LibreNotebook

LibreNotebook

A Giant Notebook

Enoch Yu

Version: v1.10.28
Last Updated: October 28, 2026

Copyright © 2026 Enoch Yu
This book is licensed under the CC BY-SA 4.0 License.

Contents

Preface **9**

About the Book 9

Purpose and Motivation 10

Contributing 10

About the Author 11

I Math Olympiad **13**

Introducing the Notes **19**

1 Prerequisites **21**

1.1 Definitions and Notations 21

1.2 Algebra 25

1.3 Geometry 28

1.4 Number Theory 28

2 [A] Inequalities **35**

2.1 Fundamental Inequalities 35

2.2 Advanced Inequalities 48

2.3 Techniques and Patterns 51

2.4 Practice Problems 54

3 [A] Functional Equations **57**
3.1 Foundations . . . 57
3.2 Cauchy's FE and Its Variants . . . 64
3.3 Symmetric and Cyclic Equations . . . 68
3.4 Invariance . . . 68
3.5 The CDE Method . . . 68
3.6 Additional Topics . . . 68

4 [A] Polynomials **71**
4.1 Foundations . . . 71
4.2 Roots of Unity . . . 74

5 [A] Sequence and Series **79**
5.1 Prerequisites . . . 79
5.2 Recurrence Relation . . . 82
5.3 Symmetric and Cyclic Sums . . . 96
5.4 Practice Problems . . . 96

6 [G] Euclidean Geometry **97**
6.1 Power of a Point . . . 97
6.2 Six Collinear Points . . . 107
6.3 Isogonal Conjugate . . . 112
6.4 Homothety . . . 116
6.5 Points and Circles of a Triangle . . . 120

7 [G] Projective Geometry **123**
7.1 Foundations . . . 123
7.2 Fundamental Theorems . . . 128

8 [N] Diophantine Equations **137**
8.1 Primitive Pythagorean Triples . . . 137
8.2 Infinite Descent . . . 141
8.3 Pell's Equations . . . 146
8.4 Practice Problems . . . 146

9 [N] Modular Arithmetic **147**

9.1 Primitive Root Modulo n 147

10 [C] Combinatorics **149**
10.1 Foundations 149

11 Miscellaneous Goodies **151**
11.1 p-adic Valuation & LTE Lemma 151
11.2 Proving and Problem Solving Tips 157
11.3 Practice Problems 159

12 Solutions **161**
12.1 Solutions for Note 1 161

II Linear Algebra **163**

Introducing the Notes **167**

13 Vectors and Vector Spaces **169**
13.1 Basic Terms and Notations 169
13.2 Fundamental Properties 173
13.3 Subspaces 180
13.4 Practice Problems 197

14 Matrices **199**
14.1 Basic Terms and Notations 199
14.2 Fundamental Properties 202
14.3 Matrix Multiplication 206
14.4 Additional Fundamental Topics 216
14.5 Determinants 227
14.6 Practice Problems 238

15 System of Linear Equations **241**
15.1 Basics and Intuitions 241
15.2 Gaussian Elimination 246
15.3 The Multiplicative Inverse 251
15.4 LU Decomposition 262

15.5 Practice Problems 268

16 Matrices and Vector Spaces **271**
16.1 Row Space and Rank of a Matrix 271
16.2 Linear Transformations 282
16.3 Eigenvalues and Eigenvectors 304
16.4 Practice Problems 319

17 Solutions **321**
17.1 Solutions for Note 1 321
17.2 Solutions for Note 2 323
17.3 Solutions for Note 3 326
17.4 Solutions for Note 4 329
17.5 Final Thoughts 332

III Multivariable Calculus **335**

Introducing the Notes **339**

18 Vectors and Geometry **341**
18.1 Basic Terms and Intuitions 341
18.2 Vectors 345
18.3 Geometry in 3-Space 362
18.4 Cylindrical and Spherical Coordinates 374

19 Vector-Valued Functions and Parametrization **379**
19.1 Vector-Valued Functions and Calculus 379
19.2 Parameterizations of Curves 384
19.3 Curvature 387

IV Machine Learning **395**

What is AI and Why Should We Learn It? **399**

20 Fundamental Math for Machine Learning **401**

20.1 Calculus 401
20.2 Linear Algebra 406
20.3 Probability Theory 408

21 Python Basics for Machine Learning 413
21.1 Basic Syntax 413
21.2 Modules and Libraries 419

22 Foundations and Intuitions 425
22.1 Fundamental Terms and Concepts 425
22.2 Introduction to LLMs 429
22.3 Improving Large Language Models 433

23 Deep Dive into Deep Neural Networks 437
23.1 Definitions and High Level Overview 437
23.2 Mathematical Interpretations 446
23.3 Building Vanilla NN From Scratch 452

24 Building Language Models 477
24.1 Bigram Language Model 477
24.2 Attention and Transformers 477
24.3 Building an LLM with PyTorch 478

25 AI and The Future 479
25.1 AI Safety 479

26 Further Resources 481

V Competitive Programming 485

Introducing the Notes 489

27 C++ Basics 491
27.1 Foundations 491
27.2 Conditionals and Loops 496
27.3 Functions and References 507

27.4 Arrays and Vectors 517
27.5 Practice Problems and Solutions 522

28 Foundations **525**
28.1 Time Complexity 525
28.2 Simulation and Complete Search 528
28.3 Practice Problems and Solutions 535

VI Appendix **537**

Bibliography **541**

Preface

§ About the Book

Teaching and sharing information, in my opinion, is the best way to learn. Whenever I explain the contents that I learned to others, I start to really distinguish what I know and what I do not know. Therefore when writing notes, I write it as if I am explaining it to someone, sharing my thought process and understandings.

LibreNotebook is a collection of my personal notes. To be honest, the contents are biased towards my interests, and includes topics that I am passionate about and concepts that I really wish to learn thoroughly. Having said that, writing requires time and dedication. I write most of my notes on physical notebook, and this includes the selected topics that I dedicated to type. Moreover, it is not meant to replace textbooks, courses, and dedicated resources. On the other hand, if you are also interested and learning more about the topics included in my notes, I believe LibreNotebook can serve as a great resource to see how other person understood the concept.

§Purpose and Motivation

LibreNotebook initially started with my personal notes. To be exact, it began with me studying for national math olympiad. Frankly, I was lost when studying for the national math olympiad as not many resources were available specifically for the high school division of our country's olympiad. Even if there are resources, it was either for middle school round or was commercial with high expense. Naturally, I began seeking for problems and resources from different countries. There, with open and public resources, I was able to learn and improve. Since then, I always wanted to return the blessings received from open education.

Over the 2026 summer break, I wrote notes on linear algebra and machine learning. With increasing lengths and details, I thought they could be more than just personal notes. With different handouts that I wrote for multiple purposes, I thought it would be possible to return the blessings received by doing the same of returning knowledge. LibreNotebook is the notebook that highlights understandings that were learned to be shared.

§Contributing

First and foremost, thank you for your interest and taking your time to contribute. Any contributions ranging from a typo to writing a new note is appreciated! The source files for LibreNotebook are hosted in GitHub, and you can contribute by creating an issue, submitting a pull request, or write a mail using the contact form from my website.

Although contributions are welcomed, I want to include a notice on AI usage. I know this topic is philosophical and political, but I want to say I really don't want LibreNotebook to be flooded with AI generated contents. Unlike AI assisted contents, many AI generated contents lack proper credits and quality checks. For transparency of contributors, it would be great if AI usage is disclosed for each contribution. With that,

thank you for your interest and efforts!

§About the Author

Enoch is an ardent student who retains keen interest in pure math, machine learning, and quantitative finance. His previous accomplishments includes honorable mention from Korean Mathematical Olympiad (300 in South Korea) and numerous awards from AMC. He currently dedicates his time on making YouTube videos on various math topics and Nanum Learn. More on Enoch can be found in his personal website.

Part I

Math Olympiad

Contents

Introducing the Notes **19**

1 Prerequisites **21**
1.1 Definitions and Notations 21
1.2 Algebra 25
1.3 Geometry 28
1.4 Number Theory 28

2 [A] Inequalities **35**
2.1 Fundamental Inequalities 35
2.1.1 AM-GM Inequality 36
2.1.2 Power Mean Inequality 39
2.1.3 Cauchy-Schwarz Inequality 40
2.1.4 Hölder's Inequality 43
2.1.5 Jensen's Inequality 45
2.1.6 Rearrangement Inequality 47
2.2 Advanced Inequalities 48
2.2.1 Symmetric and Cyclic Inequalities 50
2.3 Techniques and Patterns 51
2.3.1 Telescoping 52
2.4 Practice Problems 54

3 [A] Functional Equations **57**

3.1 Foundations 57
3.2 Cauchy's FE and Its Variants 64
3.2.1 Pexider Functional Equation 68
3.3 Symmetric and Cyclic Equations 68
3.4 Invariance 68
3.5 The CDE Method 68
3.6 Additional Topics 68

4 [A] Polynomials **71**
4.1 Foundations 71
4.2 Roots of Unity 74
4.2.1 Roots of Unity Filter 76

5 [A] Sequence and Series **79**
5.1 Prerequisites 79
5.1.1 Telescoping 79
5.1.2 Abel Transformation 81
5.2 Recurrence Relation 82
5.3 Symmetric and Cyclic Sums 96
5.3.1 Elementary Symmetric Polynomials 96
5.3.2 Newton's Identities 96
5.3.3 Clearing Denominators 96
5.4 Practice Problems 96

6 [G] Euclidean Geometry **97**
6.1 Power of a Point 97
6.1.1 Definitions and Theorems 97
6.1.2 Brain Teasers 105
6.2 Six Collinear Points 107
6.3 Isogonal Conjugate 112
6.4 Homothety 116
6.5 Points and Circles of a Triangle 120
6.5.1 Servoir's Theorem and Euler Line 120
6.5.2 Nine-Point Circle 122

7 [G] Projective Geometry **123**
7.1 Foundations . . . 123
7.1.1 Cross Ratio . . . 124
7.2 Fundamental Theorems . . . 128
7.2.1 Pascal's Theorem . . . 129
7.2.2 Desargues' Theorem . . . 132

8 [N] Diophantine Equations **137**
8.1 Primitive Pythagorean Triples . . . 137
8.2 Infinite Descent . . . 141
8.2.1 Vieta Jumping . . . 144
8.3 Pell's Equations . . . 146
8.4 Practice Problems . . . 146

9 [N] Modular Arithmetic **147**
9.1 Primitive Root Modulo n . . . 147

10 [C] Combinatorics **149**
10.1 Foundations . . . 149

11 Miscellaneous Goodies **151**
11.1 p-adic Valuation & LTE Lemma . . . 151
11.2 Proving and Problem Solving Tips . . . 157
11.2.1 Proving Inequalities . . . 157
11.2.2 Solving FE and Geometry Problems . . . 159
11.2.3 Proving Techniques . . . 159
11.3 Practice Problems . . . 159

12 Solutions **161**
12.1 Solutions for Note 1 . . . 161

Introducing the Notes

Math olympiads are competitive proof-based events where students gather to showcase their mathematical and critical thinking skills. Some major events include International Mathematical Olympiad (IMO), European Girls' Mathematical Olympiad (EGMO), Asian Pacific Mathematical Olympiad (APMO), and National Mathematical Olympiads. The contents in this collection of notes includes must know theorems, lemmas, tips, techniques, patterns, and their applications with problems from diverse sources.

I don't know why (maybe it was due to my naiveness), but I thought I could solve an IMO problems after winning some competitions like AMC. Like I thought "How bad could it be?" One day passed, two days, and after about a week or so, I decided to look at the solution. Of course as a student who only studied AMC 8/10 style problems, it took me only minutes to fully understand their solutions. Just kidding, I took days and weeks trying to digest the official solutions, and it actually took me more time to understand the solutions than to attempt them. This is how I first became aware of what olympiads are and also a moment where I humble myself due to the vastness of math. It is really interesting because the more I learn about math, the more I realize that I know nothing about math, and I think that is one of the reasons why I do math everyday.

After that incident, I really wanted to study more for olympiads because the rewards that you get for yourself when you solve an IMO problem is incredible when you are just starting. This is also the moment when I realized how hard it is to study olympiads. Not only the concepts become very (really, extremely) difficult, it is also very difficult to start without an experienced mentor, whose classes are beyond expensiveness in South Korea. Also, the amount of time needed to dedicate in order to be competitive is incredible.

I initially quite gave up and just casually read one or two solutions a week just because I didn't know where to start. I have been searching online for resources, doomscrolling through AOPS forums, and collecting the resources. It was around 8th to 9th grade when I wanted to do more of olympiads and started learning the very basics starting from how to write a proof for olympiads, how to type math, what are different types of proof, etc. Of course students who start at the age of 6 were way ahead of me. However, I just continued trying because it is just fun to solve problems that you don't know and experience that *AHA moment*.

Despite my experience in competition math, the transition was honestly very difficult. Therefore I started writing these notes with the things that I already knew and what I think would be best way to start. Please keep in mind that I, to, am still a learning student and in no position to "teach" olympiads. However, I am constantly updating the notes with the contents that I learned and am learning. If you are also interested in math olympiads, or if you are unsure whether to start, I really hope my notes help.

Note 1

Prerequisites

Before getting into actual contents, I wanted to briefly review some of the topics that we learn in competitions because many of the olympiad topics builds from similar intuitions. This note includes basic definitions and notations that would be used throughout the notes and some concepts that must be covered before expanding them later in the notes.

§1.1 Definitions and Notations

Let's first get started with definitions for different means.

Definition 1.1.1

Generalized mean, also known as power mean, is a family of means M_p that is generalized and defined for $a_i \geq 0$ as the following [WC26].

$$M_p := \left(\frac{\sum_{k=1}^{n} a_k^p}{n} \right)^{\frac{1}{p}}$$

For weighted means, we can do the same as we would normally do with

$w_i \geq 0$.

$$M_w^p := \left(\frac{\sum_{k=1}^{n} w_k a_k^p}{\sum_{k=1}^{n} w_k} \right)^{\frac{1}{p}}$$

Note that the definition above is for generalized mean. Meaning, depending on the values of p, we can derive some of more familiar means.

