive in with a surprising example: breaking the *square wave* down into a sum of very round sin waves.

14.1 Fourier analysis: the square wave

Square waves are used as "clocks" in electronics: the edges can trigger logic circuits to do an action simultaneously with other circuits.

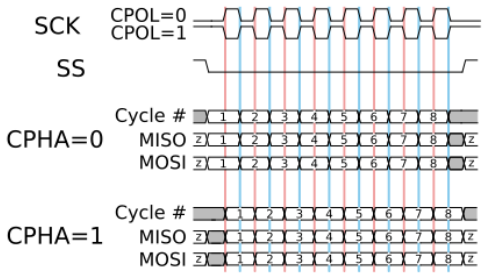

Figure 14.1.1: An example of timing diagram: the Synchronous Serial Peripheral Interface used for embedded system communication. Credit: Wikipedia

Let us start gnuplot and plot a sin wave, then look at some sin waves with higher frequencies. Then add them together and see what you get:

```
$ gnuplot
# and at the gnuplot> prompt:
set samples 1000
plot [] [-1.2:1.2] sgn(sin(x))
replot (4.0/pi) * sin(x)
replot (4.0/pi) * (1.0/3)*sin(3*x)
replot (4.0/pi) * (1.0/5)*sin(5*x)
replot (4.0/pi) * (sin(x) + (1.0/3)*sin(3*x) + (1.0/5)*sin(5*x))
## now look at the summed-up plot by itself:
plot (4.0/pi) * (sin(x) + (1.0/3)*sin(3*x) + (1.0/5)*sin(5*x)) t '5x'
```

In an online plotting calculator you might have:

```
sgn(sin(x))
(4/\pi) sin(x)
(4/\pi) (sin(x) + 1/3 sin(3 x))
(4/\pi) (sin(x) + 1/3 sin(3 x) + 1/5 sin(5 x))
(4/\pi) (sin(x) + 1/3 sin(3 x) + 1/5 sin(5 x) + 1/7 sin(7 x))
(4/\pi) (sin(x) + 1/3 sin(3 x) + 1/5 sin(5 x) + 1/7 sin(7 x) + 1/9 sin(9 x))
(4/\pi) (sin(x) + 1/3 sin(3 x) + 1/5 sin(5 x) + 1/7 sin(7 x) + 1/9 sin(9 x) + 1/11␣
↪sin(11 x))
```

You should start seeing that you go from a sin wave to something that looks a bit more like a square wave. shows what the individual sin waves look like and shows how you can add up to 13 of them and get something that starts to look quite square instead of wavy.

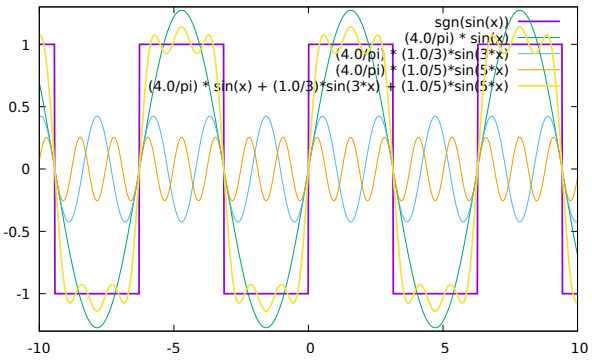

Figure 14.1.2: The first few terms of the Fourier series for the square wave.

This kind of series that sums up to the square function is called the *Fourier series* for the square wave. The full expression is:

$$s(t) = \frac{4}{\pi} \sum_{k=1}^{\infty} \frac{\sin((2k-1)\omega t)}{2k-1}$$

where $\omega = 2\pi f$ is the *angular frequency*.

Note: Here the instructor should spend some time talking about the angular frequency. The nature of this conversation depends on how much exposure to physics the students have had. The simplest discussion of it could be that instead of counting how many times a periodic function repeats in a unit of time, we count how much angle a circular movement covers in a unit of time. Of course the angle is measured in radians, which is how we get $\omega = 2\pi f$.

More generally *you can approximate any periodic function* with period T with the series:

$$s(t) = \sum_{k=0}^{\infty} (a_k \cos(k\omega t) + b_k \sin(k\omega t))$$

where $\omega = \frac{2\pi}{T}$.

Terminology:

fourier series The way of writing a *periodic* function broken down as a sum of sin and cos waves with frequencies that are multiples of a fundamental frequency.

fourier coefficients The values a_k and b_k are the *fourier coefficients* for this function.

signal One of the most active areas of science and engineering in which Fourier analysis is used is called *signal processing*, where we record electrical signals and analyze them in various ways. In this kind of setting we often use the word *signal* to mean the function we are discussing.

fourier spectrum The ensemble of all of the fourier coefficients, $\{a_k, b_k\}$, is called the *fourier spectrum* of the signal.

fourier transform The technique that transforms a function of time into a function of frequency.

reciprocal space The space in which the fourier coefficients live. For functions of time, the reciprocal space will be space of frequencies.

time domain and frequency domain For signals that are a function of time, then the transform is a function of frequency. Both representations have the same information, so we sometimes talk about looking at the signal *in the time domain* or *in the frequency domain*.

14.2 Calculating the Fourier coefficients for periodic signals

We will talk here about periodic functions of a single real variable (we'll call it t) with period T. In these functions $f(t+T) = f(t)$.

For this arbitrary periodic function $f(t)$, the Fourier series for a signal $s(t)$ with period $\omega = \frac{2\pi}{T}$ is given by the expression shown above:

$$s(t) = \sum_{k=0}^{\infty} (A_k \cos(k\omega t) + B_k \sin(k\omega t))$$

and the coefficients are given by:

$$A_k = \frac{2}{T} \int_0^T s(t) \cos(\frac{2\pi k}{T} t) dt$$
$$B_k = \frac{2}{T} \int_0^T s(t) \sin(\frac{2\pi k}{T} t) dt$$

How do you derive these expressions for the a_n and b_n? Start by remembering this fact from integration:

$$\int_0^{2\pi} \cos(mx) \cos(nx) dx = \begin{cases} \pi & \text{if } m = n \\ 0 & \text{if } m \neq n \end{cases}$$

(Similar formulas apply to the $\sin()$ function.)

This allows us to see that multiplying our series by $\cos(m\omega t)$ will cause some simplifications when $m \neq n$. We won't look at the full derivation, which can be found as a straightforward result of searching for *fourier series derivation* - I just wanted to point out that this kind of integral over a full period of trigonometric functions allows us to pick out the coefficients.

Note: The fourier series is also sometimes written as:

$$s(t) = \sum_{k=1}^{\infty} D_k \cos\left(\frac{2\pi}{T} kt - \phi_k\right)$$

where the coefficients are D_k and ϕ_k. The trigonometric identity for $\cos(\alpha - \beta)$ will quickly show you that you can translate this into our sin + cos formula, mapping these coefficients to the A_k and B_k.

Caution: The Fourier transform is so often used in science, mathematics, engineering, and music that scholars have come up with many different conventions for how to write it. All these different conventions have slightly different notations or choices on whether to put certain factors in certain places, but they all end up having the same information. This means that when you read a paper where Fourier analysis is used, you have to look carefully to understand what convention they use. If the author did not specify a convention, then you should resent their sloppiness.

14.3 Developing intuition for the fourier coefficients

An expression like:

$$A_k = \frac{2}{T}\int_0^T s(t)\cos(\frac{2\pi k}{T}t)dt = \frac{2}{T}\int_0^T s(t)\cos(k\omega t)dt$$

can be daunting for a student: it has very many parts, and the instructor should spend some time breaking it down. What I do is to discuss all the letters that appear in the formula, preferring the second form that has $\cos(k\omega t)$

A_k This is the coefficient that will multiply the $\cos()$ term in the Fourier series. It has some analogy to the a_k term in the Taylor series $f(x) = \sum_{k=0}^{\infty} a_k x^k$

k The k will be the *integer multiple* of the *fundamental frequency*. Remember... we're talking about periodic signals $s(t)$ which has angular frequence $\omega = \frac{2\pi}{T}$

T The full period, which is related to ω as I just showed. We integrate over a full period because the function is periodic, so all the information is contained in a full period.

t The variable t *inside the integral* is just an integration variable - it does not appear outside. It will be used to step through the infinitely many little rectangles that form the integral.

$k\omega t$ I find this to be the most interesting thing to notice about the calculation of the transform: it is good to remember often that we are looking at trigonometric functions with *integer multiples of a fundamental frequency* ω. That fundamental frequence comes from the fact that the original signal is periodic.

After discussing those in detail, I often go to an online graphical calculator and have the students work with me at plotting a few periodic functions that are multiples of $\sin()$ or $\cos()$, and then of the square wave multiplies by $sin()$.

This picture allows the students to see that the integral of the *even* terms over a period will always be zero, and that of the *odd* terms will not.

But maybe the best intuition will come from looking at the next section: a tour of some well-known periodic functions and their Fourier series.

14.4 A tour of functions and their Fourier series

We now apply our formulas for a_n and b_n to a few common types of periodic signals. Remember that if it is periodic then we can just act like the function is defined on the interval $[0 : 2\pi]$ or $[-\pi : \pi]$. To get the full extent of the function we just duplicate that single period forever in both directions.