M_p	Definition
$M_{-\infty}$	Minimum
M_{-1}	Harmonic Mean
M_0	Geometric Mean
M_1	Arithmetic Mean
M_2	Quadratic Mean
M_{∞}	Maximum

First, let's substitute $p = 1$ to begin with arithmetic mean.

$$M_1 = \left(\frac{\sum_{k=1}^{n} a_k}{n} \right)$$

This is **arithmetic mean**, the mean that we are all familiar with! Using the weighted generalized mean, we can get the formula for weighted arithmetic mean.

$$M_w^1 = \left(\frac{\sum_{k=1}^{n} w_k a_k}{\sum_{k=1}^{n} w_k} \right)$$

Similarly, we can obtain **quadratic mean** by substituting $p = 2$.

$$M_2 = \sqrt{\frac{\sum_{k=1}^{n} a_k^2}{n}}$$

Deriving geometric mean is quite different because $p = 0$ gives us $\frac{1}{0}$ in the exponent, which is undefined. Therefore by convention, we define **geometric mean** with $M_0(a_1, \ldots, a_n) \equiv \lim_{p \to 0} M_p(a_1, \ldots, a_n)$. Substituting, we obtain the following equations.

$$M_0 = \lim_{p \to 0} \left(\frac{\sum_{k=1}^{n} a_k^p}{n} \right)^{\frac{1}{p}}$$

$$\ln(M_0) = \ln\left(\lim_{p\to 0}\left(\frac{\sum_{k=1}^{n} a_k^p}{n}\right)^{\frac{1}{p}}\right) = \lim_{p\to 0}\frac{\ln\left(\frac{1}{n}\sum_{k=1}^{n} a_k^p\right)}{p}$$

Therefore using L'Hôpital's rule,

$$\ln(M_0) = \lim_{p\to 0}\frac{\frac{1}{n}\sum_{k=1}^{n} a_k^p \ln(a_k)}{\frac{1}{n}\sum_{k=1}^{n} a_k^p} = \frac{1}{n}\sum_{k=1}^{n}\ln(a_k) = \ln\left(\prod_{k=1}^{n} a_k\right)^{\frac{1}{n}}$$

is obtained. Rewriting the equation, the familiar version of the geometric mean formula is obtained.

$$M_0 = \left(\prod_{k=1}^{n} a_k\right)^{\frac{1}{n}} = \prod_{k=1}^{n} a_k^{\frac{1}{n}} = \sqrt[n]{a_1 a_2 \cdots a_{n-1} a_n}$$

Continuing, we can obtain **harmonic mean** with $p = -1$.

$$M_{-1} = \left(\frac{\sum_{k=1}^{n} a_k^{-1}}{n}\right)^{\frac{1}{-1}} = \frac{n}{\sum_{k=1}^{n}\frac{1}{a_k}}$$

Lastly for unbounded p, we define $M_{-\infty}$ and M_{∞} as the following by convention.

$$M_{-\infty}(a_1,\ldots,a_n) \equiv \lim_{p\to-\infty} M_p(a_1,\ldots,a_n) = \min(a_1,\ldots,a_n)$$
$$M_{\infty}(a_1,\ldots,a_n) \equiv \lim_{p\to\infty} M_p(a_1,\ldots,a_n) = \max(a_1,\ldots,a_n)$$

Continuing, here is another very important definition for sequences.

Definition 1.1.2

Let $a = (a_1,\ldots,a_n)$ and $b = (b_1,\ldots,b_n)$ be sequences such that $a_1 \geq \cdots \geq a_n$ and $b_1 \geq \cdots \geq b_n$. Sequence a **majorizes** b, denoted as $a \succ b$, if $a_1 + \cdots + a_n = b_1 + \cdots + b_n$ and $\sum_{i=1}^{m} a_i \geq \sum_{i=1}^{m} b_i$ holds for each $m = 1,\ldots,n$.

We will see more on majorization when we discuss inequalities. Continuing, let's discuss some common notations.

Below are very common sum notations that will be used throughout the notes. They are very common in olympiads, and is worth noting even if you are not seriously studying for olympiads.

Definition 1.1.3

For a function $f(x_1, x_2, \ldots, x_n)$, $\sum_{\text{cyc}}$ is defined as the following.

$$\sum_{\text{cyc}} f(x_1, x_2, \ldots, x_n) := f(x_1, x_2, \ldots, x_n) + f(x_2, x_3, \ldots, x_n, x_1) + \cdots + f(x_n, x_1, x_2, \ldots, x_{n-1})$$

You will most likely see this notation with three variables, and we can also specify with variables we are going to cycle.

$$\sum_{x,y,z} \frac{xw}{yz} = \frac{xw}{yz} + \frac{yw}{zx} + \frac{zw}{xy}$$

As we have for cyclic sum, we also have notation for symmetric sums.

Definition 1.1.4

Let $f(x_1, x_2, \ldots, x_n)$ be a function and S_n be the symmetric group of degree n, i.e. a set of all sets that are permutation of numbers 1 through n. Then, $\sum_{\text{sym}}$ is defined as the following.

$$\sum_{\text{sym}} f(x_1, x_2, \ldots, x_n) := \sum_{\sigma \in S_n} f(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)})$$

Notice that permutation of variables does not affect the value for a symmetric function $f(s_1, s_2, \ldots, s_n)$. Therefore,

$$\sum_{\text{sym}} f(s_1, s_2, \ldots, s_n) = n! f(s_1, s_2, \ldots, s_n)$$

holds for such functions.

These are the introduction to basic definition and notations! For our next section, let's review some of the topics that we learn from competitions.

§1.2 Algebra

Continuing from the review on definitions and notations, let's discuss some foundational knowledge in algebra that we learn from competition math. I intend to keep this section short because many of the topics reappear in olympiad style problems, and I think we can discuss the generalization of the topics in latter notes. In this section particular, let's discuss polynoimal expansion.

Before discussing binomial theorem, here is one notation to keep in mind. I think many of us initially learn the notation for combination as ${}_nC_k$. However in our notes, let's use the standard notation $\binom{n}{k}$ as they are used for olympiad and higher level math. Below is an important theorem to keep in mind for our future proof.

Theorem 1.2.1: Pascal's Identity

For positive integers $n > k$, the following equations hold.

$$\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$$

There are few ways to prove this identity with easiest being drawing Pascal's triangle in my opinion. However besides the visual proof, here are two very famous proofs. In our first proof, we will be approaching this problem with combinatoric ideas.

Proof.

Assume that we are selecting k leaders from n people. You being one of the people, there are two possibilities. In the first case, you are the leader and the number of cases of choosing the rest of the leaders is $\binom{n-1}{k-1}$. In the second case, you are not a leader and there are $\binom{n-1}{k}$ number of cases of choosing the remaining leaders. Therefore, $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.

This was the first proof using combinatoric idea, and here is a proof with

a more algebraic approach.

Proof.

Consider the following equations.

$$\begin{aligned}\binom{n}{k} &= \frac{n!}{k!(n-k)!} = \frac{(n-1)!(n-k)+(n-1)!k}{k!(n-k)!} \\ &= \frac{(n-1)!}{k!(n-k-1)!} + \frac{(n-1)!}{(k-1)!(n-k)!} = \binom{n-1}{k} + \binom{n-1}{k-1}\end{aligned}$$

Therefore, the identity holds.

Personally, I prefer the first proof than the second one because it feels more intuitive and second proof feels like a fallback just in case I cannot think of combinatoric approach to an algebra problem, but I think it depends on you!

Okay, with this in mind, let's discuss Binomial Theorem. I wanted to discuss this because for some reason, it quite appears a lot in unexpected problems and places like inequality and algebraic combinatorics.

Theorem 1.2.2: Binomial Theorem

For nonnegative integer n, the expansion of the term $(x+y)^n$ can be represented as the following.

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

Again there are few different ways to prove this, and let's prove this using two different ways.

Proof.

Consider the term $x^k y^{n-k}$ for some appropriate integer k. Notice that the number of such combination equals number of different ways of choosing k number of $(x+y)$ in $(x+y)^n$. Therefore for all $x^k y^{n-k}$, their coefficient is $\binom{n}{k}$. In other words, $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$.

Continuing with this combinatoric approach, here is another one using induction.

Proof.

First, notice that $x + y = \binom{1}{0}x + \binom{1}{1}y$. Therefore, it suffices to show that the equation holds for $n+1$ assuming that is holds for n. Consider the following equations.

$$\begin{aligned}(x+y)^n(x+y) &= \left(\sum_{k=0}^{n}\binom{n}{k}x^k y^{n-k}\right)(x+y) \\ &= \sum_{k=0}^{n}\left(\binom{n}{k}x^{k+1}y^{n-k} + \binom{n}{k}x^k y^{n-k+1}\right)\end{aligned}$$

Rewriting the terms, the following equation holds by Pascal's Identity.

$$\begin{aligned}(x+y)^n(x+y) = &\binom{n}{0}xy^n + \binom{n}{0}y^{n+1} \\ &+ \binom{n}{1}x^2y^{n-1} + \binom{n}{1}xy^n \\ &+ \binom{n}{2}x^3y^{n-2} + \binom{n}{2}x^2y^{n-1} \\ &\vdots \\ = &\binom{n}{n}x^{n+1} + \binom{n}{n}x^n y \\ = &\binom{n}{0}y^{n+1} + \binom{n}{n}x^{n+1} + \sum_{k=1}^{n}\binom{n+1}{k}x^k y^{n+1-k} \\ = &\sum_{k=0}^{n+1}\binom{n+1}{k}x^k y^{n+1-k}\end{aligned}$$

Therefore, the theorem holds.

For this theorem too, I personally prefer the first method because I think it is more intuitive and the second method of using induction can works as a fallback. However, both proofs are both great, and I think the choice depends on you!

This was a very short review of algebra that we learn from competition math. Unlike geometry and number theory that requires strong background understandings of topics from competition math, I think algebra is one of those fields that are very different when it comes to olympiad-style proof-based problems because there are so many new topics that were not covered in competition math. Although this section was short, we will discuss much in-depth in future notes! For now, let's review some key properties from geometry.

§1.3 Geometry

Theorem 1.3.1: Menelaus' Theorem

Assume for $\triangle ABC$, X, Y, Z are defined as $X \in \overline{AB}$, $Y \in \overline{BC}$, and $Z \in \overline{CA}$. Then, the following equation of cross-ratio holds.

$$\frac{XA}{XB} \cdot \frac{YB}{YC} \cdot \frac{ZC}{ZA} = -1$$

§1.4 Number Theory

There are quite a few important that I wish to discuss before we do Diophantine equations. I don't think some of the contents come out in the actual exam as many are focused on Diophantine equations, but let's go over them anyways as a review.

Definition 1.4.1

A **residue class** of a mod n is the collection of all numbers that are congruent to $a \pmod{n}$. The **complete residue system** mod n is the set of exactly one representative from each of the n residue classes. The **reduced residue system** is a set of all elements from the complete residue system that only contains the elements that are relatively prime to n.

An example for modulo 6 is that the complete residue system is $\{0, 1, 2, 3, 4, 5\}$ while the reduced residue system is $\{1, 5\}$. The reduced residue system for modulo 6 does not have to be $\{1, 5\}$, but $\{7, -1\}$ too.

With this definition, we can also define a very special and important function.

Definition 1.4.2: Euler's Totient Function

Euler's Totient Function, denoted as $\phi(n)$ is the number of elements in the reduced residue system mod n.

By definition, there are few ways to think about this function.

> $\phi(n)$ can be interpreted as the number of integers from 1 to n that are relatively prime to n.

As you can see from our previous example of residue systems, we can see that $\phi(6) = 2$. Another very important topic about Euler's Totient function is its formula. Consider the following equation for prime p.

$$\phi\left(p^{\alpha}\right) = p^{\alpha} - \left\lfloor \frac{p^{\alpha}}{p} \right\rfloor = p^{\alpha} - p^{\alpha-1} = p^{\alpha}\left(1 - \frac{1}{p}\right)$$

This holds because the floor function, or Gauss function, represents the number of elements that are multiples of p and p is relatively prime to other numbers that are not multiple of p. Before deriving the formula, let's briefly take a look at its multiplicative property.

> $\phi(ab) = \phi(a)\phi(b)$ for $(a, b) = 1$.

There are few ways to prove this property, but let's prove with the

following arrangement.

$$\begin{array}{ccccc} 1 & 2 & 3 & \cdots & b \\ b+1 & b+2 & b+3 & \cdots & 2b \\ 2b+1 & 2b+2 & 2b+3 & \cdots & 3b \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \cdots & \cdots & \cdots & \cdots & ab \end{array}$$

As you can see, $\phi(b)$ is the number of integers that are relatively prime to b in the first row. In other words, there exists exactly $\phi(b)$ columns that contain the integers that are relatively prime to b. Let j be a relatively prime integer to b. Because $(a, b) = 1$, it is self-evident that there exists $\phi(a)$ numbers that are relatively prime to a in j^{th} column. Because there exists $\phi(b)$ such columns, the total number of integers that are relatively prime to both a and b, or relatively prime to ab, is $\phi(a)\phi(b)$ and $\phi(ab) = \phi(a)\phi(b)$ holds for $(a, b) = 1$.

Continuing with this property, we can see that the same applies if we prime factorize n as $n = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_k^{\alpha_k}$ for prime k. In other words, we can write the formula as the following.

$$\phi(n) = p_1^{\alpha_1}\left(1 - \frac{1}{p_1}\right) \cdots p_k^{\alpha_k}\left(1 - \frac{1}{p_k}\right) = n\prod_{i=1}^{k}\left(1 - \frac{1}{p_i}\right)$$

There is also very important property

$$\sum_{d|n} \phi(d) = n$$

but maybe we can discuss this later.

Continuing, we can define another very important term.

Definition 1.4.3

For integers n and a such that $(a, n) = 1$, the **order of** a **mod** n, denoted as $\text{ord}_n a$ is the least positive integer k such that $a^k \equiv 1 \pmod{n}$.

Some examples of this would include $\text{ord}_n 1 = 1$ and $\text{ord}_n(-1) = 2$ for all $n > 3$. We will discuss more on primitive roots and the properties of the order of a mod n later in the notes. Continuing with the totient function, we can derive two important theorems.

Theorem 1.4.4: Euler's Theorem

$a^{\phi(n)} \equiv 1 \pmod{n}$ if $(a, n) = 1$.

Proof.

Let $\{r_1, r_2, \ldots, r_{\phi(n)}\}$ be the set of integers that are relatively prime to n. Note that for positive integer a such that $(a, n) = 1$, $\{ar_1, ar_2, \ldots, ar_{\phi(n)}\}$ is the same set since $ar_i \not\equiv ar_j \pmod{n}$ for $i \neq j$.

Therefore,

$$r_1 r_2 \ldots r_{\phi(n)} \equiv ar_1 ar_2 \ldots ar_{\phi(n)} \pmod{n}$$

holds and $a^{\phi(n)} \equiv 1 \pmod{n}$.

The next theorem Fermat's Little Theorem can be proven with multiple ways, but let's use Euler's Theorem to prove it. They are frequently introduced together and some say Fermat's Little Theorem is a corollary of Euler's Theorem, but let's consider it as a standalone theorem because there are plenty of other ways to prove the theorem.

Theorem 1.4.5: Fermat's Little Theorem

For prime p and positive integer a such that $(a, p) = 1$, $a^{p-1} \equiv 1 \pmod{p}$.

Proof.

By Euler's Theorem, $a^{\phi(p)} \equiv 1 \pmod{p}$. Because $\phi(p) = p - 1$, the theorem holds.

Another very common way of representing this theorem is $a^p \equiv a \pmod{p}$. Before we move on to the final theorem that we will discuss in this section, let's take a look at an example of how they are used.

Exercise 1.4.6

Show that $8^{43} \equiv 2 \pmod{10}$.

Solution.

We can use CRT for this, but let's use Fermat's Little Theorem here. How can we use it if $(8, 10) \neq 1$? Well, we can force it. Consider the following congruences with $8^{43} \equiv 0 \pmod{2}$ in mind.

$$\begin{aligned} 8^4 &\equiv 1 \pmod{5} \\ 8^{43} \equiv 8^3 \equiv 2^9 &\equiv 2 \pmod{5} \end{aligned}$$

In other words, $8^{43} = 5m + 2 = 2n$ for some integers m and n. Because $2 \mid m$, $8^{43} \equiv 2 \pmod{10}$.

Like this, we can split the modulo n for our appropriate need. Before moving on to the final one and concepts related to the theorem, here is another example.

Exercise 1.4.7

Show that $2009^{2008^{2007}} \equiv 81 \pmod{1000}$ [Win26].

Solution.

First, notice that $(2009, 1000) = 1$ and $\phi(1000) = 400$. Therefore, $2009^{400} \equiv 1 \pmod{1000}$ and the following congruence holds for

some n and k.

$$2009^{2008^{2007}} \equiv 2009^{400n+k} \equiv 2009^k \equiv 9^k \pmod{1000}$$

Continuing, note that $2008^{2007} \equiv k \pmod{400}$. Consider the following congruences.

$$2008^{2007} \equiv 2008^7 \equiv 8^7 \equiv 2^{21} \equiv 2 \pmod{25}$$
$$2008^{2007} \equiv 0 \pmod{16}$$

Therefore for some integer p and q, $2008^{2007} = 25p + 2 = 16q$. In other words, $2008^{2007} = 25(16p' + 14) + 2 = 400p' + 352$ for some integer p'. Therefore $k = 352$ and it suffices to find 9^{352} modulo 1000.

Continuing, $2009^{2008^{2007}} \equiv 9^{352} \equiv 3^{704} \pmod{1000}$. Therefore, we can consider the following congruences with $3^4 \equiv 1 \pmod{8}$ and $3^{100} \equiv 1 \pmod{125}$.

$$3^{704} \equiv 81 \pmod{125}$$
$$3^{704} \equiv 1 \pmod{8}$$

Let $3^{704} = 125r + 81 = 8s + 1$ for some integers r and s. Therefore, $3^{704} = 1000r' + 81$ for $r' \in \mathbb{N}$ and $2009^{2008^{2007}} \equiv 81 \pmod{1000}$.

With these techniques in mind, let's conclude our review on number theory with a theorem and modular multiplicative inverse.

Definition 1.4.8

For two integers a and n such that $ax \equiv 1 \pmod{n}$ and $(a, n) = 1$, x often denoted as a^{-1} is the **modular multiplicative inverse** of a.

With this definition, we can see an interesting property. If we consider the equation $ax + nm = 1$, we can know that there always exists integer x, m that satisfies the equation by **Bézout's Identity**. In other words, integer x such that $ax \equiv 1 \pmod{n}$ always exist if $(a, n) = 1$. Here, we could

also see that if $x_1 \neq x_2$ are both modular multiplicative inverse of a, then $ax_1 \equiv ax_2 \pmod{n}$ and $x_1 \equiv x_2 \pmod{n}$, implying the uniqueness of the modular multiplicative inverse of a in the standard representatives.

> For an integer a and modulo n, if $(a, n) = 1$, then there always exists the modular multiplicative inverse of a and the inverse is unique in the residue system of n.

With this, we can consider Wilson's Theorem.

Theorem 1.4.9: Wilson's Theorem

For an integer $p > 1$, $(n-1)! \equiv -1 \pmod{p}$ if and only if p is a prime.

Proof.

To prove this theorem, first consider the case when an integer is a modular multiplicative inverse of itself.

$$x \equiv x^{-1} \pmod{p}$$

Note that by definition, $x^2 \equiv 1 \pmod{p}$ holds if $x \equiv 1 \pmod{p}$ or $x \equiv p-1 \pmod{p}$. Moreover, there exists even number of integers $1, \ldots, p-1$ for prime $p > 2$. Because those numbers are prime to p, there exists a unique modular multiplicative inverse, and this implies that the terms can be grouped in pairs, leaving only $(p-1)! \equiv p-1 \equiv -1 \pmod{p}$. Moreover, it is evident that the theorem holds for $p = 2$. Furthermore, because the pairing does not apply when p is not a prime, the theorem holds.

This was it for the review on number theory! In our next section, we will review some of the basic topics in combinatorics.

Note 2

[A] Inequalities

Inequality is perhaps the most amiable, but extremely in depth section to study in Algebra. It could range from simple inequalities such as solving for x in $x + 1 > 2$ to problems involving numerous advanced techniques and known results. As a result, although few inequalities in advanced problems could be proven by bashing, most would require an elegant solution. With that in mind, this note will begin with numerous classical inequalities such as the AM-GM inequality to advanced concepts such as majorization.

§2.1 Fundamental Inequalities

One of the first fields that I studied for olypmiads is inequality, and I think that is the same for many other students. Indeed, naive me when I first started inequalities didn't know that inequalities is one of the fields that really requires you to have lots of background knowledge because many problems are solved by tweaking known inequalities and doing some clever substitutions. In this section, let's discuss some fundamental must-know inequalities as an introduction to olympiad inequalities.

2.1.1 AM-GM Inequality

I assume most of you know or at least have heard of this inequality as we learn it in middle and high school. AM-GM inequality stands for Arithmetic Mean-Geometric Mean inequality and literally compares those two means.

Theorem 2.1.1: AM-GM Inequality

For $a_k \geq 0$, the following inequality holds with equality if and only if $a_1 = a_2 = \cdots = a_n$.

$$\frac{\sum_{i=1}^{n} a_i}{n} \geq \sqrt[n]{\prod_{i=1}^{n} a_i}$$

Proof.

Given a sequence $a_1, \ldots, a_n$, let A and G be the arithmetic and geometric means respectively. First, consider the function $y = e^x$. Because $y' = e^x$ and $y'' = e^x > 0$, it is evident that $x + 1$ is the tangent line of $y = e^x$ at $x = 0$ and $e^x \geq x + 1$. Substituting $x = \frac{a_i}{G} - 1$ and multiplying all inequalities gives the following.

$$e^{\frac{\sum_{i=1}^{n} a_i}{G} - n} \geq \frac{\prod_{i=1}^{n} a_i}{G^n}$$

Therefore, $e^{\frac{An}{G} - n} \geq \frac{G^n}{G^n}$ and $\frac{An}{G} \geq n$. Therefore, $A \geq G$. Moreover, the equality holds if the terms are all equal to each other.

You might be more familiar with $a + b \geq 2\sqrt{ab}$, but the inequality above is the generalized version.

> In the context of olympiads, AMGM is used the most for cancellation and investigating the minimum.

Let's take a look at very simple example.

Exercise 2.1.2

Prove the following inequality for $a, b, c, d \geq 0$.

$$(a+b)(b+c)(c+d)(d+a) \geq 16abcd$$

Solution.

Because a, b, c, d are non-negative, we could utilize AM-GM inequality. By AM-GM inequality, the following inequalities hold.

$$\begin{aligned} a+b &\geq 2\sqrt{ab} \\ b+c &\geq 2\sqrt{bc} \\ c+d &\geq 2\sqrt{cd} \\ d+a &\geq 2\sqrt{da} \end{aligned}$$

Multiplying the inequalities,

$$(a+b)(b+c)(c+d)(d+a) \geq 2\sqrt{ab} \cdot 2\sqrt{bc} \cdot 2\sqrt{cd} \cdot 2\sqrt{da} = 16abcd$$

hold and $(a+b)(b+c)(c+d)(d+a) \geq 16abcd$.

Here is another example for AMGM.

Exercise 2.1.3: 2010 KMO Problem 5 (● ○ ○)

Given positive reals x, y, z such that $x+y+z=1$, show that the following inequality holds.

$$\sqrt{\frac{x}{1-x}} + \sqrt{\frac{y}{1-y}} + \sqrt{\frac{z}{1-z}} > 2$$

Solution.

First, consider AMGM with t and $1-t$ where $0 < t < 1$. $\frac{1}{2} \geq \sqrt{t(1-t)}$ and the equality holds if and only if $t = \frac{1}{2}$. Rearranging the inequality, $\frac{1}{\sqrt{t(1-t)}} \geq 2$ and $\sqrt{\frac{t}{1-t}} \geq 2t$. Substituting x, y, z,

$\sqrt{\frac{x}{1-x}} + \sqrt{\frac{y}{1-y}} + \sqrt{\frac{z}{1-z}} \geq 2$ is obtained. However, because $x = y = z = \frac{1}{2}$ cannot hold, the inequality is strict.

When we discuss arithmetic and geometric mean, we normally include weighted means, so let's do the same for inequality.

Definition 2.1.4: Weighted AM-GM Inequality

For weights $w_1, w_2, \ldots, w_n \geq 0$ such that $\sum_{i=1}^{n} w_i = w$ the following inequality holds.

$$\frac{\sum_{i=1}^{n} w_i a_i}{w} \geq \sqrt[w]{\prod_{i=1}^{n} a_i^{w_i}}$$

The equality holds if and only if $a_1 = a_2 = \cdots = a_n$.

From the definition, we can notice that the original inequality is obtained with $w_1 = w_2 = \cdots = w_n = \frac{1}{n}$. Here is a problem on weighted AM-GM inequality.

Exercise 2.1.5: 2020 IMO Problem 2 (● ● ○)

Show that for $a, b, c, d \in \mathbb{R}$ such that $a \geq b \geq c \geq d > 0$ and $a + b + c + d = 1$,

$$(a + 2b + 3c + 4d)a^a b^b c^c d^d < 1$$

holds.

(Video Solution)

Solution.

First, consider the following weighted AM-GM inequality for a, b, c, d with weights a, b, c, d respectively.

$$\frac{a \cdot a + b \cdot b + c \cdot c + d \cdot d}{a + b + c + d} \geq \sqrt[a+b+c+d]{a^a b^b c^c d^d}$$

$$a^2 + b^2 + c^2 + d^2 \geq a^a b^b c^c d^d$$

Therefore, it suffices to show that the following inequality holds.

$$(a + 2b + 3c + 4d)(a^2 + b^2 + c^2 + d^2) < 1$$

Consider inequalities below.

$$\begin{aligned}
a^2(a + 2b + 3c + 4d) &< a^2(a + 3b + 3c + 3d) \quad (\because b \geq d) \\
b^2(a + 2b + 3c + 4d) &< b^2(3a + b + 3c + 3d) \quad (\because 2a \geq b + d) \\
c^2(a + 2b + 3c + 4d) &< c^2(3a + 3b + c + 3d) \quad (\because 2a + b \geq 2c + d) \\
d^2(a + 2b + 3c + 4d) &< d^2(3a + 3b + 3c + d) \quad (\because 2a + b \geq 3d)
\end{aligned}$$

Integrating,

$$\begin{aligned}
&(a + 2b + 3c + 4d)(a^2 + b^2 + c^2 + d^2) \\
&\qquad < a^2(a + 3b + 3c + 3d) + b^2(3a + b + 3c + 3d) \\
&\qquad\quad + c^2(3a + 3b + c + 3d) + d^2(3a + 3b + 3c + d) \\
&\qquad < (a + b + c + d)^3 = 1
\end{aligned}$$

and the inequality holds.

With the ideas of weighted AMGM in mind, we can further generalize our results for different means just as we have generalized the means.

2.1.2 Power Mean Inequality

Power Mean inequality, also known as QM-AM-GM-HM inequality, compares the four major means with generalization of AMGM. Without simple notations, we can represent the Power Mean inequality as $M_2 \geq M_1 \geq M_0 \geq M_{-1}$.

Definition 2.1.6: Power Mean Inequality

For positive real numbers a_i, the inequality

$$\sqrt{\frac{\sum_{i=1}^{n} a_i^2}{n}} \geq \frac{\sum_{i=1}^{n} a_i}{n} \geq \sqrt[n]{\prod_{i=1}^{n} a_i} \geq \frac{n}{\sum_{i=1}^{n} \frac{1}{a_i}}$$

holds with equality if and only if $a_1 = \cdots = a_n$.

With the definition in mind, we can naturally apply the same definition for weighted power means.

Definition 2.1.7: Weighted Power Mean Inequality

For positive real numbers a_i, the inequality

$$\sqrt{\frac{\sum_{i=1}^{n} w_i a_i^2}{w}} \geq \frac{\sum_{i=1}^{n} w_i a_i}{w} \geq \sqrt[w]{\prod_{i=1}^{n} a_i^{w_i}} \geq \frac{w}{\sum_{i=1}^{n} \frac{w_i}{a_i}}$$

holds with equality if and only if $a_1 = \cdots = a_n$.

Continuing from AMGM and its variations, let's discuss arguably the second most famous inequality in middle and high schools: Cauchy-Schwarz Inequality.

2.1.3 Cauchy-Schwarz Inequality

As always, let's get started with the definition.

Definition 2.1.8: Cauchy-Schwarz Inequality

For all real numbers a_i and b_i,

$$\left(\sum_{i=1}^{n} a_i^2\right)\left(\sum_{i=1}^{n} b_i^2\right) \geq \left(\sum_{i=1}^{n} a_i b_i\right)^2$$

with equality if and only if $\frac{a_i}{b_i} = k$ for all integer $1 \leq i \leq n$, where $k > 0$ and $a_i b_i \neq 0$.

When $n = 2$ we get the form $(a^2 + b^2)(x^2 + y^2) \geq (ax + by)^2$ that we initially learn.

Now there are few standard proofs for Cauchy inequality, and I personally learned the method using determinants, but there is another very interesting way that uses AMGM and normalization, and in my opinion much better than the one that uses determinants. Here is a video explaining two different proofs if you prefer videos instead of text below.

Proof.

Let $\sum_{i=1}^{n} a_i^2 = A^2$ and $\sum_{i=1}^{n} b_i^2 = B^2$. Then, the terms can be rewritten as $a_i = Ax_i$ and $b_i = By_i$. Naturally by construction, it is evident that $\sum_{i=1}^{n} x_i^2 = \sum_{i=1}^{n} y_i^2 = 1$. Rewriting the Cauchy inequality, we obtain the following.

$$A^2B^2 \geq A^2B^2\left(\sum_{i=1}^{n} x_i y_i\right)$$

In other words, it suffices to show that $\sum_{i=1}^{n} x_i y_i \leq 1$ for positive x_i and y_i. Using AMGM,

$$\sum_{i=1}^{n} x_i y_i \leq \sum_{i=1}^{n}\left(\frac{x_i^2 + y_i^2}{2}\right)$$

holds and $\sum_{i=1}^{n} x_i y_i \leq \frac{2}{2} = 1$. Moreover, the equality holds if and only if $x_i = y_i$, i.e. $\frac{a_i}{A} = \frac{b_i}{B}$ for all integer $i \in [1, n]$ and $\frac{a_i}{b_i}$ is constant.