14.4.1 Sawtooth wave

$$f_{\text{sawtooth}}(t) = \frac{1}{2} + \frac{t}{2} \text{ for } -1 < t \leq 1$$

The coefficients are:

$$A_0 = \frac{1}{2} \tag{14.4.1}$$
$$A_k = 0 \quad \text{for } k > 0 \tag{14.4.2}$$
$$B_k = -\frac{1}{\pi}\frac{(-1)^k}{k} \tag{14.4.3}$$

So the series is:

$$f_{\text{sawtooth}}(t) = \frac{1}{2} - \frac{1}{\pi}\sum_{k=1}^{\infty}\frac{(-1)^k}{k}\sin(k\pi t/T)$$

Let us plot this (using a period of $T = 1$) with:

```
reset
set grid
set samples 1000
set xrange [-2:2]
set yrange [-1.3:1.3]
# set terminal qt lw 2
plot 0 lw 1 lc 'black'
replot [-1:1] 1./2 + x/2 lw 2 lc 'green'
set arrow from -1, 0 to -1, 1 nohead lc 'green' lw 2
replot [1:2] 1./2 + x/2 - 1 lw 2 lc 'green'
set arrow from 1, 0 to 1, 1 nohead lc 'green' lw 2
replot [-2:-1] 1./2 + x/2 + 1 lw 2 lc 'green'
# that was the sawtooth wave.  pause to contemplate it, and then
# start with the Fourier series:
replot 1./2
replot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1)
replot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2)
replot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2 + (-1)**3 *␣
↪sin(3*pi*x)/3 + (-1)**4 * sin(4*pi*x)/4 + (-1)**5 * sin(5*pi*x)/5 + (-1)**6 *␣
↪sin(6*pi*x)/6)
replot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2 + (-1)**3 *␣
↪sin(3*pi*x)/3 + (-1)**4 * sin(4*pi*x)/4 + (-1)**5 * sin(5*pi*x)/5 + (-1)**6 *␣
↪sin(6*pi*x)/6 + (-1)**7 * sin(7*pi*x)/7 + (-1)**8 * sin(8*pi*x)/8 + (-1)**9 *␣
↪sin(9*pi*x)/9 )
replot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2 + (-1)**3 *␣
↪sin(3*pi*x)/3 + (-1)**4 * sin(4*pi*x)/4 + (-1)**5 * sin(5*pi*x)/5 + (-1)**6 *␣
↪sin(6*pi*x)/6 + (-1)**7 * sin(7*pi*x)/7 + (-1)**8 * sin(8*pi*x)/8 + (-1)**9 *␣
↪sin(9*pi*x)/9 + (-1)**10 * sin(10*pi*x)/10 + (-1)**11 * sin(11*pi*x)/11 + (-1)**12 *␣
↪sin(12*pi*x)/12 + (-1)**13 * sin(13*pi*x)/13 + (-1)**14 * sin(14*pi*x)/14 + (-1)**15 *␣
↪sin(15*pi*x)/15 + (-1)**16 * sin(16*pi*x)/16 + (-1)**17 * sin(17*pi*x)/17 + (-1)**18 *␣
↪sin(18*pi*x)/18 + (-1)**19 * sin(19*pi*x)/19 + (-1)**20 * sin(20*pi*x)/20 + (-1)**21 *␣
↪sin(21*pi*x)/21 + (-1)**22 * sin(22*pi*x)/22 + (-1)**23 * sin(23*pi*x)/23 + (-1)**24 *␣
↪sin(24*pi*x)/24 + (-1)**25 * sin(25*pi*x)/25 + (-1)**26 * sin(26*pi*x)/26 + (-1)**27 *␣
↪sin(27*pi*x)/27 + (-1)**28 * sin(28*pi*x)/28 + (-1)**29 * sin(29*pi*x)/29 + (-1)**30 *␣
↪sin(30*pi*x)/30 +  sin(31*pi*x)/31 )
# contemplate this for a moment and then let's simplify it by just
```

(continues on next page)

(continued from previous page)

```
# plotting the last of those partial series:
reset
set samples 1000
plot 1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2 + (-1)**3 *␣
↪sin(3*pi*x)/3 + (-1)**4 * sin(4*pi*x)/4 + (-1)**5 * sin(5*pi*x)/5 + (-1)**6 *␣
↪sin(6*pi*x)/6 + (-1)**7 * sin(7*pi*x)/7 + (-1)**8 * sin(8*pi*x)/8 + (-1)**9 *␣
↪sin(9*pi*x)/9 + (-1)**10 * sin(10*pi*x)/10 + (-1)**11 * sin(11*pi*x)/11 + (-1)**12 *␣
↪sin(12*pi*x)/12 + (-1)**13 * sin(13*pi*x)/13 + (-1)**14 * sin(14*pi*x)/14 + (-1)**15 *␣
↪sin(15*pi*x)/15 + (-1)**16 * sin(16*pi*x)/16 + (-1)**17 * sin(17*pi*x)/17 + (-1)**18 *␣
↪sin(18*pi*x)/18 + (-1)**19 * sin(19*pi*x)/19 + (-1)**20 * sin(20*pi*x)/20 + (-1)**21 *␣
↪sin(21*pi*x)/21 + (-1)**22 * sin(22*pi*x)/22 + (-1)**23 * sin(23*pi*x)/23 + (-1)**24 *␣
↪sin(24*pi*x)/24 + (-1)**25 * sin(25*pi*x)/25 + (-1)**26 * sin(26*pi*x)/26 + (-1)**27 *␣
↪sin(27*pi*x)/27 + (-1)**28 * sin(28*pi*x)/28 + (-1)**29 * sin(29*pi*x)/29 + (-1)**30 *␣
↪sin(30*pi*x)/30 +  sin(31*pi*x)/31 )
```

Users of geogebra can simply punch in:

```
1./2 - 1/(pi)*( (-1)**1 * sin(1*pi*x)/1 + (-1)**2 * sin(2*pi*x)/2 + (-1)**3 *␣
↪sin(3*pi*x)/3 + (-1)**4 * sin(4*pi*x)/4 + (-1)**5 * sin(5*pi*x)/5 + (-1)**6 *␣
↪sin(6*pi*x)/6 + (-1)**7 * sin(7*pi*x)/7 + (-1)**8 * sin(8*pi*x)/8 + (-1)**9 *␣
↪sin(9*pi*x)/9 + (-1)**10 * sin(10*pi*x)/10 + (-1)**11 * sin(11*pi*x)/11 + (-1)**12 *␣
↪sin(12*pi*x)/12 + (-1)**13 * sin(13*pi*x)/13 + (-1)**14 * sin(14*pi*x)/14 + (-1)**15 *␣
↪sin(15*pi*x)/15 + (-1)**16 * sin(16*pi*x)/16 + (-1)**17 * sin(17*pi*x)/17 + (-1)**18 *␣
↪sin(18*pi*x)/18 + (-1)**19 * sin(19*pi*x)/19 + (-1)**20 * sin(20*pi*x)/20 + (-1)**21 *␣
↪sin(21*pi*x)/21 + (-1)**22 * sin(22*pi*x)/22 + (-1)**23 * sin(23*pi*x)/23 + (-1)**24 *␣
↪sin(24*pi*x)/24 + (-1)**25 * sin(25*pi*x)/25 + (-1)**26 * sin(26*pi*x)/26 + (-1)**27 *␣
↪sin(27*pi*x)/27 + (-1)**28 * sin(28*pi*x)/28 + (-1)**29 * sin(29*pi*x)/29 + (-1)**30 *␣
↪sin(30*pi*x)/30 +  sin(31*pi*x)/31 )
```

As for Desmos, it has this problem where it does not handle a pasting-in of exponents with more than one digit. So you can punch in this expression where exponents are surrounded by `{}`. In addition π is written as `\pi` instead of `pi` to make the string pasting work for Desmos.

First put in the linear rising bit:

```
1 / 2 + x / 2
```

Then paste in a few terms of the sin approximation:

```
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2)
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2 + (-1)^{3} *␣
↪sin(3*\pi*x)/3)
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2 + (-1)^{3} *␣
↪sin(3*\pi*x)/3 + (-1)^{4} * sin(4*\pi*x)/4)
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2 + (-1)^{3} *␣
↪sin(3*\pi*x)/3 + (-1)^{4} * sin(4*\pi*x)/4 + (-1)^{5} * sin(5*\pi*x)/5 + (-1)^{6} *␣
↪sin(6*\pi*x)/6 + (-1)^{7} * sin(7*\pi*x)/7 + (-1)^{8} * sin(8*\pi*x)/8)
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2 + (-1)^{3} *␣
↪sin(3*\pi*x)/3 + (-1)^{4} * sin(4*\pi*x)/4 + (-1)^{5} * sin(5*\pi*x)/5 + (-1)^{6} *␣
↪sin(6*\pi*x)/6 + (-1)^{7} * sin(7*\pi*x)/7 + (-1)^{8} * sin(8*\pi*x)/8 + (-1)^{9} *␣
↪sin(9*\pi*x)/9 + (-1)^{10} * sin(10*\pi*x)/10 + (-1)^{11} * sin(11*\pi*x)/11 + (-1)^
↪{12} * sin(12*\pi*x)/12)
```

Then paste in a lot of them:

```
1./2 - 1/(\pi)*( (-1)^{1} * sin(1*\pi*x)/1 + (-1)^{2} * sin(2*\pi*x)/2 + (-1)^{3} *␣
↪sin(3*\pi*x)/3 + (-1)^{4} * sin(4*\pi*x)/4 + (-1)^{5} * sin(5*\pi*x)/5 + (-1)^{6} *␣
↪sin(6*\pi*x)/6 + (-1)^{7} * sin(7*\pi*x)/7 + (-1)^{8} * sin(8*\pi*x)/8 + (-1)^{9} *␣
↪sin(9*\pi*x)/9 + (-1)^{10} * sin(10*\pi*x)/10 + (-1)^{11} * sin(11*\pi*x)/11 + (-1)^
↪{12} * sin(12*\pi*x)/12 + (-1)^{13} * sin(13*\pi*x)/13 + (-1)^{14} * sin(14*\pi*x)/14␣
↪+ (-1)^{15} * sin(15*\pi*x)/15 + (-1)^{16} * sin(16*\pi*x)/16 + (-1)^{17} * sin(17*\
↪pi*x)/17 + (-1)^{18} * sin(18*\pi*x)/18 + (-1)^{19} * sin(19*\pi*x)/19 + (-1)^{20} *␣
↪sin(20*\pi*x)/20 + (-1)^{21} * sin(21*\pi*x)/21 + (-1)^{22} * sin(22*\pi*x)/22 + (-1)^
↪{23} * sin(23*\pi*x)/23 + (-1)^{24} * sin(24*\pi*x)/24 + (-1)^{25} * sin(25*\pi*x)/25␣
↪+ (-1)^{26} * sin(26*\pi*x)/26 + (-1)^{27} * sin(27*\pi*x)/27 + (-1)^{28} * sin(28*\
↪pi*x)/28 + (-1)^{29} * sin(29*\pi*x)/29 + (-1)^{30} * sin(30*\pi*x)/30 +  sin(31*\
↪pi*x)/31 )
```

(side note from Noah: Desmos tricks for sawtooth and triangle wave:)

```
Noah says:I accidentally did this last class cause I messed up the formula
Noah says:\frac{1}{2}\sum_{n=1}^{10}\frac{1}{n}\sin\left(n\pi x\right)
Noah
Noah says:\sum_{k=1}^{10}\frac{4(1-(-1)^{\left(2k-1\right)})}{\pi^{2}\left(2k-1\right)^
↪{2}}\cos((2k-1)\pi x)
Noah says:second one is the even triangle
```

14.4.2 Triangle wave

Let us look at an *even* triangle wave. The Fourier series is:

$$f_{\text{EvenTriangle}}(t) = \sum_{k=1}^{\infty} \frac{4(1-(-1)^k)}{\pi^2(2k-1)^2} \cos((2k-1)\pi t)$$

```
$ gnuplot
# at the gnuplot> prompt type:
reset
set grid
set samples 1000
set xrange [-2:2]
set yrange [-1.3:1.3]
set terminal qt lw 2
plot 0 lw 1 lc 'black'
replot [-1:0] 2*x + 1 lw 2 lc 'green'
replot [1:2] 2*(x-1) - 1 lw 2 lc 'green'
replot [-2:-1] -2*x -3 lw 2 lc 'green'
replot [0:1] -2*x + 1 lw 2 lc 'green'
# now that we have a triangle wave, we try a few terms of the
# Fourier series
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x)
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x) + 4*(1 - (-1)**3)/(pi**2 * 3**2) *␣
↪cos(3*pi*x)
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x) + 4*(1 - (-1)**3)/(pi**2 * 3**2) *␣
↪cos(3*pi*x) + 4*(1 - (-1)**5)/(pi**2 * 5**2) * cos(5*pi*x)
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x) + 4*(1 - (-1)**3)/(pi**2 * 3**2) *␣
↪cos(3*pi*x) + 4*(1 - (-1)**5)/(pi**2 * 5**2) * cos(5*pi*x) + 4*(1 - (-1)**7)/(pi**2 *␣
↪7**2) * cos(7*pi*x)
```

(continues on next page)

(continued from previous page)

```
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x) + 4*(1 - (-1)**3)/(pi**2 * 3**2) *␣
↪cos(3*pi*x) + 4*(1 - (-1)**5)/(pi**2 * 5**2) * cos(5*pi*x) + 4*(1 - (-1)**7)/(pi**2 *␣
↪7**2) * cos(7*pi*x) + 4*(1 - (-1)**9)/(pi**2 * 9**2) * cos(9*pi*x)
# after contemplating each of those, put a *ton* of terms in there:
replot 4*(1 - (-1)**1)/(pi**2 * 1**2) * cos(1*pi*x) + 4*(1 - (-1)**3)/(pi**2 * 3**2) *␣
↪cos(3*pi*x) + 4*(1 - (-1)**5)/(pi**2 * 5**2) * cos(5*pi*x) + 4*(1 - (-1)**7)/(pi**2 *␣
↪7**2) * cos(7*pi*x) + 4*(1 - (-1)**9)/(pi**2 * 9**2) * cos(9*pi*x) + 4*(1 - (-1)**11)/
↪(pi**2 * 11**2) * cos(11*pi*x) + 4*(1 - (-1)**13)/(pi**2 * 13**2) * cos(13*pi*x) +␣
↪4*(1 - (-1)**15)/(pi**2 * 15**2) * cos(15*pi*x) + 4*(1 - (-1)**17)/(pi**2 * 17**2) *␣
↪cos(17*pi*x) + 4*(1 - (-1)**19)/(pi**2 * 19**2) * cos(19*pi*x) + 4*(1 - (-1)**21)/
↪(pi**2 * 21**2) * cos(21*pi*x) + 4*(1 - (-1)**23)/(pi**2 * 23**2) * cos(23*pi*x) +␣
↪4*(1 - (-1)**25)/(pi**2 * 25**2) * cos(25*pi*x) + 4*(1 - (-1)**27)/(pi**2 * 27**2) *␣
↪cos(27*pi*x) + 4*(1 - (-1)**29)/(pi**2 * 29**2) * cos(29*pi*x) + 4*(1 - (-1)**31)/
↪(pi**2 * 31**2) * cos(31*pi*x) + 4*(1 - (-1)**33)/(pi**2 * 33**2) * cos(33*pi*x) +␣
↪4*(1 - (-1)**35)/(pi**2 * 35**2) * cos(35*pi*x) + 4*(1 - (-1)**37)/(pi**2 * 37**2) *␣
↪cos(37*pi*x) + 4*(1 - (-1)**39)/(pi**2 * 39**2) * cos(39*pi*x) + 4*(1 - (-1)**41)/
↪(pi**2 * 41**2) * cos(41*pi*x) + 4*(1 - (-1)**43)/(pi**2 * 43**2) * cos(43*pi*x) +␣
↪4*(1 - (-1)**45)/(pi**2 * 45**2) * cos(45*pi*x) + 4*(1 - (-1)**47)/(pi**2 * 47**2) *␣
↪cos(47*pi*x) + 4*(1 - (-1)**49)/(pi**2 * 49**2) * cos(49*pi*x) + 4*(1 - (-1)**51)/
↪(pi**2 * 51**2) * cos(51*pi*x) + 4*(1 - (-1)**53)/(pi**2 * 53**2) * cos(53*pi*x) +␣
↪4*(1 - (-1)**55)/(pi**2 * 55**2) * cos(55*pi*x) + 4*(1 - (-1)**57)/(pi**2 * 57**2) *␣
↪cos(57*pi*x) + 4*(1 - (-1)**59)/(pi**2 * 59**2) * cos(59*pi*x)
```

14.4.3 The Gaussian

The Fourier transform of a gaussian is fascinating from more than one point of view: the mathematical tricks used to calculate it are delightful and the applications are metaphorically rich.

Visuals on the Gaussian

Let us start by exploring the Gaussian function. We will use the form that gives the *probability density function* for a normally distributed random variable x with mean μ and standard deviation σ:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$$

If our mean is zero and standard deviation is one then we get:

$$g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}$$

The Gaussian function comes up in so many places that it pays to become very comfortable with its behavior. Let us start with some plots to get a feeling for what it looks like:

```
reset
set grid
set terminal qt lw 3
set samples 1000
set xrange [-5:5]
set yrange [0:1]
plot (1.0/(1*sqrt(2*pi))) * exp(- (1./2) * (x - 0)**2 / 1**2) title 'mean 0 sigma 1'
```

(continues on next page)

(continued from previous page)

```
replot (1.0/(2*sqrt(2*pi))) * exp(- (1./2) * (x - 0)**2 / 2**2) title 'mean 0 sigma 2'
replot (1.0/(0.5*sqrt(2*pi))) * exp(- (1./2) * (x - 0)**2 / 0.5**2) title 'mean 0 sigma␣
↪0.5'
replot (1.0/(0.4*sqrt(2*pi))) * exp(- (1./2) * (x - 1.3)**2 / 0.4**2) title 'mean 1.3␣
↪sigma 0.4'
```

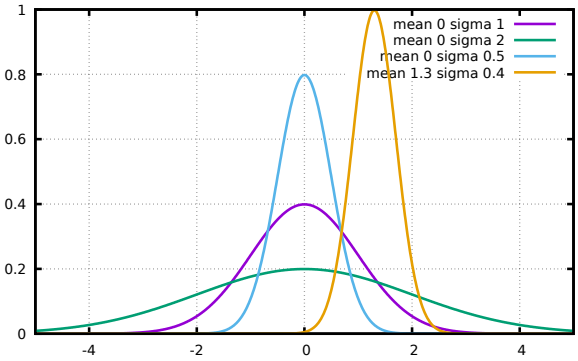

Figure 14.4.1: Gaussian curves for various standard deviations and a couple of means.

Looking closely at the formula for the Gaussian you will notice that σ appears in the denominator of the exponential, as well as the denominator of the constant scale factor $\frac{1}{\sigma\sqrt{2\pi}}$. This means that:

σ **is big** The curve will spread out and not be tall. This matches our intuition: if there is a big uncertainty then our random variable will often be quite different from the mean value.

σ **is small** The curve will be skinny and taller. The intuition here is that random values are likely to be close to the mean, and not *deviate* from it much.

Normalization and total probability

If we think of the Gaussian as a probability distribution function then it is important that the probabilities of a random variable being in all its possible states add up to 1. This means that we insist that this integral:

$$\int_{-\infty}^{\infty} g(x)dx = \int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} dx$$

equal 1 for all values of μ and σ.

This is guaranteed by the *Gaussian integral* formula:

$$\int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi}$$

If we substitute $x \to ax$ then we get:

$$\int_{-\infty}^{\infty} e^{-a^2x^2} d(ax) = a\int_{-\infty}^{\infty} e^{-a^2x^2} dx = \sqrt{\pi} \tag{14.4.4}$$

$$\implies \int_{-\infty}^{\infty} e^{-a^2x^2} dx = \sqrt{\pi}/a \tag{14.4.5}$$

For the Gaussian our a^2 is $1/(2\sigma^2)$, so we get:

$$\int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} dx = \sqrt{\pi}/a = \sigma\sqrt{2\pi}$$

which implies that

$$\int_{-\infty}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} dx = 1$$

This means that the Gaussian has been crafted carefully so that, no matter what σ is, we always end up with a total probability of 1.