I like this proof because we can expand Cauchy inequality as the following.

$$\left(\sum_{i=1}^{n} a_i^3\right)\left(\sum_{i=1}^{n} b_i^3\right)\left(\sum_{i=1}^{n} c_i^3\right) \geq \left(\sum_{i=1}^{n} a_i b_i c_i\right)^3$$

We cannot use determinants in this case. However letting $\sum_{i=1}^{n} a_i^3 = A^3$, $\sum_{i=1}^{n} b_i^3 = B^3$, and $\sum_{i=1}^{n} c_i^3 = C^3$, we can set the individual terms $a_i = Ax_i$, $b_i = By_i$, and $c_i = Cz_i$. Therefore, it suffices to show

$$A^3B^3C^3 \geq A^3B^3C^3\left(\sum_{i=1}^{n} z_i y_i z_i\right)^3$$

and that is equivalent of showing $\left(\sum_{i=1}^{n} z_i y_i z_i\right)^3 \leq 1$. Using AMGM,

$$\left(\sum_{i=1}^{n} z_i y_i z_i\right)^3 \leq \sum_{i=1}^{n}\left(\frac{x^3+y^3+z^3}{3}\right) = 1$$

holds and our new version of Cauchy inequality holds. With this method, we can continue generalizing for $n > 3$.

Continuing, we can obtain an interesting direct consequence. The lemma below known as Titu's lemma, T2 Lemma, Sedrakyan's inequality, and Engel's form and was named after one of the greatest mathematician Titu Andreescu.

Corollary 2.1.9: Titu's Lemma

For positive real numbers a_i and b_i,

$$\sum_{i=1}^{n} \frac{a_i^2}{b_i} \geq \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{\sum_{i=1}^{n} b_i}$$

Proof.

Substituting $x_i = \frac{a_i}{\sqrt{b_i}}$ and $y_i = \sqrt{b_i}$ to the Cauchy-Schwarz inequal-

ity, the following inequalities are obtained.

$$\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right) \geq \left(\sum_{i=1}^{n} x_i y_i\right)^2$$

$$\left(\sum_{i=1}^{n} \left(\frac{a_i}{\sqrt{b_i}}\right)^2\right)\left(\sum_{i=1}^{n} \left(\sqrt{b_i}\right)^2\right) \geq \left(\sum_{i=1}^{n} \frac{a_i}{\sqrt{b_i}} \cdot \sqrt{b_i}\right)^2$$

Recasting,

$$\left(\sum_{i=1}^{n} \frac{a_i^2}{b_i}\right)\left(\sum_{i=1}^{n} b_i\right) \geq \left(\sum_{i=1}^{n} a_i\right)^2$$

$$\sum_{i=1}^{n} \frac{a_i^2}{b_i} \geq \frac{\left(\sum_{i=1}^{n} a_i\right)^2}{\sum_{i=1}^{n} b_i}$$

are obtained and the lemma holds.

This lemma comes especially handy if an inequality involves fractions with square numbers. Now that we discussed a direct consequence of Cauchy-Schwarz inequality, we can consider its generalization.

2.1.4 Hölder's Inequality

Cauchy-Schwarz inequality is actually a special case of Hölder's inequality. Both inequalities can be used in multiple ways such as eliminating radicals and fractions. Below is the definition of Hölder's inequality.

Definition 2.1.10: Hölder's Inequality

Let $a_i, b_i, \ldots, z_i > 0$ and $\lambda_a, \lambda_b, \ldots, \lambda_z \geq 0$ such that $\lambda_a + \lambda_b + \cdots + \lambda_z = 1$. Then,

$$\left(\sum_{i=1}^{n} a_i\right)^{\lambda_a}\left(\sum_{i=1}^{n} b_i\right)^{\lambda_b} \cdots \left(\sum_{i=1}^{n} z_i\right)^{\lambda_z} \geq \sum_{i=1}^{n} a_i^{\lambda_a} b_i^{\lambda_b} \cdots z_i^{\lambda_z}$$

holds with equality if $a_1 : a_2 : \cdots : a_n \equiv b_1 : b_2 : \cdots : b_n \equiv \cdots \equiv z_1 : z_2 : \cdots : z_n$.

Indeed, Cauchy inequality does resemble Hölder's Inequality! To derive Cauchy inequality from Hölder's, it suffices to show that a specific case of Hölder's inequality will lead to Cauchy inequality.

Let $\lambda_a = \lambda_b = \frac{1}{2}$. Substituting, the following inequalities hold.

$$\left(\sum_{i=1}^{n} a_i\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} b_i\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} c_i\right)^{0} \cdots \left(\sum_{i=1}^{n} z_i\right)^{0} \geq \sum_{i=1}^{n} a_i^{\frac{1}{2}} b_i^{\frac{1}{2}} \cdot c_i^0 \cdots z_i^0$$

$$\left(\sum_{i=1}^{n} a_i\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} b_i\right)^{\frac{1}{2}} \geq \sum_{i=1}^{n} a_i^{\frac{1}{2}} b_i^{\frac{1}{2}}$$

Let $\alpha_i = \sqrt{a_i}$ and $\beta_i = \sqrt{b_i}$. Substituting,

$$\left(\sum_{i=1}^{n} \alpha_i^2\right)^{\frac{1}{2}} \left(\sum_{i=1}^{n} \beta_i^2\right)^{\frac{1}{2}} \geq \sum_{i=1}^{n} \alpha_i \beta_i$$

and

$$\left(\sum_{i=1}^{n} \alpha_i^2\right) \left(\sum_{i=1}^{n} \beta_i^2\right) \geq \left(\sum_{i=1}^{n} \alpha_i \beta_i\right)^2$$

are obtained. This is Cauchy-Schwarz inequality! Below is a very classic problem on Hölder's Inequality

Exercise 2.1.11

Show that the following inequality holds for all $a, b, c > 0$.

$$(a^3 + 2)(b^3 + 2)(c^3 + 2) \geq (a + b + c)^3$$

(Video Solution)

Solution.

The key to this problem is changing the left-hand side to apply Hölder's Inequality. Consider the following inequality by Hölder.

$$(a^3 + 1 + 1)^{\frac{1}{3}} (1 + b^3 + 1)^{\frac{1}{3}} (1 + 1 + c^3)^{\frac{1}{3}} \geq a + b + c$$

Thus, cubing both sides leads to the desired result.

Continuing from Cauchy-Schwarz inequality and its variation, another fundamental inequality of different type is Jensen's inequality.

2.1.5 Jensen's Inequality

The next inequality Jensen's inequality is arguably one of the most powerful inequalities in olympiad. It used to be one of the niche inequality in Korea, but I think every elementary and middle school students preparing for KMO knows this now. Anyhow, here is the statement.

Theorem 2.1.12: Jensen's Inequality

Given a convex function f and $\lambda_1, \ldots, \lambda_n \geq 0$ such that $\sum_{i=1}^{n} \lambda_i = 1$, the following inequality holds.

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \leq \sum_{i=1}^{n} \lambda_i f(x_i)$$

If f is concave, i.e. $f''(x) < 0$, then the inequality is flipped.

$$f\left(\sum_{i=1}^{n} \lambda_i x_i\right) \geq \sum_{i=1}^{n} \lambda_i f(x_i)$$

Proof.

First, notice that it suffices to show the case for convex f. The proof can be done by induction from $n = 2$. If $n = 1$, $\lambda = 1$ and the inequality holds.

First, notice that if $n = 2$, then $\lambda_1 f(x_1) + \lambda_2 f(x_2)$ represents a point that lies on the segment that passes through $f(x_1)$ and $f(x_2)$. Similarly, $f(\lambda_1 x_1 + \lambda_2 x_2)$ represents a point on the graph of f where the x coordinate is the same as the other point that lies on the segment that passes through $f(x_1)$ and $f(x_2)$. Because f is convex, the inequality holds.

Continuing, it suffices to show that the inequality holds for when $n = k+1$ if it holds for $n = k$. Consider the following equations.

$$\begin{aligned}\sum_{i=1}^{k+1} \lambda_i f(x_i) &= \sum_{i=1}^{k} \lambda_i f(x_i) + \lambda_{k+1} f(x_k + 1) \\ &= (1 - \lambda_{k+1}) \sum_{i=1}^{k} \frac{\lambda_i}{1 - \lambda_{k+1}} f(x_i) + \lambda_{k+1} f(x_k + 1) \\ &\geq (1 - \lambda_{k+1}) f\left(\sum_{i=1}^{k} \frac{\lambda_i x_i}{1 - \lambda_{k+1}} \right) + \lambda_{k+1} f(x_k + 1)\end{aligned}$$

Moreover, because the inequality holds for $n = 2$, the form above can be written as

$$(1 - \lambda_{k+1}) f\left(\sum_{i=1}^{k} \frac{\lambda_i x_i}{1 - \lambda_{k+1}} \right) + \lambda_{k+1} f(x_k + 1) \geq f\left(\sum_{i=1}^{k+1} \lambda_i x_i \right)$$

and the inequality holds.

This proof is the classic reuse the base case to show the induction! Here is an interesting connection between AMGM and Jensen.

Exercise 2.1.13

Show AMGM inequality from Jensen's inequality.

Solution.

To derive AMGM from Jensen, consider the function $f(x) = \ln x$. Notice that because $f''(x) = -x^{-2} < 0$, the following equation holds by Jensen's inequality.

$$\ln\left(\sum_{i=1}^{n} \frac{a_i}{n} \right) \geq \sum_{i=1}^{n} \frac{\ln(a_i)}{n} = \ln\left(\prod_{i=1}^{n} a_i^{\frac{1}{n}} \right)$$

In other words, $\sum_{i=1}^{n} \frac{a_i}{n} \geq \left(\prod_{i=1}^{n} a_i \right)^{\frac{1}{n}}$ and AMGM inequality holds.

2.1.6 Rearrangement Inequality

Let's start with definition.

Definition 2.1.14: Rearrangement Inequality

Let a_i and b_i be real numbers such that $a_1 \geq a_2 \geq \cdots \geq a_n$ and $b_1 \geq b_2 \geq \cdots \geq b_n$. Moreover, let $b_{\sigma(i)}$ be the i^{th} term of the permutations of $b_1, \cdots, b_n$. Then, the following inequalities hold.

$$\sum_{i=1}^{n} a_i b_i \geq \sum_{i=1}^{n} a_i b_{\sigma(i)} \geq \sum_{i=1}^{n} a_i b_{n-i+1}$$

The equality holds if at least one sequence is constant.

With the definition, we can obtain a special consequence.

Corollary 2.1.15: Chebyshev's Inequality

For real numbers $a_1 \geq a_2 \geq \cdots \geq a_n$ and $b_1 \geq b_2 \geq \cdots \geq b_n$,

$$n\left(\sum_{i=1}^{n} a_i b_i\right) \geq \left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} b_i\right) \geq n\left(\sum_{i=1}^{n} a_i b_{n-i+1}\right)$$

holds with the equality if at least one sequence is constant.

Proof.

Consider the following cases where $b_1 \geq b_2 \geq \cdots \geq b_n$ are cycled

through in the rearrangement inequality.

$$\sum_{i=1}^{n} a_i b_i \geq a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \geq \sum_{i=1}^{n} a_i b_{n-i+1}$$
$$\sum_{i=1}^{n} a_i b_i \geq a_1 b_2 + a_2 b_3 + \cdots + a_n b_1 \geq \sum_{i=1}^{n} a_i b_{n-i+1}$$
$$\vdots$$
$$\sum_{i=1}^{n} a_i b_i \geq a_1 b_n + a_2 b_1 + \cdots + a_n b_{n-1} \geq \sum_{i=1}^{n} a_i b_{n-i+1}$$

Adding the inequalities,

$$n\left(\sum_{i=1}^{n} a_i b_i\right) \geq \left(\sum_{i=1}^{n} a_i\right)\left(\sum_{i=1}^{n} b_i\right) \geq n\left(\sum_{i=1}^{n} a_i b_{n-i+1}\right)$$

and the corollary holds.

§2.2 Advanced Inequalities

Muirhead's inequality is one of the powerful proving tools that is also a generalization of AM-GM inequality.

Theorem 2.2.1: Muirhead's Inequality

Let (a_i) and (b_i) be sequences such that $(a_i) \succ (b_i)$. Then, the following inequality holds for all nonnegative x_i.

$$\sum_{\text{sym}} {x_1}^{a_1} \cdots {x_n}^{a_n} \geq \sum_{\text{sym}} {x_1}^{b_1} \cdots {x_n}^{b_n}$$

The equality holds if and only if (a_i) and (b_i) are identical or if all x_i are identical, which is the most common case.

Another very powerful inequality is Shur's inequality. Shur's inequality comes with many alternative forms, and it is generally encouraged to

prove the derivation as there are also not-so-famous ones.

Theorem 2.2.2: Schur's Inequality

For $a, b, c \geq 0$ and $r \geq 0$,

$$\sum_{\text{cyclic}} a^r(a-b)(a-c) \geq 0$$

$$a^r(a-b)(a-c) + b^r(b-a)(b-c) + c^r(c-a)(c-b) \geq 0$$

holds with equality if $a = b = c$.

Proof.

First, notice that the left hand side of the inequality is cyclic. Therefore without loss of generality, let $a \geq b \geq c \geq 0$. Rewriting the inequality,

$$\begin{aligned} a^r(a-b)(a-c) + b^r(b-a)(b-c) + c^r(c-a)(c-b) &\geq 0 \\ (a-b)\left[a^2(a-c) - b^r(b-c)\right] + c^r(c-a)(c-b) &\geq 0 \end{aligned}$$

holds. Moreover by construction, $a-b$, $a^2(a-c) - b^r(b-c)$, and $c^r(c-a)(c-b)$ are all greater than or equal to zero. Since the sum of all non-negative numbers is a non-negative number, the inequality holds.

It won't be an exaggeration to say that the real power of Schur's inequality in olympiads come from its corollaries.

Corollary 2.2.3

The following inequality holds for all reals a, b, c.

$$a^2 + b^2 + c^2 \geq ab + bc + ca$$

Proof.

Substituting $r = 0$ in Schur's inequality, the following is obtained.

$$(a-b)(a-c) + (b-a)(b-c) + (c-a)(c-b) \geq 0$$
$$\therefore a^2 + b^2 + c^2 \geq ab + bc + ca$$

However, notice that unlike the condition for a, b, c in Schur's inequality, the inequality above holds for negative a, b, c. Without loss of generality, let $a \geq b \geq c$. If $c < 0$ and $a, b \geq 0$, then it is self evident that the inequality is holds. Similarly, if two variables are negative, the inequality also holds. Lastly, when all a, b, c are all negative, then the inequality holds as positive numbers are greater than negative sum.

Minkowski's inequality is a generalization of the triangle inequality.

Theorem 2.2.4: Minkowski's Inequality

For $a_i, b_i > 0$ and $p > 1$, the following inequality is holds.

$$\left(\sum_{k=1}^{n}(a_k + b_k)^p\right)^{\frac{1}{p}} \leq \left(\sum_{k=1}^{n} a_k^p\right)^{\frac{1}{p}} + \left(\sum_{k=1}^{n} b_k^p\right)^{\frac{1}{p}}$$

The equality holds if and only if there exists $\lambda > 0$ such that $a_i = \lambda b_i$ for all $1 \leq i \leq n$.

* Bernoulli's Inequality * Equal Value Principle

2.2.1 Symmetric and Cyclic Inequalities

* Newton's Inequality * Maclaurin's Inequality * Popoviciu's Inequality

Nesbitt's inequality can be considered as the direct consequence of Cauchy inequality.