The factor of $\frac{1}{\sigma\sqrt{2\pi}}$ which multplies our exponential function is called a *normalization constant*. Normalization constants are widely used term in mathematics: we apply a constant scale to a function so that it satisfies some necessary condition.

Note that I have not mentioned the mean in discussing these integrals. The reason is that the gaussian goes from $-\infty$ to ∞, so a shift of the hump to the right or to the left does not affect the integral at all.

Fourier-transforming the Gaussian

The first thing I have to do here is be a bit apologetic: the Gaussian is *not* a periodic function. Nor is it defined on a finite domain (which can be treated as a periodic function. Rather, it unfolds on the entire real line with no repetitions.

This means that we cannot apply the mechanisms we saw earlier to calculate the Fourier transform of the Gaussian: this will be a different beast.

But the beast does exist: you can take an arbitrary function and write it down as a *Fourier integral* (not a sum; an integral!) which integrates sin and cos functions with a continuum of frequencies, and we will get our function back.

The Fourier transform of this more general function $s(t)$ will not be a collection of discrete *coefficients* a_k, b_k (remember: those k indices were multiples of a fundamental frequency).

Instead it will involve a pair of *functions* $A(\omega), B(\omega)$ where ω is the angular frequency $\omega = 2\pi f$:

$$s(t) = \int_{-\infty}^{\infty} \left(A(\omega)\cos(\omega t) + B(\omega)\sin(\omega t)\right) d\omega$$

You then calculate $A(\omega)$ and $B(\omega)$ quite analogously to the discrete Fourier coefficients:

$$A(\omega) = \frac{2}{\pi}\int_0^{\infty} s(t)\cos(\omega t)dt$$
$$B(\omega) = \frac{2}{\pi}\int_0^{\infty} s(t)\sin(\omega t)dt$$

Let us apply this to the Gaussian. We will center it at a mean of zero then we have an even function. This means that $B(\omega) = 0$, so we have: transform:

$$g(t) = \int_{-\infty}^{\infty} A(\omega)\cos(\omega t)d\omega \tag{14.4.6}$$

where (14.4.7)

$$A(\omega) = \frac{2}{\pi}\int_0^{\infty} g(t)cos(\omega t)dt = \frac{1}{\sigma\sqrt{2\pi}}\frac{1}{\pi}\int_{-\infty}^{\infty} e^{-t^2/(2\sigma^2)}\cos(\omega t)dt \tag{14.4.8}$$

the application of some integration techniques gives us:

$$A(\omega) = \int_{-\infty}^{\infty} e^{-t^2/(2\sigma^2)}\cos(\omega t)dt = e^{-\sigma^2\omega^2/2}$$

The important thing to take home here is that *the Fourier transform of a Gaussian is another Gaussian!* And this other Gaussian has $2/\sigma$ in all the places where the original one has σ. If you let $\delta = 2/\sigma$ then you get:

$$A(\omega) = e^{-\omega^2/(2\delta^2)}$$

so we can say that:

The Fourier transform of a Gaussian is a gaussian with the reciprocal standard deviation.

Caution: The equations above for $s(t)$ and $A(\omega)$ are subject to normalization. Researchers in different fields use a different normalization convention, with factors of 2 and π (possibly with square roots) that can be applied in one formula or in the other, as long as it is consistent. The lack of a single convention is frustrating, and ends up being costly, as collaborators have to be very careful. At this time I have not yet double-checked that I am using a uniform normalization convention in the equations above. Still, the fundamental conclusion holds: the transform of a Gaussian is a Gaussian with reciprocal standard deviation.

Visualization

Here is an animated visualization of the Gaussian and its transform for values of sigma that go from 10 to 1/10.

```
set grid
set samples 1000
do for [ii=1:100] {
    sigma = 10.0 / ii
     plot 1./(sigma*sqrt(2*pi)) * exp(-x**2 / (2*sigma**2))
     replot exp(-x**2 * sigma**2 / 4)
     pause(1.3)
}
```

We will show more visualizations when we look at uses of the Fourier transform on discrete data.

Thoughts about the transform of the Gaussian

[this section is not yet elaborated]

The general idea that if you have a narrowly localized signal in time, its Fourier transform will be spread out in frequency. Conversely, if you have a spread out signal, then its Fourier transform will be concentrated in a narrow band of frequencies.

CHAPTER

FIFTEEN

FOURIER ANALYSIS ON REAL DATA

In the previous chapter we learned about Fourier Analysis and showed how to take the Fourier transform of some known functions.

Here we will show how to apply the Fourier transform to "data", for example to data coming from experiments. We will start with some artificial data, and then look at the output from an experiment.

15.1 What's hidden in noisy data?

Fourier analysis suggests that we could think of the spectrum (how strong each frequency is in a signal) as a sort of "character" of the signal: a deep description which might not be obvious on the surface, but it is there.

Here we will do some experiments to demonstrate this idea, and then we will use this idea to devise a marvel of data analysis: *signal filtering*.

15.1.1 A contrived double-period signal

Start with this plot:

```
$ gnuplot
# at the gnuplot> prompt type:
set samples 1000
plot sin(x) + 0.3*sin(365*x)
```

Then we will download the program `code/filter_toy.py` and look at how it works. Then we will run it with:

```
python3 filter_toy.py
```

to see an output like that in Figure 15.1.1.

15.1.2 An artificial noisy signal

Now we will download the program `code/filter_random.py` and look at how it works. Then we will run it with:

```
python3 filter_random.py
```

to see an output like that in Figure 15.1.2.

The instructor can now experiment changing the `0.4*np.random.randn(...)` to be `3.4*` – this will dwarf the signal with noise, but we will still find an approximate sin wave.

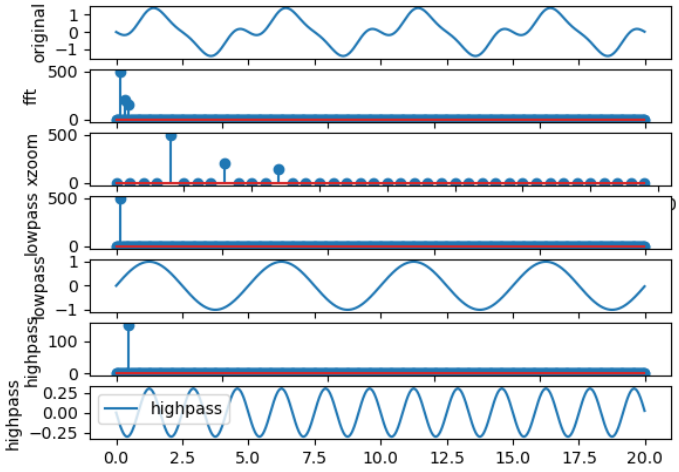

Figure 15.1.1: An example of filtering for a very simple signal: the sum of 3 sin waves. The first filter zeros out all the *high frequency* fourier coefficients (low pass filter). The second one zeroes out all the *low frequency* fourier coefficients (high pass filter).

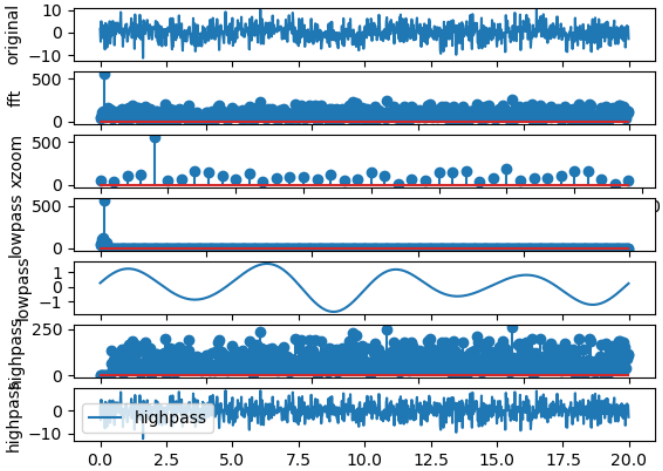

Figure 15.1.2: An example of filtering for a noisy signal: a sin() wave with random noise added to it. The first filter zeros out all the *high frequency* fourier coefficients (low pass filter). The second one zeroes out all the *low frequency* fourier coefficients (high pass filter).

Another interest change is to apply the lowpass filter to the highpass filter result (instead of to the original signal). This will give a *bandpass* filter: a narrow band around the fundamental frequency.

All these experiments can be done live in class.

15.1.3 Experimental double-period noisy data

Let us explore the data from a weather station. We will pick the Las Cruces weather station from the climate research network coordinated by NOAA (National Oceanic and Atmospheric Administration).

We will find that this data has *both* the features we saw above: it has an underlying doubly periodic behavior, and it also has noise. It also has plenty more information, related to weather (and, over the course of many years) climate change, but we will focus on finding the two periodic signals, and on the noise.

Start with:

```
wget https://www.ncei.noaa.gov/pub/data/uscrn/products/subhourly01/2014/CRNS0101-05-2014-
↪NM_Las_Cruces_20_N.txt
```

Then plot the 9th column in gnuplot with:

```
$ gnuplot
# then at the gnuplot> promt type:
plot 'CRNS0101-05-2014-NM_Las_Cruces_20_N.txt' using 9 with lines
```

We immediately see two undesirable spikes that come down to almost -10000. We spend some time looking at the file, we puzzle over it, and then realize that:

Caution: Data often needs cleaning. In this case you see that an old (and poor) programmer habit crept into the pipeline of programs that generated this data: when samples were missing they replaced them with `-9999.0`.