Theorem 2.2.5: Nesbitt's Inequality

For arbitrary $a, b, c > 0$, the following inequality holds.

$$\sum_{\text{cyc}} \frac{a}{b+c} \geq \frac{3}{2}$$

Proof.

Consider the following inequality by Cauchy inequality.

$$((a+b)+(b+c)+(c+a))\left(\frac{1}{a+b}+\frac{1}{b+c}+\frac{1}{c+a}\right) \geq (1+1+1)^2 = 9$$

Therefore, the following inequalities hold.

$$2\left(\frac{a+b+c}{a+b}+\frac{a+b+c}{b+c}+\frac{a+b+c}{c+a}\right)$$
$$= 2\left(\frac{c}{a+b}+\frac{a}{b+c}+\frac{b}{c+a}+3\right) \geq 9$$

Thus, the inequality holds.

Nesbitt's inequality is less frequently used than other inequalities, but when applied, it can be a very powerful inequality.

§2.3 Techniques and Patterns

One of the many fields in olympiad algebra that has known techniques and patterns that could come very handy would be inequality. This section mostly includes the techniques that can be implemented with simple pattern recognition, and shares some of my personal experience that I built from solving inequality problems.

2.3.1 Telescoping

Telescoping series and generalization powered with classic inequalities like AM-GM and Cauchy Schwarz can really come handy in solving some problems.

Definition 2.3.1

Telescoping series refers to a substantially large series that cancels out to leave only the beginning and the ending terms.

The definition above is not "formal" per se, but an example of such series would be the following.

$$\begin{aligned}\sum_{k=1}^{n}(a_k - a_{k+1}) &= a_1 - a_2 + a_2 - a_3 + a_4 - a_5 + \cdots + a_n - a_{n+1} \\ &= a_1 - a_{n+1}\end{aligned}$$

Of course telescoping series do not reveal themselves in problems easily. A common form in which they are found is fraction. For instance, the following sequence telescopes.

$$\begin{aligned}\sum_{k=1}^{n}\left(\frac{1}{k(k+1)}\right) &= \sum_{k=1}^{n}\left(\frac{1}{k} - \frac{1}{k+1}\right) \\ &= \frac{1}{1} - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \cdots + \frac{1}{n} - \frac{1}{n+1} \\ &= 1 - \frac{1}{n+1} = \frac{n}{n+1}\end{aligned}$$

Below is another classic problem with an interesting property.

Exercise 2.3.2

Show that the following inequality holds.

$$\sum_{i=1}^{n}\frac{1}{i^2} < 2$$

Proof.

Consider the identity $\frac{1}{k(k+1)} = \frac{1}{k} - \frac{1}{k+1}$ for $k \neq 0, -1$. Expanding, the following inequalities are obtained.

$$\begin{aligned}\frac{1}{1^2} + \frac{1}{2^2} + \cdots + \frac{1}{n^2} &< 1 + \frac{1}{1 \cdot 2} + \cdots + \frac{1}{(n-1) \cdot n} \\ &= 1 + \left(\frac{1}{1} - \frac{1}{2}\right) + \cdots + \left(\frac{1}{n-1} - \frac{1}{n}\right) \\ &= 2 - \frac{1}{n} < 2\end{aligned}$$

Therefore, the inequality holds.

With the idea in mind, let's solve a problem to see how they can be powered with fundamental inequalities to solve problems.

Exercise 2.3.3: 2007 Belarusian MO Final Problem 2 (● ○ ○)

Show that the following inequality holds for all $a_1, \ldots, a_{n+1} > 0$.

$$\frac{1}{a_1} + \frac{a_1}{a_2} + \frac{a_1 a_2}{a_3} + \cdots + \frac{a_1 a_2 \cdots a_n}{a_{n+1}} \geq 4(1 - a_1 \cdots a_{n+1})$$

Solution.

Before applying any techniques or inequalities, let's first analyze the inequality. Notice that in our left hand side of the inequality, we have products $a_1 \cdots a_k$ that grows after each term. On our right hand side, we have the product $a_1 \cdots a_{n+1}$. Now letting $p_k = a_1 \cdots a_k$ and $p_0 = 1$, we can see that our right hand side becomes $4(p_0 - p_{n+1})$, which hints that the sequence telescopes. Indeed, consider the following expansion.

$$\begin{aligned}4(1 - a_1 \cdots a_{n+1}) = 4\{&(1 - a_1) + (a_1 - a_1 a_2) + \cdots \\ &+ (a_1 \cdots a_n - a_1 \cdots a_{n+1})\}\end{aligned}$$

Matching each expanded term with the terms in left hand side, it

suffices to show that

$$\frac{a_1 \cdots a_k}{a_{k+1}} \geq 4(a_1 \cdots a_k - a_1 \cdots a_{k+1})$$

holds. Rearranging the terms the following inequality hold.

$$\frac{a_1^2 a_2^2 \cdots a_k^2}{a_1^2 a_2^2 \cdots a_{k+1}^2} + 4a_1 \cdots a_{k+1} \geq 4a_1 \cdots a_k$$

Because $a_k > 0$ by definition, the inequality holds by AM-GM inequality.

The problem above uses the property of telescoping series and AM-GM inequality for a proof. If you see such form, it would be worth suspecting that we can use telescoping technique.

* Homogenization, Normalization * Substitution, uvw method, Ravi Transformation * Fudging, Smoothing, and Ordering * Super-Sums of Square-Weights (SOS) * Majorization Theory * Mixing Variables * Isoperimetric Inequalities

§2.4 Practice Problems

Below are some practice problems with a subjective rating on difficulties. The solutions are included at the end of Math Olympiad part of LibreNotebook.

2.4.1. **2004 USAMO Problem 5 (● ○ ○)**
Show that the following holds for $a, b, c > 0$.

$$(a^5 - a^2 + 3)(b^5 - b^2 + 3)(c^5 - c^2 + 3) \geq (a + b + c)^3$$

2.4.2. **2001 IMO Shortlist A3 (● ○ ○)**

Show that the following inequality holds for arbitrary reals $x_1, x_2, \cdots, x_n$.

$$\frac{x_1}{1+x_1^2} + \frac{x_2}{1+x_1^2+x_2^2} + \cdots + \frac{x_n}{1+x_1^2+\cdots+x_n^2} < \sqrt{n}$$

2.4.3. 2025 KMO Problem 6 (● ● ○)

For arbitrary reals $x_1, x_2, \ldots, x_{99}$ that satisfy $0 < x_1 < x_2 < \cdots < x_{99}$, determine the minimum value of $c > 0$ such that the following inequality always holds.

$$3\sqrt{x_1} + 4\left(\sqrt{x_2 - x_1} + \sqrt{x_3 - x_2} + \cdots + \sqrt{x_{99} - x_{98}}\right) \leq c\left(\sqrt{x_2} + \sqrt{x_3} + \cdots + \sqrt{x_{98}}\right) + 5\sqrt{x_{99}}$$

2.4.4. 2023 IMO Problem 4 (● ● ○)

Let $x_1, x_2, \cdots, x_{2023} > 0$ be pairwise different numbers such that a_n defined as the following is an integer for all $n = 1, 2, \cdots, 2023$.

$$a_n = \sqrt{(x_1 + x_2 + \cdots + x_n)\left(\frac{1}{x_1} + \frac{1}{x_2} + \cdots + \frac{1}{x_n}\right)}$$

Show that $a_{2023} \geq 3034$.

Note 3

[A] Functional Equations

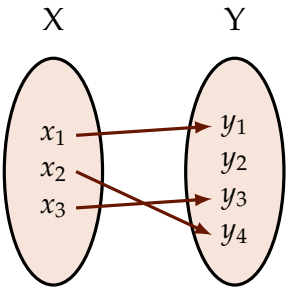
Personally I think one of the most difficult sections in olympiad algebra is functional equations. Unlike other sections that involves application of background knowledge, functional equation problems tend to prioritize ingenuity and creativity. This could be a good news if you are cramming for an exam, but I think it is important to solve many problems to find different patterns in problems involving functional equations.

§3.1 Foundations

Before we get into what a functional equation is, let's briefly review the terms related to functions as they appear quite a lot.

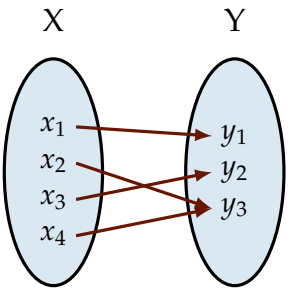
Injective Function

If $f(x_1) = f(x_2)$,
then $x_1 = x_2$



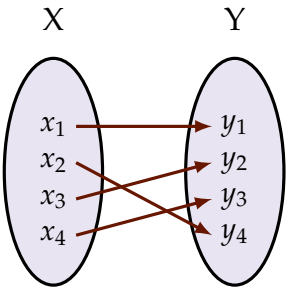
Surjective Function

$$\forall\, y_i \in Y,\ \exists\, x_j \in X$$
$$s.t.\ f(x_j) = y_i$$



Bijective Function

$$\forall\, y_i \in Y,\ \exists!\, x_j \in X$$
$$s.t.\ f(x_j) = y_i$$



While we will be using the terms above, it is worth noting other famous terms describing the same functions.

> Some people use the term **one-to-one** for injective, **onto** for surjective, and **one-to-one onto** for bijective functions.

As mentioned, we don't really need other strong backgrounds for functional equations, though there are plenty of techniques to study. However, the terms injective, surjective, and bijective really cannot be stressed enough.

> A function f has an inverse if and only if it is bijective.

This seemingly obvious statement can come very handy. To show it, you can consider the diagrams above. If the function is only injective, then we have inputs with no corresponding output. If f is surjective only, then there are inputs with two outputs, which is not a function anymore. Another very interesting property to note is strictly increasing and decreasing functions.

A strictly increasing or decreasing function is always injective.

First, note that a strictly increasing function f have the property where if $x < y$ then $f(x) < f(y)$. This satisfies the definition of injective function because f is *strictly* increasing. The case for strictly decreasing is proven in analogous way. With these in mind, let's get started with functional equations!

A functional equation as the name suggest is simply an equation where the variable is a function. To see what it means, let's solve practice problems. Really, there is nothing more to it and we now just need to keep practicing pattern recognition and thinking creatively. However before starting, there is an important note to keep in mind.

The phrase **if and only if** cannot be stressed enough! If you think you've found all the solutions, chances are you probably haven't. Many points are taken off because of the lack of the proof for the **if and only if** condition, and it is always nice to suspect your assumption that you have found all the solutions.

For our first problem, let's solve a classic problem that involves injection, surjection, and bijection.

Exercise 3.1.1: Classic Problem (● ○ ○)

Find all functions f such that $f(f(x) + f(y)) = x + y$ for all $x, y \in \mathbb{Z}$ [Win26].

Solution.

Unlike the traditional approach of substituting different values to x and y, let's start by substituting $y = 1$. Substituting, $f(f(x) + f(1)) = x + 1 = f(f(x + 1) + f(0))$. To find injectivity, surjectivity, or bijectivity of the function f, assume $f(a) = f(b)$. Then, $f(f(a) + f(y)) = f(f(b) + f(y))$ and by definition, $f(a) = f(b)$ leads to $a = b$. In other words, f is injective.

Returning to our previous equation $f(f(x)+f(1)) = f(f(x+1)+f(0))$, we could see that $f(x)+f(1) = f(x+1)+f(0)$ since f is injective. Rearranging, $f(x+1)-f(x) = f(1)-f(0)$ and $f(x)$ is arithmetic sequence. Therefore let $f(x) = ax+b$. Substituting to our given equation, $f(ax+b+ay+b) = a(ax+b+ay+b)+b = x+y$. Solving the equation,

$$\begin{aligned} a(ax+b+ay+b)+b &= x+y \\ \left(a^2-1\right)x + \left(a^2-1\right)y + 2ab + b &= 0 \end{aligned}$$

holds for all $x, y \in \mathbb{R}$. Thus $a = \pm 1$ and $b = 0$. Therefore, $f(x) = \pm x$ are all solutions.

Continuing from this problem, we have a harder problem that requires smart substitutions.

Exercise 3.1.2: 2008 IMO Problem 4 (• • •)

Find all functions $f : (0,\infty) \mapsto (0,\infty)$ that satisfy the following equation for all positive reals a, b, c, d where $ab = cd$.

$$\frac{f(a)^2 + f(b)^2}{f(c^2) + f(d^2)} = \frac{a^2+b^2}{c^2+d^2}$$

Solution.

First, let's start with what we can observe from the problem. The first thing that we can see is that we have squares everywhere. The second thing to note is that substituting $a = b = c = d = 1$ gives us $f(x) = 1$. We can also notice that if we substitute (n, n, n, n) we get $f(n)^2 = f(n^2)$.

With the observations in mind, what we can try is getting rid of the square by substituting $(\sqrt{x}, \sqrt{x}, 1, x)$. This will lead us to following

equations.

$$\begin{aligned}\frac{f(\sqrt{x})^2 + f(\sqrt{x})^2}{1 + f(x^2)} &= \frac{x + x}{1 + x^2} \\ \frac{2f(x)}{1 + f(x)^2} &= \frac{2x}{1 + x^2} \\ 2f(x)(1 + x^2) &= 2x(1 + f(x)^2)\end{aligned}$$

Continuing, we can rearrange the equations to solve for f.

$$\begin{aligned}xf(x)^2 - (1 + x^2)f(x) + x &= 0 \\ (xf(x) - 1)(f(x) - x) &= 0 \\ \therefore f(x) = x \text{ or } f(x) &= \frac{1}{x}\end{aligned}$$

Now we can see that if a function f satisfy the equation given by the problem, then $f(x) = x$ or $f(x) = \frac{1}{x}$. Moreover, if $f(x) = x)$ or $f(x) = \frac{1}{x}$, then f satisfies the original equation as we can see from direct substitution.

If we end our solution here thinking that we have solved the problem, we will lose points for not showing that x and $\frac{1}{x}$ are the only solutions. Although we showed the if and only if condition, we have not showed that f is not a piece-wise function.

Lemma. *$f(x) = x$ and $f(x) = \frac{1}{x}$ are the only solutions to the equation.*

Proof. For the sake of contradiction, let $f(\alpha) \neq \alpha$ and $f(\beta) \neq \frac{1}{\beta}$ for $\alpha, \beta \neq 1$. By our previous conclusions, we can see that it implies $f(\alpha) = \frac{1}{\alpha}$ and $f(\beta) = \beta$.

Substitute $(\sqrt{\alpha}, \sqrt{\beta}, 1, \sqrt{\alpha\beta})$ to eliminate the squares and use both

α and β.

$$\frac{f(\sqrt{\alpha})^2 + f(\sqrt{\beta})^2}{f(1) + f(\alpha\beta)} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\frac{\frac{1}{\alpha} + \beta}{1 + f(\alpha\beta)} = \frac{\alpha+\beta}{1+\alpha\beta}$$

Notice that $f(\alpha\beta)$ is either $\alpha\beta$ or $\frac{1}{\alpha\beta}$. Therefore, let $f(\alpha\beta) = \alpha\beta$. Solving the equation, the following equations are obtained.

$$\frac{\frac{1}{\alpha} + \beta}{1 + f(\alpha\beta)} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\frac{\frac{1}{\alpha} + \beta}{1 + \alpha\beta} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\frac{1}{\alpha} = \alpha$$

This contradicts with the fact that $\alpha \neq 1$. Continuing, let $f(\alpha\beta) = \frac{1}{\alpha\beta}$.

$$\frac{\frac{1}{\alpha} + \beta}{1 + f(\alpha\beta)} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\frac{\frac{1}{\alpha} + \beta}{1 + \frac{1}{\alpha\beta}} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\frac{\beta + \alpha\beta^2}{\alpha\beta + 1} = \frac{\alpha+\beta}{1+\alpha\beta}$$
$$\alpha\beta^2 = \alpha$$
$$\beta^2 = 1$$

This also contradicts with $\beta \neq 1$. Therefore, there exists no $\alpha, \beta \neq 1$ such that $f(\alpha) \neq \alpha$ and $f(\beta) \neq \frac{1}{\beta}$. □

Hence, $f(x) = x$ and $f(x) = \frac{1}{x}$ are the only solutions.

As you probably have noticed from the solution, while vast amount of knowledge in different functions may come handy, ingenuity and clever substitution would do most of the work when solving problems. Here is

another problem on using clever substitutions.

Exercise 3.1.3: 2020 Baltic Way Problem 4 (● ○ ○)

Find all functions $f : \mathbb{R} \to \mathbb{R}$ that satisfy the following for all $x, y \in \mathbb{R}$.

$$f(f(x) + x + y) = f(x + y) + yf(y)$$

(Video Solution)

Solution.

First, let's start by doing typical substitutions. Substituting $x = y = 0$ gives $f(f(0)) = f(0)$ and $x = -y$ gives $f(f(x)) = f(0) - xf(-x)$. Also, trying $y = 0$ gives $f(f(x) + x) = f(x)$. Looking at the last observation, we could assume that $f(x) = 0$. Moreover our first observation also suggests that $f(0) = 0$. With the observations, let's first investigate the value of $f(0)$.

Consider the following equation by the last observation with $y = f(x) + x$.

$$\begin{aligned} f(x) = f(y) = f(f(y) + y) &= f(f(x) + f(x) + x) \\ &= f(x + f(x)) + f(x)f(f(x)) = f(x) + f(x)f(f(x)) \end{aligned}$$

Therefore, $f(x)f(f(x)) = 0$ and $f(0)f(f(0)) = 0$. Note that because $f(f(0)) = f(0)$, $f(0)^2 = 0$ and $f(0) = 0$ holds. Indeed $f(0) = 0$!

Continuing, substitute $x = 0$ to the given equation.

$$f(f(0) + 0 + y) = f(y) = f(0 + y) + yf(y)$$

In other words, $yf(y) = 0$ holds for all $y \in \mathbb{R}$. Specially, $f(y) = 0$ for all $y \neq 0$ and we have found that $f(0) = 0$. Therefore, $f(x) = 0$ for all $x \in \mathbb{R}$ is the only solution.

In this problem, using the observation carefully and letting $y = f(x) + x$ was the key part of the solution. There are many topics to discuss even

if it requires less prerequisite knowledge to actually solve the problem. With the intuition discussed in this note, let's discuss few important topics and patterns that are widely used.

§3.2 Cauchy's FE and Its Variants

Unlike the other problems, if there is one thing that you must know to solve a FE problem, it would be Cauchy FE. This could be a very nice problem, but could also be a very mean one. Before we look at what Cauchy FE is, let's discuss some very introductory topics from real analysis.

Theorem 3.2.1: Density of Rational Numbers

For any real numbers a and b, it is possible to choose a rational number q such that $a < q < b$.

Proof.

Consider the interval (na, nb) for some integer n. Choose a sufficiently large positive integer n such that $nb - na > 1$. Because the length of the interval is greater than 1, we can choose another integer m such that $na < m < nb$. Therefore, $a < \frac{m}{n} < b$ and $\frac{m}{n} \in \mathbb{Q}$ since $m, n \in \mathbb{Z}$.

This seems quite obvious when you think about it. However, it is important to consider and question the very seemingly obvious properties since conditions are very important for Cauchy FE. Continuing, we could try the same for irrational numbers.

Theorem 3.2.2: Density of Irrational Numbers

Given arbitrary real numbers a and b, it is possible to choose a irrational number q such that $a < q < b$.

Proof.

First, consider the inequality $\frac{a}{\sqrt{2}} < r < \frac{b}{\sqrt{2}}$ for some rational number r by Theorem 3.2.1. Multiplying $\sqrt{2}$, $a < \sqrt{2}r = q < b$ holds.

Using these two, we can notice that given two reals, we can choose a real number between them. Now before moving on to Cauchy FE, consider the following theorem.

Theorem 3.2.3: Bolzano-Weierstrass Theorem

Any real number x can be written as the limit of bounded sequence of rational numbers and of irrational numbers.

Proof.

Let r_n be an increasing sequence of rational numbers defined as $x - \frac{1}{n} < r_n < x - \frac{1}{n+1}$ by Theorem 3.2.1. Therefore,

$$x - \frac{1}{n} < r_n < x - \frac{1}{n+1} < r_{n+1}$$

holds and $x = \lim_{n\to\infty} r_n$ by Squeeze Theorem. If r_n is a decreasing sequence defined as $x + \frac{1}{n+1} < r_n < x + \frac{1}{n}$, then $r_{n+1} < x + \frac{1}{n+1} < r_n < x + \frac{1}{n}$. Therefore $x = \lim_{n\to\infty} r_n$ by Squeeze Theorem. The case of irrational numbers are proven in analogous way.

Now with these ideas in mind, let's finally discuss Cauchy's Functional Equations!

Definition 3.2.4

Equations with the following form is known as **Cauchy's Functional Equations**.

$$f(x+y) = f(x) + f(y)$$

Now this is pretty straight forward for certain input domain. Let's start by what we can notice just by looking at the equations. Substituting

$x = y = 0$, we get $f(0) = 0$. Moreover, substituting $y = -x$ gives $f(x) = -f(-x)$ and we can see that f is an odd function. With these in mind and having $k = f(1)$, let's start looking into different input domains.

I. $f : \mathbb{N} \to \mathbb{R}$

This case is pretty straightforward. Consider the following equations for some integer n.

$$f(n) = f(n-1) + f(1) = f(n-2) + f(1) + f(1) = \cdots = nf(1)$$

Therefore, $f(x) = kx$ if x is defined under natural numbers. An extension would be using integers.

II. $f : \mathbb{Z} \to \mathbb{R}$

For this case, we will be using the fact that f is an odd function. For $x > 0$, we have showed that $f(x) = kx$. For $x < 0$, consider the following equations.

$$f(x) = -f(-x) = -k(-x) = kx$$

Therefore with $f(0) = 0$, f is still linear.

III. $f : \mathbb{Q} \to \mathbb{R}$

The next extension that we can have is defining x over the rational numbers. Let p and q be relatively prime integers. First consider the following equation.

$$f\left(\frac{p}{q}\right) = f\left(\frac{1}{q} + \cdots + \frac{1}{q}\right) = f\left(\frac{1}{q}\right) \cdot p$$

Therefore, $f(1) = f\left(\frac{1}{q}\right) \cdot q$ and substituting back to the equation gives the following.

$$f\left(\frac{p}{q}\right) = \frac{f(1)}{q} \cdot p = \frac{p}{q}k$$

Therefore, $f(x) = kx$ also holds in this case. The final case that we can

consider is when $f : \mathbb{R} \to \mathbb{R}$.

IV. $f : \mathbb{R} \to \mathbb{R}$

This is the part where it gets quite tricky and where conditions becomes very important. There are many different conditions that we could consider for this case, but let's focus on two major conditions for now.

IV.I f is an increasing or decreasing function.

First, let's consider the case when f is increasing. Assume for the sake of contradiction that $f(x) \neq kx$. If $f(x) > kx$, then $x < \frac{f(x)}{k}$ and we can choose a rational q such that $x < q < \frac{f(x)}{k}$ by Theorem 3.2.1. Therefore, $f(x) \leq f(q) = kq$ since $q \in \mathbb{Q}$. Continuing, $\frac{f(x)}{k} \leq q < \frac{f(x)}{k}$ is obtained, which is a contradiction.

This time, assume that $f(x) < kx$. We can choose a rational q such that $\frac{f(x)}{k} < q < x$. Continuing, we get $kq = f(q) \leq f(x) < kq$ which again is a contradiction. This proves that if f is an increasing function, then $f(x) = kx$ for $x \in \mathbb{R}$.

The case when f is decreasing is quite simple. Let $g(x) = -f(x)$. By construction, g is an increasing function and the following equations hold.

$$-f(x+y) = g(x+y) = g(x)+g(y) = -f(x)-f(y) = -(f(x)+f(y))$$

Therefore, $f(x+y) = f(x)+f(y)$. This completes the proof!

Note that these conditions are necessary. We cannot blindly assume that $f(x) = kx$ for $x \in \mathbb{R}$ without the condition. One very, very popular way of knowing if a function is increasing is $f(xy) = f(x)f(y)$.

If $f(xy) = f(x)f(y)$, then $f(x^2) = f(x)^2 \geq 0$. Therefore for positive y, $f(x) = f(x-y)+f(y) \geq f(x-y)$ and f is increasing.

> If $f(xy) = f(x)f(y)$, then f is an increasing function.

This is very popular, and will make our lives greatly easier to use $f(x) = kx$.

IV.II f is continuous.

The second very popular condition is when f is continuous. Note that by Bolzano-Weierstrass Theorem, there exists an increasing rational sequence a_n and decreasing rational sequence b_n such that $x = \lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n$ for any real x. Therefore, the following equations hold.

$$\begin{aligned} f(x) &= f\left(\lim_{n\to\infty} a_n\right) = \lim_{n\to\infty} f(a_n) = \lim_{n\to\infty} ka_n = kx \\ &= f\left(\lim_{n\to\infty} b_n\right) = \lim_{n\to\infty} f(b_n) = \lim_{n\to\infty} kb_n = kx \end{aligned}$$

Therefore, $f(x) = kx$ holds if f is continuous.

This is the basic idea of Cauchy FE! With this in mind, let's solve few practice problems to see how they are used.

3.2.1 Pexider Functional Equation

§3.3 Symmetric and Cyclic Equations

§3.4 Invariance

§3.5 The CDE Method

§3.6 Additional Topics

Involution, monotonicity

Some interesting problems gives bound for the function for us to use

bounding argument.

> The key to these kind of problems is to perform certain operations on the function and show that it contradicts with the given criteria.

Here is an interesting problem from the document *169 Functional Equations* by Moubinool the showcase the bounding through a specific criteria given by the problem.

Exercise 3.6.1

Find all functions $f : \mathbb{R} \to \mathbb{R}$ such that $\left\{ \frac{f(x)}{x} \mid x \in \mathbb{R} \text{ and } x \neq 0 \right\}$ is a finite set and $f(x - 1 - f(x)) = f(x) - x - 1$ for all real x.

Solution.

For this problem, because the terms inside f in the left-hand side and the terms in the right-hand side are similar, let's try and make them equal like the following.

$$f(x - 1 - f(x)) = -(x - 1 - f(x)) - 2$$

Let $\alpha = x - 1 - f(x)$ and we get $f(\alpha) = -\alpha - 2$. But there is a problem here. The number of terms for $\frac{f(x)}{x}$ is bounded, but we have infinitely many $\frac{-\alpha-2}{\alpha}$. This tells us that α is finite for the condition to be satisfied.

Further observing the problem, let's substitute α for x. We obtain

$$\begin{aligned} f(\alpha - 1 - f(\alpha)) &= -(\alpha - 1 - f(\alpha)) - 2 \\ f(\alpha - 1 + \alpha + 2) &= -\alpha - 1 - \alpha - 2 \end{aligned}$$

and $f(2\alpha + 1) = -(2\alpha + 1) - 2$. This shows that $2\alpha + 1$ is also an element of a finite set, say A, that contains α.

Continuing, we can try and substitute $2\alpha + 1$ for x. We obtain $f(2\alpha + 1 - 1 + 2\alpha + 1 + 2) = -(2\alpha + 1) - 2 - 2\alpha - 1 - 1$ and $f(4\alpha + 3) =$

$-(4\alpha+3)-2$. This shows that $4\alpha+3 \in A$. From the patterns, we could hypothesize that $2^n\alpha+2^n-1 \in A$. To show this, it suffices to show that if $2^{n-1}\alpha+2^{n-1}-1 \in A$, then $2^n\alpha+2^n-1 \in A$. Consider the following equations that we get for substituting $x = 2^{n-1}\alpha + 2^{n-1}-1$.

$$\begin{aligned} & f\left(2^{n-1}\alpha+2^{n-1}-1-1+\left(2^{n-1}\alpha+2^{n-1}-1)+2\right)\right) \\ =& -\left(2^{n-1}\alpha+2^{n-1}-1\right)-2-\left(2^{n-1}\alpha+2^{n-1}-1\right)-1 \\ & f\left(2^n\alpha+2^n-1\right) = -\left(2^n\alpha+2^n-1\right)-2 \end{aligned}$$

Indeed by induction, $2^n\alpha+2^n-1 \in A$ for all positive integers n. In other words, $\alpha = -1$ is the only possible case and $\alpha = x-1-f(x) = -1$. Therefore, $f(x) = x$ is the only solution.

Note 4

[A] Polynomials

§4.1 Foundations

Before we get into the world of polynomials in olympiad algebra, let's discuss some notations that we will be using throughout the notes and key theorems as a review. Please feel free to skip over to the next section if you are already familiar with the topics below.

Definition 4.1.1

A **monomial** is a single term composed with variables raised to powers of whole numbers and a coefficient. **Polynomial**, denoted as $P(x)$, is a sum of finitely many monomials.

Some examples of monomial includes $x, 3a^2, 9, (5-2i)x^3y^2$ while $x^{-2}, a+b, \sin x$ are not monomials. Polynomials are typically written with the following sum notation.

$$P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

With this in mind, we can define some notations that occasionally appear.

Definition 4.1.2

Below are common set notations for polynomials where n is a non-negative integer.

$$\mathbb{Z}[x] := \left\{ \sum_{i=0}^{n} a_i x^i \mid a_i \in \mathbb{Z} \right\}$$

$$\mathbb{Q}[x] := \left\{ \sum_{i=0}^{n} a_i x^i \mid a_i \in \mathbb{Q} \right\}$$

$$\mathbb{R}[x] := \left\{ \sum_{i=0}^{n} a_i x^i \mid a_i \in \mathbb{R} \right\}$$

$$\mathbb{C}[x] := \left\{ \sum_{i=0}^{n} a_i x^i \mid a_i \in \mathbb{C} \right\}$$

Continuing with the notations, let's discuss some key theorems that cannot be omitted when discussing polynomials.

Theorem 4.1.3: Fundamental Theorem of Algebra

Any nonconstant polynomial $P(x) \in \mathbb{C}[x]$ has at least one complex root [LNS07].