The files are also a jumble of numbers, so to understand what is in those files we visit https://www.ncei.noaa.gov/pub/data/uscrn/products/subhourly01/headers.txt and https://www.ncei.noaa.gov/pub/data/uscrn/products/subhourly01/readme.txt to see what columns mean what.

We conclude that we have data spaced out by 5 minutes, and the 9th column is a temperature reading, and that we need to eliminate all lines that have -9999.0 in them:

```
grep -v -e -9999.0 CRNS0101-05-2014-NM_Las_Cruces_20_N.txt > Las_Cruces_filtered.txt
$ gnuplot
# then at the gnuplot> prompt type:
plot 'Las_Cruces_filtered.txt' using 9 with lines
```

We might want to look at more than one year of data. We can do this with:

```
for year in 2014 2015 2016
do
    wget https://www.ncei.noaa.gov/pub/data/uscrn/products/subhourly01/${year}/CRNS0101-
↪05-${year}-NM_Las_Cruces_20_N.txt
done
cat CRNS*Las_Cruces*.txt | grep -v -e -9999.0 > temperatures.dat
```

and now we have several years to plot:

```
$ gnuplot
# then at the gnuplot> prompt type:
plot 'temperatures.dat' using 9 with lines
```

Now let us download the program `code/temperature_fft.py` that analyzes this data with the fourier transform, and run it:

```
python3 temperature_fft.py
```

Caution: These are big data files: 5 minute intervals for a year make for more than 100 thousand samples per year. Three years of data will require the Fourier transform of a signal with more than 300000 samples, which might strain the CPU and/or memory of your computer. This might be even more of a problem if you use an web-based python interpreter.

15.2 A breather and some terminology

We have just gone through three examples of using the Fourier transform on real data. This was a lot of new material, so let us take a moment and discuss what we have seen.

First some terminology that accompanies these new ideas:

Continuous Fourier Transform The coefficients of the sin and cos functions at various frequencies which allow us to reproduce a periodic function of *continuous* time. We discussed this in Section 14

Discrete Fourier Transform (DFT) A form of Fourier analysis that is applicable to a sequence of values. Instead of a function $f(t)$ with t being a real number, we have a function given by pairs t_k, f_k. The t_k values are discrete time samples, and f_k are the samples of the function at those time points. The DFT is clearly what we will be using to examine experimental data.

Fast Fourier Transform (FFT) An algorithm to calculate the DFT in $O(N \log N)$ time instead of $O(N^2)$. This is also known as the Cooley Tuckey algorithm.

So is it DFT or FFT? The FFT is used almost universally to calculate the DFT, so FFT is sometimes used to mean the specific algorithm used, and sometimes just as a placeholder for DFT.

Signal processing A field of electrical engineering focused on analyzing, modifying, and synthesizing signals. Signals can refer to sound, images, and scientific measurements.

Noise Unwanted distortions that a signal can suffer during propagation, transmission, and processing. Often the specific origin of the noise cannot be picked out, but it can be modeled in various ways that help with finding a signal that might otherwise be obscured.

White noise A form of noise that gets added to a signal and that has equal intensity at different frequencies. In acoustics it sounds like a hissing sound.

15.3 Fourier analysis: sound and music

When we mentioned waves with different frequencies you might have thought of sound, and you would have been right. Fourier analysis is a good tool for understanding what makes up sound waves.

Let us take a tour through a series of signals, and we will look at their fourier transforms.

15.3.1 Tuning fork

A tuning fork puts out a very pure single $\sin$ wave at a "middle A" note, also known as A_4 or concert pitch. A_4 has a frequency of 440 Hz.

We can download a stock mp3 file of the sound of a tuning fork, then we can use standard command line utilities to convert it to a text file. For example:

```
wget https://freesound.org/data/previews/126/126352_2219070-lq.mp3 -O tuning-fork.mp3
# convert through various steps to .aif and then to .dat
ffmpeg -i tuning-fork.mp3 tuning-fork.aif
sox tuning-fork.aif tuning-fork.dat
ls -lsh tuning-fork.*
# check out what it sounds like with
vlc tuning-fork.mp3
```

You can plot it with:

```
gnuplot
# then at the gnuplot> prompt:
plot 'tuning-fork.dat' using 1:2 with lines
```

This shows you the plot of amplitude versus time, and you should notice some correspondence between it and the sound you heard - the person who recorded this file kept modulating how loud it was.

But that does not tell us what frequencies are in it! Let us adapt the `filter_random.py` program to work with this data file.

You can download `code/fft_audio.py` and try running it on this `tuning-fork.dat` file:

```
python3 fft_audio.py
```

So the bottom line is: once we have it as a text file we can do the following:

- Use our standard plotting techniques to see that it looks like a $\sin$ wave, but with lots of amplitude variation.
- Write a program which uses the powerful Python scientific libraries to calculate the Fourier transform of the tuning fork signal.
- Look at the Fourier transform, hoping to see that a clean $\sin$ wave will appear as a single spike, indicating that there is only one $\sin$ wave in the signal.

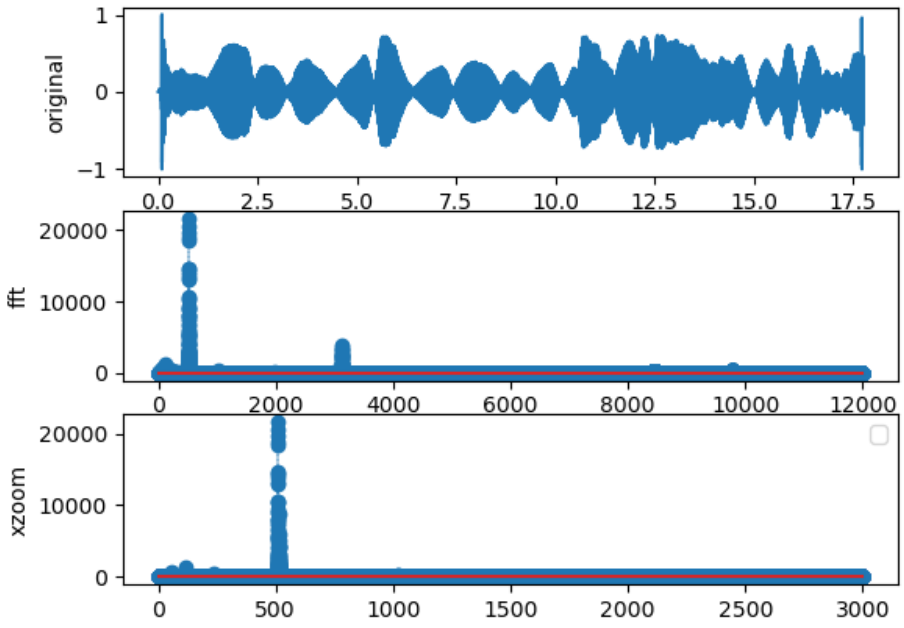

Figure 15.3.1: The FFT of a tuning fork. Notice the single dominant frequency, which shows that the tuning fork is close to being a perfect $\sin()$ wave.

15.3.2 White noise

Now let us download an example of *white noise*. This is a random wave with no discernible pattern.

```
wget -q --continue http://www.vibrationdata.com/white1.mp3 -O white-noise.mp3
ffmpeg -n -i white-noise.mp3 white-noise.aiff
sox -q white-noise.aiff -t dat white-noise.dat
head -7000 white-noise.dat | tail -2000 > white-noise-small-sample.dat
# check out what it sounds like with
vlc white-noise.mp3
```

Notice how we removed some data from the start and some from the end so as to have a shorter data file.

```
gnuplot
# then at the gnuplot> prompt:
reset
set grid
plot 'white-noise-small-sample.dat' with lines
```

Now change `fft_audio.py` to have `fname = white-noise-small-sample.dat` and run it again.

You could say that the Fourier spectrum for white noise is the opposite of that for the pure tuning fork signal: instead of a single spike you have random-looking spikes all over the spectrum.

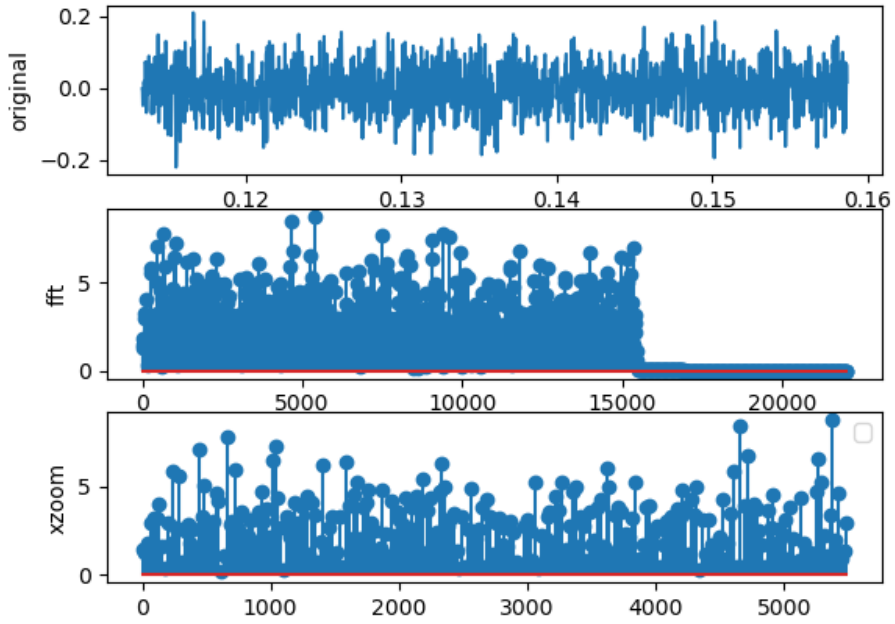

Figure 15.3.2: The FFT of a white noise sample. Notice the broad and flat value of the transform for almost all the frequencies.