There are multiple proofs for this beautiful theorem, but I think they are out of scope for the notes. Trust me, the theorem is true, and maybe I will include a rigorous proof of the statement in other parts of LibreNotebook. Now this theorem can lead us to another very beautiful proven assumption that we can use when solving problems.

Corollary 4.1.4

All polynomials $P(x) \in \mathbb{C}[x]$ with $\deg(P) = n > 1$ can be factored into n linear factors with multiplicity.

Proof.

Notice that by FTA, $P_n(x)$ can be represented as $P_n(x) = a(x - z_1)P_{n-1}(x)$ where $P_k(x)$ is a polynomial with degree k. Continuing

recursively until $n = 1$, $P_n(x)$ can be written as the following.

$$\begin{aligned} P_n(x) &= a(x - z_1)(x - z_2) \cdots (x - z_{n-1})P_1(x) \\ &= a(x - z_1)(x - z_2) \cdots (x - z_{n-1})(x - z_n) \end{aligned}$$

Therefore, any polynomial with degree $n > 1$ can be factored into exactly n linear factors with multiplicity.

The corollary also implies that there exists n roots counting multiplicity. Another key property to remember is remainder theorem and factor theorem. They are pretty self-evident, so I will omit the proof here, but below is the simple statement.

Theorem 4.1.5

For any polynomial P, if the polynomial is divided by $x - a$, then the remainder is $P(a)$. Moreover if $P(a) = 0$, then $(x - a)$ is a factor of P.

The theorem above is basically stating that polynomials can be written as the following.

$$P(x) = (x - a)Q(x) + P(a)$$

This is pretty self evident as we get $P(a) = P(a)$ if we substitute $x = a$ and $Q(x)$ is the quotient polynomial.

Before we finish our review, here is the final theorem.

Theorem 4.1.6: Rational Root Theorem

Every rational root $\frac{p}{q}$ in lowest terms of a polynomial $P(x) = \sum_{i=0}^{n} a_i x^i$ with $a_0, a_n \neq 0$ satisfies the following conditions: $p \mid a_0$ and $q \mid a_n$.

Proof.

First and foremost, notice that if $\frac{p}{q}$ is a solution to $P(x) = 0$, then

the following equations hold.

$$P\left(\frac{p}{q}\right) = \sum_{i=0}^{n} a_i \left(\frac{p}{q}\right)^i = 0$$
$$a_0 q^n + a_1 p q^{n-1} + a_2 p^2 q^{n-2} + \cdots + a_n p^n = 0$$

Therefore, the following equations can be derived.

$$q\left(a_0 q^{n-1} + a_1 p q^{n-2} + a_2 p^2 q^{n-3} + \cdots + a_{n-1} p^{n-1}\right) = -a_n p^n$$
$$p\left(a_1 q^{n-1} + a_2 p q^{n-2} + a_3 p^2 q^{n-3} + \cdots + a_n p^{n-1}\right) = -a_0 q^n$$

Because p and q are coprime, the first equation reveals that $q \mid a_n$ and the second equation shows that $p \mid a_0$.

This theorem sometimes comes handy if we had to blindly guess a solution to further factorize as it narrows down our choices. Now with the reminders, let's discuss the topic that we all may have heard of: Roots of Unity.

§4.2 Roots of Unity

Theorem 4.2.1: De Moivre's Theorem

For any real x and integer n, the following equation holds.

$$(\cos\theta + i\sin\theta)^n = \cos n\theta + i\sin n\theta$$

Exercise 4.2.2: 1976 USAMO Problem 5 (● ● ○)

For polynomials $P(x)$, $Q(x)$, $R(x)$, and $S(x)$ that satisfy the following equation,

$$P(x^5) + xQ(x^5) + x^2R(x^5) = (x^4 + x^3 + x^2 + x + 1)S(x),$$

show that $x - 1$ is a factor of $P(x)$.

Solution.

Looking at the equation, we can see $x^4 + x^3 + x^2 + x + 1$ and x^5. This should remind the equation $x^5 - 1 = (x-1)(x^4 + x^3 + x^2 + x + 1)$. In other words, we should be reminded of the fifth roots of unity where we define $\omega_5 = 1$ and $\omega_1 + \omega_2 + \cdots + \omega_5 = 0$.

The next thing that we want to try is substituting $\omega_1, \cdots, \omega_4$ since we know that the equation always hold.

$$\begin{aligned} P(1) + \omega_1 Q(1) + \omega_1^2 R(1) &= 0 \\ P(1) + \omega_2 Q(1) + \omega_2^2 R(1) &= 0 \\ P(1) + \omega_3 Q(1) + \omega_3^2 R(1) &= 0 \\ P(1) + \omega_4 Q(1) + \omega_4^2 R(1) &= 0 \end{aligned}$$

Now we see that only $P(1)$ has equal coefficient across all four equations. We also have defined $\omega_1, \cdots \omega_5$ to be the fifth root of unity. Meaning, if we multiply the equations appropriately, we would be able to cancel out $Q(1)$ and $R(1)$.

$$\begin{aligned} \omega_1 P(1) + \omega_1^2 Q(1) + \omega_1^3 R(1) &= 0 \\ \omega_2 P(1) + \omega_2^2 Q(1) + \omega_2^3 R(1) &= 0 \\ \omega_3 P(1) + \omega_3^2 Q(1) + \omega_3^3 R(1) &= 0 \\ \omega_4 P(1) + \omega_4^2 Q(1) + \omega_4^3 R(1) &= 0 \end{aligned}$$

Subtracting the four equations from the original equations, we ob-

tain the following equations.

$$\begin{aligned}
&P(1)+\omega_1 Q(1)+\omega_1^2 R(1)\\
-&\left(\omega_1 P(1)+\omega_1^2 Q(1)+\omega_1^3 R(1)\right)\\
+&\quad P(1)+\omega_2 Q(1)+\omega_2^2 R(1)\\
-&\left(\omega_2 P(1)+\omega_2^2 Q(1)+\omega_2^3 R(1)\right)\\
+&\quad P(1)+\omega_3 Q(1)+\omega_3^2 R(1)\\
-&\left(\omega_3 P(1)+\omega_3^2 Q(1)+\omega_3^3 R(1)\right)\\
+&\quad P(1)+\omega_4 Q(1)+\omega_4^2 R(1)\\
-&\left(\omega_4 P(1)+\omega_4^2 Q(1)+\omega_4^3 R(1)\right)=0
\end{aligned}$$

Solving the equation, we get $5P(1)=0$ and $x-1$ is a factor of $P(x)$.

4.2.1 Roots of Unity Filter

$$a_0+a_n+a_{2n}+\cdots=\frac{f(\omega)+f\left(\omega^2\right)+\cdots+f\left(\omega^n\right)}{n}$$

Exercise 4.2.3

In Enoch's dreamland, there are 999 mysterious boxes each labeled uniquely from 1 to 999. If Enoch chooses the box(es) randomly and the sum of the number of chosen box(es) is divisible by 3, then he earns a dollar from the owner of the boxes. However, he is not granted money if he chooses the same combination next time. What is the maximum amount of money Enoch can earn?

Solution.

This problem is basically asking for the number of ways to choose numbers randomly from 1 through 999 such that their sum is divisible by 3. In other words, how many subsets of the set $\{1,2,\ldots,999\}$ have the sum of their elements divisible by three?

When a large sequence appears, it is reasonable to generate func-

tions. For the function that we are generating, we will associate the subsets with the its coefficients. This allows an algebraic approach towards the combinatorial problem.

Consider the following function.

$$f(x) = \prod_{i=1}^{999} \left(1 + x^i\right) = (1+x)\left(1+x^2\right)\left(1+x^3\right)\cdots\left(1+x^{999}\right)$$

Notice that each number in the subset corresponds to the exponents of x. Consider the following samples to see how they are related.

$$\text{Exponent: } 3 \Rightarrow \text{Coefficient: } 2$$
$$\left.\begin{aligned} 3 &= 3 && \Rightarrow \{3\} \\ 3 &= 1+2 && \Rightarrow \{1,2\} \end{aligned}\right\} \quad 2 \text{ Cases}$$

$$\text{Exponent: } 6 \Rightarrow \text{Coefficient: } 4$$
$$\left.\begin{aligned} 6 &= 6 && \Rightarrow \{6\} \\ 6 &= 1+5 && \Rightarrow \{1,5\} \\ 6 &= 2+4 && \Rightarrow \{2,4\} \\ 6 &= 1+2+3 && \Rightarrow \{1,2,3\} \end{aligned}\right\} \quad 4 \text{ Cases}$$

The process of investigating the subsets with sum n perfectly mirrors the behavior of terms of x^n and its coefficients. Therefore, the sum of coefficients of x^{3k} where $k \in \mathbb{N}_0$ is the number of subsets whose sum of the elements are divisible by 3.

Continuing, we can rewrite the function f.

$$f(x) = a_0 + a_1x^1 + a_2x^2 + \cdots + a_{499500}x^{499500}$$

Using the third roots of unity with $\omega_1, \omega_2, \omega_3$,

$$\begin{aligned} f(\omega_1) &= a_0 + a_1\omega_1 + \cdots + a_{499500}\omega_1^{499500} \\ f(\omega_2) &= a_0 + a_1\omega_2 + \cdots + a_{499500}\omega_2^{499500} \\ f(\omega_3) &= a_0 + a_1\omega_3 + \cdots + a_{499500}\omega_3^{499500} \end{aligned}$$

$$\begin{aligned} f(\omega_1) + f(\omega_2) + f(\omega_3) &= 3a_0 + 3a_1(\omega_1 + \omega_2 + \omega_3) + \cdots \\ &\quad + a_{499500}\left(\omega_1^{499500} + \omega_2^{499500} + \omega_3^{499500}\right) \\ &= 3a_0 + 3a_3 + 3a_6 + \cdots + 3a_{499500} \end{aligned}$$

holds since $e^{i\frac{0\pi}{3}}, e^{i\frac{2\pi}{3}}, e^{i\frac{4\pi}{3}}$ will continue to repeat. Therefore, it suffices to find the values of $\frac{f(\omega_1)+f(\omega_2)+f(\omega_3)}{3}$. Using the period again, we can obtain the following.

$$\begin{aligned} \frac{f(\omega_1) + f(\omega_2) + f(\omega_3)}{3} &= \frac{\left((1+\omega_1)\left(1+\omega_1^2\right)\left(1+\omega_1^3\right)\right)^{333}}{3} \\ &\quad + \frac{\left((1+\omega_2)\left(1+\omega_2^2\right)\left(1+\omega_2^3\right)\right)^{333}}{3} \\ &\quad + \frac{\left((1+\omega_3)\left(1+\omega_3^2\right)\left(1+\omega_3^3\right)\right)^{333}}{3} \end{aligned}$$

Without loss of generality, assume that $\omega_1 = e^{i\frac{0\pi}{3}}$, $\omega_2 = e^{i\frac{2\pi}{3}}$, and $\omega_3 = e^{i\frac{4\pi}{3}}$. Except the term $(1+\omega_1)\left(1+\omega_1^2\right)\left(1+\omega_1^3\right)$, the other expressions could be rewritten as $(1+\omega_1)(1+\omega_2)(1+\omega_3)$. Moreover since $1+\omega_i$ is the root of $(x-1)^3 = 1$,

$$(1+\omega_1)(1+\omega_2)(1+\omega_3) = 2.$$

by Vieta's formula. Continuing,

$$\frac{f(\omega_1) + f(\omega_2) + f(\omega_3)}{3} = \frac{2^{999} + 2^{333} + 2^{333}}{3} = \frac{2^{334} + 2^{999}}{3}$$

Therefore, Enoch can earn $\frac{2^{334}+2^{999}}{3}$ dollars in total.

Note 5

[A] Sequence and Series

Sequence and series are one of the common topics in algebraic combinatory and even just algebra problems. In this note, we review some of the foundational topics and intuition on sequence and series, then we discuss more advanced topics from recursion and different type of sums.

§5.1 Prerequisites

Problems involving sequence and series appeared a lot in competition style problems. Let's review some of the topics and gain some intuition on olympiad style sequence and series with Abel transformation.

5.1.1 Telescoping

Telescoping is an essential skill that we learn when we first get started with competition math. Because this content itself is not too difficult, let's finish our review with an example problem.

Exercise 5.1.1: USAMTS Year 11 Round 4 Problem 3 (● ○ ○)

Find S defined as the following.

$$S = \sqrt{1+\frac{1}{1^2}+\frac{1}{2^2}} + \sqrt{1+\frac{1}{2^2}+\frac{1}{3^2}} + \cdots + \sqrt{1+\frac{1}{1999^2}+\frac{1}{2000^2}}$$

Solution.

First, let's rewrite the sum S using the sum notation.

$$S = \sum_{i=1}^{1999} \sqrt{1+\frac{1}{i^2}+\frac{1}{(i+1)^2}}$$

Now we see the terms $1+\frac{1}{i^2}+\frac{1}{(i+1)^2}$ inside a square root. To get rid of the annoying square root, we can try writing the terms in a perfect square. Consider the following equation.

$$\begin{aligned}&\left(1+\frac{1}{n}+\frac{1}{n+1}\right)^2\\ &= 1+\frac{1}{n^2}+\frac{1}{(n+1)^2}+2\left(\frac{1}{n}+\frac{1}{n+1}+\frac{1}{n(n+1)}\right)\end{aligned}$$

Notice that because $\frac{1}{n}-\frac{1}{n+1}=\frac{1}{n(n+1)}$, it would be much preferable to have $\frac{1}{n}-\frac{1}{n}-\frac{1}{n(n+1)}$ instead of $\frac{1}{n}+\frac{1}{n+1}+\frac{1}{n(n+1)}$. Therefore, the following equations are obtained.

$$\begin{aligned}&\left(1+\frac{1}{n}-\frac{1}{n+1}\right)^2\\ &= 1+\frac{1}{n^2}+\frac{1}{(n+1)^2}+2\left(\frac{1}{n}-\frac{1}{n+1}-\frac{1}{n(n+1)}\right)\\ &= 1+\frac{1}{n^2}+\frac{1}{(n+1)^2}\end{aligned}$$

Substituting the equation to the series, we obtain the following

equations.

$$\begin{aligned} S &= \sum_{i=1}^{1999} \sqrt{1+\frac{1}{i^2}+\frac{1}{(i+1)^2}} = \sum_{i=1}^{1999} \sqrt{\left(\frac{1}{i}-\frac{1}{i+1}+1\right)^2} \\ &= \sum_{i=1}^{1999} \left(\frac{1}{i}-\frac{1}{i+1}+1\right) = 1999+\frac{1}{1}-\frac{1}{2000} \\ &= \frac{2000^2-1}{2000} = \frac{3999999}{2000} \end{aligned}$$

Thus, $S = \frac{3999999}{2000}$.

I think telescoping when paired with inequalities like Cauchy and products becomes much powerful! Continuing, let's discuss Abel's transformation.

5.1.2 Abel Transformation

There are many different names of this, and when you search "Abel Transformation" you would probably get different summation concerning integration by parts. But let's stick with this term because I think it is quite popular.