15.3.3 Violin playing single “F” note

Now let us look at some musical notes from real instruments. Each note corresponds to a certain frequency [Backus, 1977]. shows a much more complex signal than the tuning fork.

```
wget --continue http://freesound.org/data/previews/153/153595_2626346-lq.mp3 -O violin-F.
↪mp3
ffmpeg -n -i violin-F.mp3 violin-F.aiff
sox violin-F.aiff -t dat violin-F.dat
python3 fft_audio.py violin-F.dat
```

The signal in has some structure to it. The Fourier transform has several peaks: the strongest peak (for a A_4 note) will be at the *fundamental frequency* of 440 Hz, but there are many other peaks. The pattern of the peaks characterizes the sound of that specific violin (or guitar or other instrument).

15.3.4 Violin playing single “A” note

This is a repeat of but with a different note on the violin: F instead of A. is hard to distinguish from since the character of the instrument is the same. At this point we have not written code to plot the actual frequencies on the x axis, so we cannot spot that the A_4 note has its highest peak at 440Hz, while the F_5 is at 698.45Hz [Backus, 1977].

```
wget --continue http://freesound.org/data/previews/153/153587_2626346-lq.mp3 -O violin-A-
↪440.mp3
ffmpeg -n -i violin-A-440.mp3 violin-A-440.aiff
```

(continues on next page)

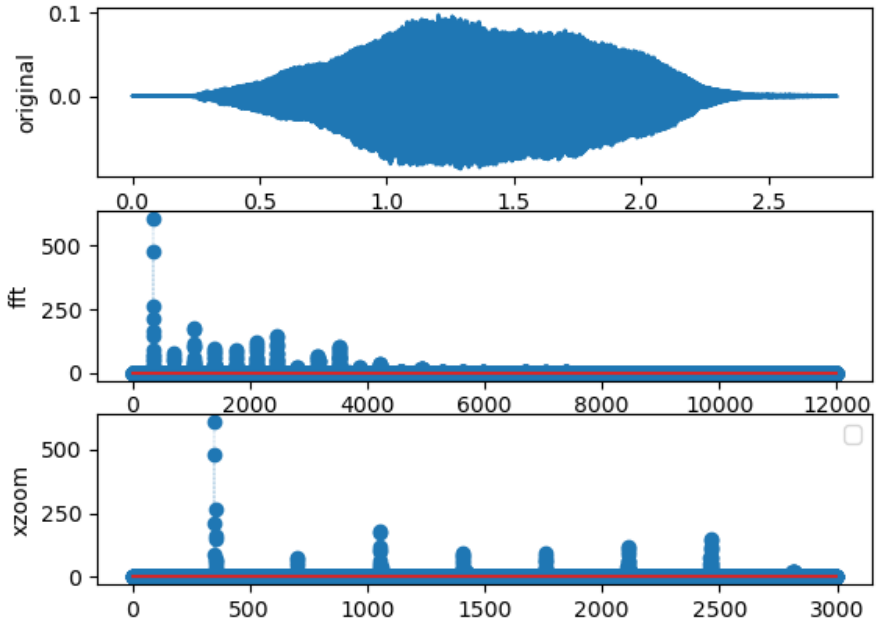

Figure 15.3.3: The FFT of a violin play an F note.

(continued from previous page)

```
sox violin-A-440.aiff -t dat violin-A-440.dat
python3 fft_audio.py violin-A-440.dat
```

15.3.5 A more complex music clip

The Pachelbel Canon is a well-known piece of baroque classical music which starts with a single note that is held for a while. shows the signal and its Fourier transform, where you can identify a dominant peak.

```
wget -q --continue http://he3.magnatune.com/music/Voices%20of%20Music/Concerto%20Barocco/
↪21-Pachelbel%20Canon%20In%20D%20Major%20-%20Johann%20Pachelbel%20-%20Canon%20And
↪%20Gigue%20For%20Three%20Violins%20And%20Basso%20Continuo%20In%20D%20Major-Voices%20of
↪%20Music_spoken.mp3 -O Canon.mp3
ffmpeg -n -i 'Canon.mp3' Canon.aiff
sox -q Canon.aiff -t dat Canon.dat
head -50000 Canon.dat | tail -4000 > canon-small-sample.dat
python3 fft_audio.py canon-small-sample.dat
```

But what happens if we have music that is not a single note? In we will look at a clip of a choir singing *Gloria in Excelsis Deo*. The clip (which you can play with `vlc gloria.ogg` after downloading it) starts in a place where many voices are singing in harmony, so there is no single note to be picked out.

```
wget -q --continue https://upload.wikimedia.org/wikipedia/en/e/ea/Angels_We_Have_Heard_
↪on_High%2C_sung_by_the_Mormon_Tabernacle_Choir.ogg -O gloria.ogg
ffmpeg -n -i gloria.ogg gloria.aiff
```

(continues on next page)

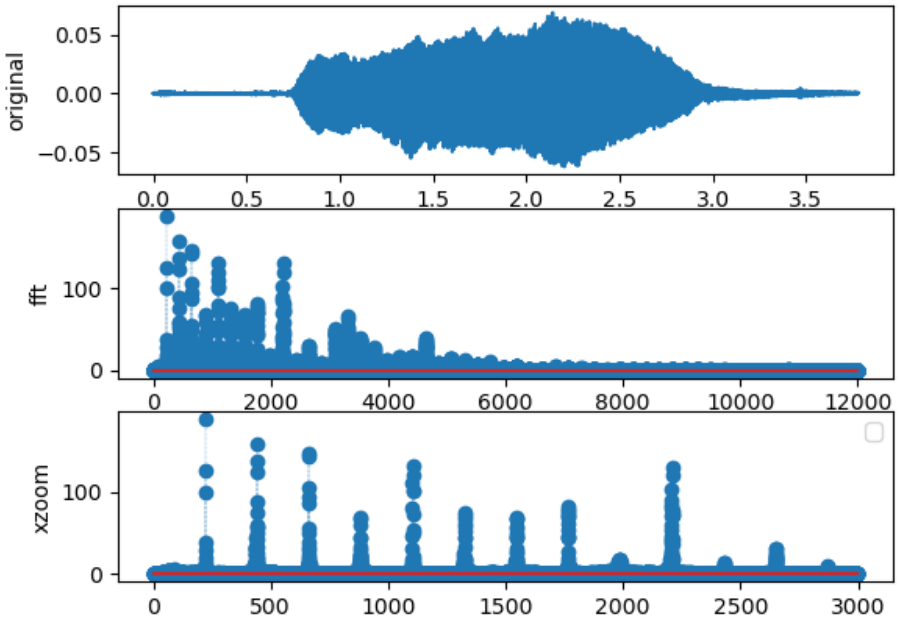

Figure 15.3.4: The FFT of a violin play a middle A note.

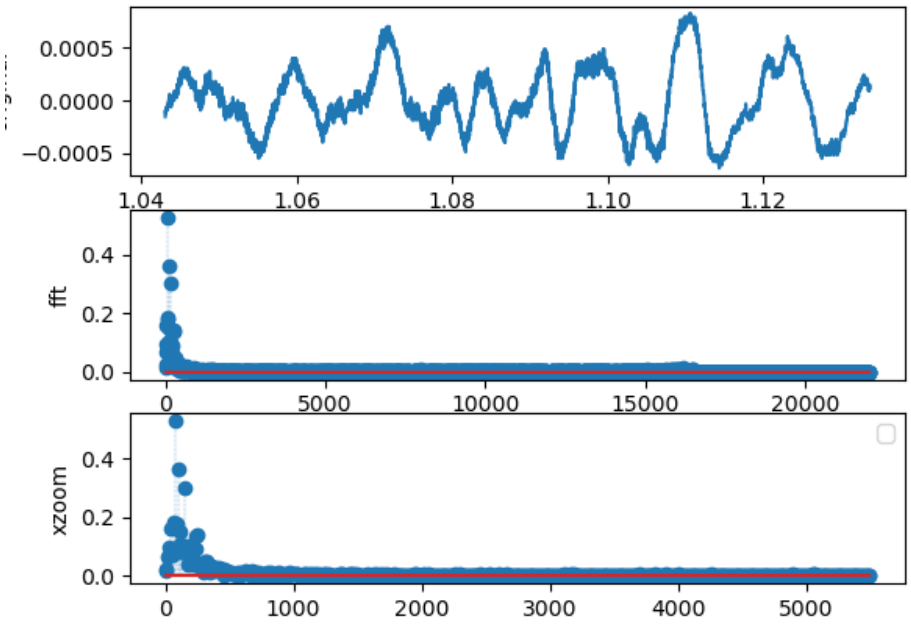

Figure 15.3.5: The FFT of the first few seconds of the Pachelbel canon.

(continued from previous page)

```
sox -q gloria.aiff -t dat gloria.dat
python3 fft_audio.py gloria.dat
```

We expect to see several different peaks in the Fourier spectrum, and that is what you see in the second panel of this plot.

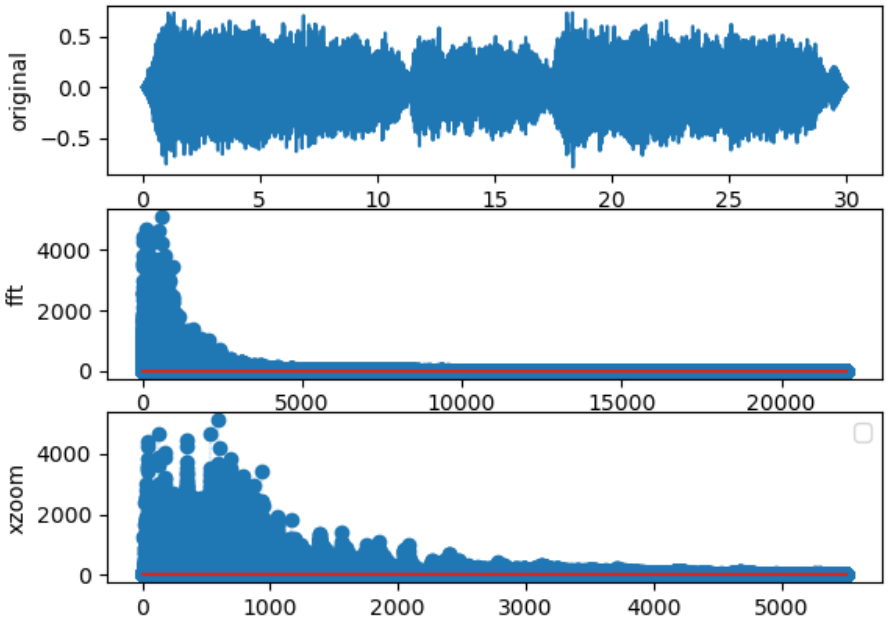

Figure 15.3.6: The FFT of the first few seconds of Gloria in Excelsis Deo.

15.3.6 Create your own audio clip and analyze it

Record some sound and analyze it.

The program `sox`, which we used to do audio format conversion, comes with two other programs `rec` (to record from your computer's microphone) and `play` to play those files back.

To try it out I ran the following:

```
rec mark-guitar-sample.dat
## I played a few guitar notes near the laptop microphone,
## emphasizing low and high frequences, then I hit control-C
play mark-guitar-sample.dat
python3 fft_audio.py mark-guitar-sample.dat
```

Now try doing this again for different things you can record with your microphone. If you have a tuning fork, tap it and then rest it on a guitar's soundboard and record that, see if you get something similar to what we saw when we discussed tuning forks. If you have a musical instrument, try recording an A note or an F note and compare them to what we discussed violins.

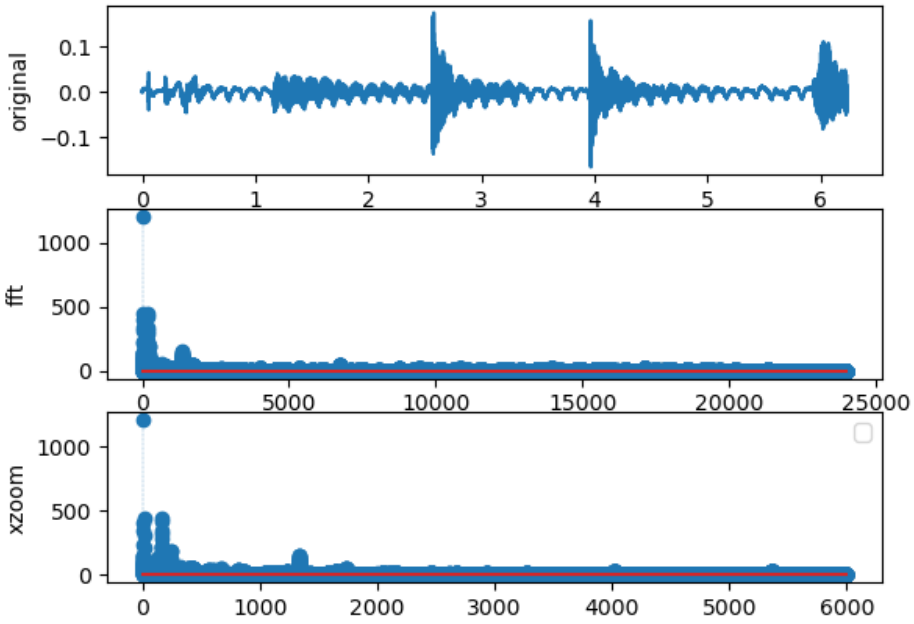

Figure 15.3.7: The FFT of my own musical sample.

CHAPTER

SIXTEEN

APPROXIMATING DIFFERENTIAL EQUATIONS

16.1 Motivation and plan

The laws of nature are expressed as differential equations.

This is because in describing nature we discuss how something changes, either from location to another, or as time goes by. These changes are expressed as *derivatives* of a function, and the laws of nature relate those derivativse to other functions.

16.2 A review of derivatives

First refer the class to a quick look at the animation of derivatives in the mini courses book, the chapter "pushing toward calculus".

Then we will simply do a few examples of this calculation. Our purpose is not to do an in-depth review, but just to feel some agility, to remember a few simple formulae, and to show some of the simplest examples to people who have not yet taken a calculus class.

$$\begin{aligned}
\frac{dx^2}{dx} &= \lim_{h\to 0} \frac{(x+h)^2 - x^2}{h} && (16.2.1)\\
&= \lim_{h\to 0} \frac{x^2 + 2xh + h^2 - x^2}{h} && (16.2.2)\\
&= \lim_{h\to 0} \frac{2xh + h^2}{h} && (16.2.3)\\
&= 2x && (16.2.4)
\end{aligned}$$

if we remember the Pascal triangle we note that $(x+h)^3 = x^3 + 3x^2h + 3xh^2 + h^3$. If we also remember that h^2 and h^3 get small very quickly and disappear in the limit, we get:

$$\begin{aligned}
\frac{dx^3}{dx} &= \lim_{h\to 0} \frac{(x+h)^3 - x^3}{h} && (16.2.5)\\
&= \lim_{h\to 0} \frac{x^3 + 3x^2h + 3xh^2 + h^3 - x^3}{h} && (16.2.6)\\
&= \lim_{h\to 0} \frac{3x^2h + 3xh^2 + h^3}{h} && (16.2.7)\\
&= 3x^2 && (16.2.8)
\end{aligned}$$

and looking at higher order binomial expansions we see that:

$$\frac{dx^n}{dx} = n \times x^{n-1}$$

People sometimes keep a mnemonic for their math formulae. I tend to think that the n "comes down in front, and gets reduced in the exponent".

Note that some basic properties hold:

$$\frac{dAf(x)}{dx} = A\frac{df(x)}{dx}$$
$$\frac{d(f(x)+g(x))}{dx} = \frac{df(x)}{dx} + \frac{dg(x)}{dx}$$

Putting those together we get that:

$$\frac{d(Af(x)+Bg(x))}{dx} = A\frac{df(x)}{dx} + B\frac{dg(x)}{dx}$$

This last property leads to the jargon that "the derivative is a *linear operator.*"

Now for some slightly more complex ones:

$$\frac{d\sin(x)}{dx} = \cos(x)$$
$$\frac{d\cos(x)}{dx} = -\sin(x)$$
$$\frac{de^x}{dx} = e^x$$
$$\frac{d\log(x)}{dx} = \frac{1}{x}$$

16.3 What is a differential equation?

A differential equation is one where you have functions of your variable and their derivatives, and you are trying to find the *function*, not the variable.

For example:

$$\frac{df(x)}{dx} = 7x + 2$$

We solve this by *educated guessing:* the derivative of $\frac{7}{2}x^2$ is $7x$, and the derivative of $2x$ is 2. This leads us to conclude that:

$$f(x) = \frac{7}{2}x^2 + 2x + C$$

where C is the famous *arbitrary constant*.

16.4 A simple example: exponential growth

An example from ecology: start with two rabbits, and these rabbits have no constraints: they can reproduce, and never run out of food, and there no predators, and lots of other simplifying assumptions. Then the population is governed by this equation:

$$\frac{dP(t)}{dt} = r * P(t)$$

Let us pick this apart.

How did you arrive to it? There are various ways in which scientists have written down equations that govern phenomena. In this case you can reason it through by saying that: *"The more rabbits you have, the higher the growth rate"*. This is the same as saying that the growth rate $(dP(t)/dt)$ is proportional to how many rabbits you already have $(P(t))$.

Is there some jargon to go with this? Of course! I usually refer to the differential equation for a function of time (in our case $P(t)$) as the injection of a *dynamic principle* into the system. Other equations might give you information about setting up an initial system, but the differential equation with respect to time introduces *dynamics* into the picture.

What about r? r can be thought of as the *rate* at which the population grows. A bigger r makes for a much faster population growth.

OK, so dynamical principle, initial, … Any other jargon? Ah yes: this was a *first order linear differential equation.*

First order differential equation The highest order of the derivative in the equation. In this case we had the first derivative of $P(t)$ so it was a *first order* differential equation.

Linear differential equation The function $P(t)$ and its derivatives always appeared *linearly*, i.e. they were all to the first power and multiplied by constants. (Think of the equation of a straight *line* $y = mx + b$. This is *linear* in x because x does not appear to higher powers, and its only operation is to be multiplied by a constant.

Can we solve this equation *analytically*? (And here you pause and ask the "class what does analytically mean?")

For this simple example yes, we can solve it analytically: we have encountered exactly *one* math function whose derivative is proportional to itself:

$$\frac{de^t}{dt} = e^t$$

and therefore:

$$\frac{de^{\alpha t}}{dt} = \alpha e^{\alpha t}$$

So the solution to our equation is:

$$P(t) = \text{constant} \times e^{rt}$$

Here you have an "arbitrary constant", as they are sometimes called. Note that the original differential equation is such that multiplying by that constant does not change the result, so the constant is a key part of the general solution.

So what is that constant physically?

We can read directly from the equation that a time $t = 0$ we will have $e^{r\times 0} = 1$, so the constant is just $P(0)$.

Using the shorthand $P_0 = P(0)$, we end up with:

$$P(t) = P_0 \times e^{rt}$$

So in this case our "arbitrary constant" was not all that arbitrary: it represented an *initial condition* of the system (another piece of jargon in differential equations).

This often happens: a differential equation representing a dynamic situation will have an *initial condition*, and that initial condition can be found in those mathematical arbitrary constants. They are arbitrary to the mathematician, but to the practicing scientist they reflect the state of a natural system at the starting moment.

And you might have noticed that our *first* order differential equation for exponential growth had *one* arbitrary constant. Second order equations will have *two* arbitrary constants, and so forth.

16.5 Solving differential equations *numerically:* Euler's method

At this time we will click here and follow that chapter to learn about solving differential equations numerically.

16.6 Some types of equations that we would like to solve

16.6.1 Ecology: the Lotka-Volterra equations.

16.6.2 Physics: free fall with air drag.

16.6.3 Ecology and economics: the logistic equation.

16.6.4 Physics: the non-linear pendulum

16.7 Scipy and the Runge-Kutta method

CHAPTER

SEVENTEEN

APPROXIMATING AREAS AND INTEGRALS

17.1 Motivation and plan

The “area under a curve” is the general 2-dimensional area problem.

We know how to calculate the area of disks, spherical surfaces, cilinders, trapezoides, and many other shapes. But the problems that come up in real work do not have exact solutions.

CHAPTER

EIGHTEEN

MONTE CARLO METHODS

Monte Carlo methods are calculation and modeling techniques in which we use random number sequences to approximate calculations that would be much too difficult otherwise. The name is inspired by the city of Monte Carlo which has a famous casino, which relates to randomness.

There are many areas in which monte carlo methods are the best choice, and we will examine a couple of these.

18.1 Areas, volumes, hypervolumes

18.1.1 Two-dimensions: calculating π: with the monte carlo method

This is a good first introduction to *monte carlo integration*, which allows us to discuss monte carlo methods in general.

The method involves shooting darts into a square which has a circle inscribed in it. Draw the picture of a circle inside a square, and draw points of random darts hitting it.

The fraction of darts that fall in the circle is proportional to the fraction of areas:

$$\frac{N_{\text{cir}}}{N_{\text{sq}}} \approx \frac{A_{\text{cir}}}{A_{\text{sq}}}$$

The area of the circle is πr^2, and that of the square is πl^2. Since we have constructed this so that $l = 2r$ we get:

$$\frac{N_{\text{cir}}}{N_{\text{sq}}} \approx \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4}$$

This gives us:

$$\pi = 4\frac{N_{\text{cir}}}{N_{\text{sq}}}$$

Now what I usually do is write a live program which has a loop that throws 1000 darts. It does so by calculating `x = random.random() * 2 - 1` and the same for y. This gives us a dart in a square. Then using the pythagoras theorm with `if sqrt(x*x + y*y) < 1` we can determine if the dart is in the circle. We add all that up and estimate π.

Here is a `download link to the program`. You can then run it with:

```
chmod +x pi_montecarlo.dat
./pi_montecarlo.py > pi_montecarlo.dat
gnuplot
# then at the gnuplot> prompt you can type:
set grid
plot 'pi_montecarlo.dat' using 1:4 with lines
```

(continues on next page)

(continued from previous page)

```
# in a separate terminal window you can run:
gnuplot
set terminal qt lw 6
set size square
# then at the gnuplot> prompt you can type:
plot 'pi_montecarlo.dat' using 2:3 with points
replot [-1:1] sqrt(1 - x**2)
replot [-1:1] -sqrt(1 - x**2)
```

I usually write this program (12 lines) live while the students write it with me. I write the program so that it prints, for each dart, four things: the index of the loop, the x coordinate, the y coordinate, and the estimate of π so far.

After experimenting with 1000 darts, then 100000, then a million, we go back to 1000 and redirect the output into a file.

This file can be plotted with a line using columns 1 and 4 (estimate of π vs. n_darts), and with points using columns 2 and 3 (the locations of the darts).

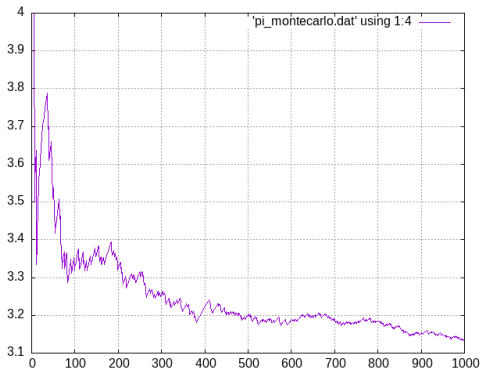

Figure 18.1.1: How the value of π improves with more darts used.

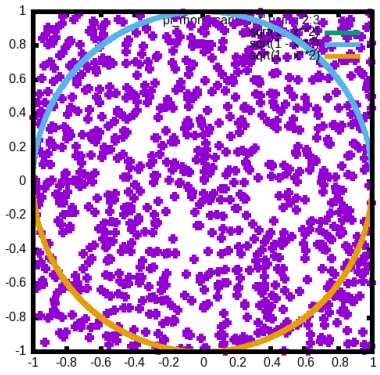

Figure 18.1.2: The places in which the darts landed.

To experiment further you can see what happens when you have as few as 100 darts, or as many as a million.

There are insights to be gained from both of these plots. The first plot shows you that the approximation to π is a painfully slow one. This is a general downside of monte carlo techniques: they are slow! You need huge numbers of darts before you start getting reasonable values for pi. Try adding many more darts to see how long it takes to get several digits of pi correctly.

The other insight comes from looking at the plot that shows the dart positions within the square. Notice how random numbers are not distributed evenly – that would be the opposite of random, since they would be more predicatble. The random generation of x and y gives us clusters and voides. This is why you need a lot of random numbers to calculate this area accurately.

18.1.2 Three dimensions: calculating the volume of a ball

The ideas from the 2-dimensional square-with-circle apply, but this time the program also generates a z coordinate, and we use the formula

$$V = \frac{4}{3}\pi r^3$$

Here is a `download link to the program`. You can then run it with:

```
chmod +x pi_montecarlo.dat
./ball_montecarlo.py > ball_montecarlo.dat
gnuplot
# then at the gnuplot> prompt you can type:
set grid
plot 'ball_montecarlo.dat' using 1:5 with lines
replot 'pi_montecarlo.dat' using 1:4 with lines

# in a separate terminal window you can run:
gnuplot
set view equal xyz
# then at the gnuplot> prompt you can type:
set samples 10000
set pm3d
splot [-1.1:1.1] [-1.1:1.1] [-1.1:1.1] 'ball_montecarlo.dat' using 2:3:4 with points pt␣
↪6 ps 0.5
replot [-1:1] sqrt(1 - x**2 - y**2) with pm3d
replot [-1:1] -sqrt(1 - x**2 - y**2) with pm3d
```

In these two examples (the disc and the ball) the analytic solution to the integral is not too difficult: one uses

CHAPTER

NINETEEN

FINDING ROOTS OF FUNCTIONS

test

19.1 Motivation and plan

19.2 Newton's method

19.3 Other methods

CHAPTER

TWENTY

RESOURCES AND FURTHER READING

20.1 Applications of Taylor series:

https://sites.math.washington.edu/~aloveles/Math126Fall2018/m126TaylorApplicationsWorksheet.pdf

https://blog.cupcakephysics.com/relativity/2015/06/14/the-low-speed-limit-of-the-lorentz-factor.html

20.2 Differential equations

Logistic differential equation:

https://en.wikipedia.org/wiki/Logistic_function#Logistic_differential_equation

CHAPTER

TWENTYONE

APPENDIX: HOW TO BUILD THIS BOOK

21.1 Prerequisites for building the book

You will also want some system and python packages for sphinx:

```
sudo apt install python3-sphinx python3-sphinx-rtd-theme
pip3 install --user --upgrade sphinx-markdown-tables
pip3 install --user --upgrade sphinx-multitoc-numbering
pip3 install --user --upgrade sphinxcontrib-svg2pdfconverter
pip3 install --user --upgrade sphinxcontrib-bibtex
```

21.2 Cloning from codeberg.org

At https://codeberg.org/markgalassi/math-science-working-groups you find git clone instructions:

```
git clone https://codeberg.org/markgalassi/math-science-working-groups.git
```

Note: If you have an account on codeberg.org you can use the "ssh URL" to clone the repo:

```
git clone git@codeberg.org:markgalassi/math-science-working-groups.git
```

21.3 Building the book

```
cd math-science-working-group
make html
```

CHAPTER

TWENTYTWO

GLOSSARY TERMS

Monomial

Monomials An expression that multplies and divides numbers and letters with no + or - signs. Examples: x^2, ab, $3ay$, $2.2x/(aby)$.

Binomial

Binomials A sum or subtraction of two *monomials*. Examples: $x^2 - x$, $x^2 - x$, $x^2 - 4$, $1 + a$, $a + b$, $a^2 - b^2$, $b + x^7$, $c + 1$, $ax^m + bx^n$, $0.9x^3 + \pi y^2$.

Factoring Factoring of an integer is to write it as a product of smaller numbers. Factoring of a polynomial is to write it as a product of smaller polynomials.

RST reStructured Text is an easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system. It is useful for in-line program documentation (such as Python docstrings), for quickly creating simple web pages, and for standalone documents. reStructured Text is designed for extensibility for specific application domains. The reStructured Text parser is a component of Docutils.

Sphinx Sphinx is a tool that makes it easy to create intelligent and beautiful documentation. It was originally created for the Python documentation, and it has excellent facilities for the documentation of software projects in a range of languages.

CHAPTER

TWENTYTHREE

BIBLIOGRAPHY

CHAPTER

TWENTYFOUR

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[Abr21] Jay Abramson. *Algebra and Trigonometry 2e*. OpenStax, Place of publication not identified, 2021. ISBN 9781938168376. URL: https://openstax.org/details/books/algebra-and-trigonometry-2e.

[Bac77] J Backus. Musical note to frequency conversion chart. http://www.audiology.org/sites/default/files/ChasinConversionChart.pdf, 1977. Accessed: 2016-05-06.

[Lin16] Samuel J. Ling. University physics. volume 1. 2016.

[Sav14] Ivan Savov. *No bullshit guide to math and physics*. Minireference Co., 2014.

INDEX