Definition 5.1.2: Abel Transformation

Given reals a_i and b_i, the following equation holds where $a_{n+1} = 0$.

$$\sum_{i=1}^{n} a_i b_i = \sum_{i=1}^{n} (a_i - a_{i+1}) \left(\sum_{k=1}^{i} b_k\right)$$

This transformation is quite unique. To derive this, simply expand the right-hand side. Then magically, every residual terms telescope. To see how this can be applied in real life, here is an example problem.

Exercise 5.1.3: Classic Problem

If $(a_1, \ldots, a_n) \succ (b_1, \ldots, b_n)$ with $a_i \geq a_j \geq 0$ and $b_i \geq b_j \geq 0$ for $i > j$, show that $\sum_{i=1}^{n} a_i^k \geq \sum_{i=1}^{n} b_i^k$ for all positive integer k.

Solution.

We could use Karamata to prove this, but that would ruin all the fun! For this problem, notice that we need to show that $\sum_{i=1}^{n}(a_i^k - b_i^k) \geq 0$. Because the following equation

$$a_i^k - b_i^k = (a_i - b_i)\left(a_i^{k-1} + a_i^{k-2}b_i + \cdots + b_i^{k-1}\right)$$

holds, let $x_i = a_i^{k-1} + a_i^{k-2}b_i + \cdots + b_i^{k-1}$ and $y_i = a_i - b_i$. Rewriting $\sum_{i=1}^{n}(a_i^k - b_i^k)$, it suffices to show that $\sum_{i=1}^{n} x_i y_i \geq 0$. By Abel transform, we obtain following the equation.

$$\sum_{i=1}^{n} x_i y_i = \sum_{i=1}^{n}(x_i - x_{i+1})(y_1 + \cdots + y_i)$$

Notice that $x_i - x_{i+1} \geq 0$ since $a_i \geq a_j \geq 0$ and $b_i \geq b_j \geq 0$ for $i > j$. Moreover, because $(a_1, \ldots, a_n) \succ (b_1, \ldots, b_n)$, $y_1 + \cdots + y_i > 0$. Therefore, the inequality holds.

This is it for this section! In the next section, we will discuss recurrence relation in depth.

§5.2 Recurrence Relation

Another very popular type of problems in math olympiads and competition in general is recurrence relation. This section discusses different forms of recurrence relation and related problems. One more thing before we get started is that this section is largely based on a series of YouTube videos from WinnersKMO [Win26].

As you probably know, recurrence relation focuses on n^{th} term given its relation with preceding k^{th} terms. One very famous example would be Fibonacci sequence whose terms are defined as $F_n = F_{n-1} + F_{n-2}$ where $F_0 = F_1 = 1$. The problems are relatively easy in my opinion until it gets incredibly hard, so let's get started with some easy cases.

I. $a_n = a_{n-1} + f(n)$

This default type has an effective way of solving because it always gets down to a single formula, which becomes quite evident when we telescope. Consider the following equations.

$$\begin{aligned} a_n - a_{n-1} &= f(n) \\ a_{n-1} - a_{n-2} &= f(n-1) \\ &\vdots \\ a_2 - a_1 &= f(2) \end{aligned}$$

Adding the equations, the left-hand side telescopes and we obtain $a_n - a_1 = \sum_{k=2}^{n} f(k)$.

> Given a recurrence relation $a_n = a_{n-1} + f(n)$, each term can be represented with the following formula.
>
> $$a_n = \sum_{k=1}^{n} f(k) + a_1 - f(1)$$

There will be lots of formulas from now, but I honestly think there is no need to memorize them because if we just remember the idea to get to the formulas, then we can solve the problems. Here is an example problem before we move on to the next case.

Exercise 5.2.1: Enoch's Problem

Let $a_1, \ldots, a_k$ be a sequence such that $a_n = a_{n-1} + n^2 - 2n + 1$ for $n > 1$ and $a_1 = 2$. Find a_{101}.

Solution.

This sequence has the form $a_n = a_{n-1} + f(n)$ where $f(n) = n^2 -$

$2n+1$. Telescoping, we obtain the following equations.

$$\begin{aligned} a_{101} &= \sum_{k=1}^{101} (k-1)^2 + 2 - 1 = \sum_{n=0}^{100} n^2 + 1 \\ &= \frac{100(100+1)(200+1)}{6} + 1 = 338351 \end{aligned}$$

Therefore, $a_{101} = 338351$.

Continuing with the first form, here is the second one.

II. $a_n = a_{n-1} f(n)$

This case is also very popular and can be solved using the same method as before.

$$\begin{aligned} \frac{a_n}{a_{n-1}} &= f(n) \\ \frac{a_{n-1}}{a_{n-2}} &= f(n-1) \\ &\vdots \\ \frac{a_2}{a_1} &= f(2) \end{aligned}$$

Telescoping, we obtain $\frac{a_n}{a_1} = \prod_{k=2}^{n} f(k)$.

The terms in a recurrence relation $a_n = a_{n-1} f(n)$ can be represented with the following formula.

$$a_n = a_1 \prod_{k=2}^{n} f(k)$$

Here is another problem derived from the previous one.

Exercise 5.2.2: Enoch's Problem

Let $a_1, \ldots, a_k$ be a sequence such that $a_n = a_{n-1}(n^2 - 2n + 1)$ for $n > 1$ and $a_1 = 1$. Find a_{101}.

Solution.

Notice that this sequence has the form $a_n = a_{n-1}f(n)$ where $f(n) = (n-1)^2$. Telescoping, we obtain the following equations.

$$a_{101} = \prod_{k=2}^{101}(k-1)^2 = \prod_{n=1}^{100} n^2 = (100!)^2$$

Therefore, $a_{101} = (100!)^2$.

The first two cases were relatively simple. Continuing, we can continue with their variations.

III. $a_n = pa_{n-1} + q$

From this case, there are many different ways to get an answer. Let's start with my personal favorite one.

Solution I.

The first method is to directly substitute.

$$\begin{aligned} a_n &= pa_{n-1} + q \\ &= p(pa_{n-2} + q) + q = p^2 a_{n-2} + (p+1)q \\ &= p^2(pa_{n-3} + q) + (p+1)q = p^3 a_{n-3} + (p^2 + p + 1)q \\ &\quad \vdots \\ &= p^{n-1}a_1 + q\sum_{k=0}^{n-2} p^k \end{aligned}$$

Rearranging the equation, we obtain the following equation.

$$a_n = p^{n-1}a_1 + \frac{1-p^{n-1}}{1-p} \cdot q$$

I personally like this method because it feels intuitive and quite natural to just substitute. The second method is slightly less intuitive, but it should be faster and better if you prone to make arithmetic mistakes.

Solution II.

This method will convert the form of this recursive equation. First, assume that there exists an α such that the following equation holds.

$$a_n - \alpha = p(a_{n-1} - \alpha)$$

If there exists such alpha, then it is evident that we can telescope to obtain the following equation.

$$a_n - \alpha = p^{n-1}(a_1 - \alpha)$$

This is much easier to work with compared to our initial equation.

From the equation $a_n - \alpha = p(a_{n-1} - \alpha)$, we can notice that $pa_{n-1} + q - \alpha = p(a_{n-1} - \alpha)$ and $q = \alpha(1-p)$. In other words, $\alpha = \frac{q}{1-p}$. This shows that α always exists given that $p \neq 1$. In other words, whenever we see this case, we can let $\alpha = \frac{q}{1-p}$ and use $a_n - \alpha = p^{n-1}(a_1 - \alpha)$. Substituting,

$$\begin{aligned} a_n &= p^{n-1}a_1 + \alpha\left(1 - p^{n-1}\right) = p^{n-1}a_1 + \frac{q}{1-p}\left(1 - p^{n-1}\right) \\ &= p^{n-1}a_1 + q\sum_{k=0}^{n-2} p^k = p^{n-1}a_1 + \frac{1-p^{n-1}}{1-p} \cdot q \end{aligned}$$

is obtained and this result is consistent.

Before moving on to the next case, here is an example problem.

Exercise 5.2.3: Enoch's Problem

Find a_{101} if $a_n = 2a_{n-1} + 3$ for $n > 1$ and $a_1 = 2$.

Solution.

First, we can try substituting a_{n-1} in the given equation in terms of

a_{n-2} and continue the process until we reach a_1.

$$\begin{aligned} a_{101} &= 2a_{100} + 3 \\ &= 2(2a_{99} + 3) + 3 = 2^2 a_{99} + (2+1)3 \\ &\vdots \\ &= 2^{100} a_1 + 3 \sum_{k=0}^{99} 2^k \end{aligned}$$

Therefore, $a_{101} = 2^{101} + \frac{1-2^{100}}{1-2} \cdot 3 = 2^{101} - 3 + 3 \cdot 2^{100} = 2^{100}5 - 3$.

Continuing from this case, we have its extension where q is not a constant but a function.

IV. $\mathbf{a_n = ka_{n-1} + f(n)}$

This problem too can be solved with very similar methods from the previous, its just a little more unkind. Let's start with my favorite method.

Solution I.

In this solution, we will define a new term $b_n = \frac{a_n}{k^n}$. Consider the following equations.

$$\begin{aligned} \frac{a_n}{k^n} &= \frac{a_{n-1}}{k^{n-1}} + \frac{f(n)}{k^n} \\ b_n &= b_{n-1} + \frac{f(n)}{k^n} = b_{n-1} + g(n) \end{aligned}$$

This is the same as Case I and telescoping gives us

$$b_n = \sum_{k=1}^{n} g(k) + b_1 - g(1)$$

and rewriting this, we obtain the following.

$$\frac{a_n}{k^n} = \sum_{l=1}^{n} \frac{f(l)}{k^l} + \frac{a_1}{k} - \frac{f(1)}{k} = \sum_{l=1}^{n} \frac{f(l)}{k^l} + \frac{a_1 - f(1)}{k}$$

This method simply defines a new sequence to represent it into an easier problem.

Solution II.

The second method is very similar to the second solution for Case III. First, we will assume that there exists a function $F(n)$ such that the following equation holds.

$$a_n - F(n) = k(a_{n-1} - F(n-1))$$

Rewriting this, we can see that $a_n = k^{n-1}(a_1 - F(1)) + F(n)$. From here it depends on the problem, but it shouldn't be too difficult because we only need to find $F(n)$.

Before moving on to the next case, here is a practice problem.

Exercise 5.2.4: Enoch's Problem

Find a_n in terms of n if $a_n = 3a_{n-1} - 2n + 1$ and $a_1 = 2$.

(Video Solution)

Solution.

First, we can try to divide both sides by 3^n and let $b_n = \frac{a_n}{3^n}$.

$$\frac{a_n}{3^n} = \frac{a_{n-1}}{3^{n-1}} - \frac{2n-1}{3^n}$$
$$b_n = b_{n-1} - \frac{2n-1}{3^n}$$

Telescoping we obtain $b_n - b_1 = -\sum_{k=2}^{n} \frac{2k-1}{3^k}$. Let $S = \sum_{k=2}^{n} \frac{2k-1}{3^k}$ and consider the following equations for index-shifting.

$$S = \sum_{k=2}^{n} \frac{2k-1}{3^k}$$
$$\frac{S}{3} = \sum_{k=2}^{n} \frac{2k-1}{3^{k+1}} = \sum_{j=3}^{n+1} \frac{2j-3}{3^j}$$

$$\frac{2}{3}S = \sum_{k=2}^{n} \frac{2k-1}{3^k} - \left(\sum_{j=2}^{n} \frac{2j-3}{3^j} - \frac{1}{9} + \frac{2n-1}{3^{n+1}} \right)$$
$$= \sum_{k=2}^{n} \frac{2}{3^k} + \frac{1}{9} - \frac{2n-1}{3^{n+1}}$$

Therefore, $S = \sum_{k=2}^{n} \frac{1}{3^{k-1}} + \frac{1}{6} - \frac{2n-1}{2\cdot 3^n} = \frac{1}{3} \cdot \frac{1-\frac{1}{3^{n-1}}}{1-\frac{1}{3}} + \frac{1}{6} - \frac{2n-1}{2\cdot 3^n}$. Substituting,

$$\frac{a_n}{3^n} - \frac{2}{3} = -\left(\frac{1}{2} - \frac{1}{2\cdot 3^{n-1}} + \frac{1}{6} - \frac{2n-1}{2\cdot 3^n} \right)$$
$$\frac{a_n}{3^n} = -\frac{1}{2} + \frac{1}{2\cdot 3^{n-1}} - \frac{1}{6} + \frac{2n-1}{2\cdot 3^n} + \frac{2}{3}$$
$$= \frac{1}{2\cdot 3^{n-1}} + \frac{2n-1}{2\cdot 3^n} = \frac{1}{2\cdot 3^{n-1}} + \frac{n}{3^n} - \frac{1}{2\cdot 3^n}$$

Therefore, $a_n = n+1$.

Continuing from this case, here is the next one. From this case the problems could get more complicated, but I don't think they are necessarily harder because they build from the four cases that we discussed so far.

V. $\mathbf{a_n = \frac{pa_{n-1}}{qa_{n-1}+r}}$

This type of problem is relatively simple. To solve them, we can first take the reciprocal of both sides to obtain a more simple problem.

$$\frac{1}{a_n} = \frac{qa_{n-1}+r}{pa_{n-1}} = \frac{q}{p} + \frac{r}{pa_{n-1}}$$

Now substituting $b_n = \frac{1}{a_n}$, we obtain Case III! The next case is a little more complex, but is also quite common.

VI. $\mathbf{pa_{n+1} + qa_n + ra_{n-1} = 0}$

For this case, we can separate the cases into two cases. If the problem maker was kind, we could sometimes get $p+q+r=0$. In that case, we

can simply substitute $q = -p - r$ to obtain the following equation.

$$p(a_{n+1} - a_n) = r(a_n - a_{n-1})$$

Letting $b_n = a_n - a_{n-1}$, it is now trivial.

If the problem is not nice enough, which is the most likely case, we will get $p + q + r \neq 0$. This now becomes quite complicated. For this case, we can try dividing both sides by p. Notice that because $\frac{q}{p}$ and $\frac{r}{p}$ are constants, we can always find α and β such that the following equation holds with Vieta's formula.

$$a_{n+1} - (\alpha + \beta)a_n + \alpha\beta a_{n-1} = 0$$

From here, we can again split the equation into two cases depending on the values of α and β.

The first case is when $\alpha \neq \beta$. Rearranging the equation, we obtain the two distinct equations.

$$a_{n+1} - \alpha a_n = \beta(a_n - \alpha a_{n-1})$$
$$a_{n+1} - \beta a_n = \alpha(a_n - \beta a_{n-1})$$

Because the equations from a geometric sequence, we can telescope to obtain the following.

$$a_n - \alpha a_{n-1} = \beta^{n-1}(a_2 - \alpha a_1)$$
$$a_n - \beta a_{n-1} = \alpha^{n-1}(a_2 - \beta a_1)$$
$$\therefore a_{n-1} = \frac{1}{\beta - \alpha}(\beta^{n-1}(a_2 - \alpha a_1) - \alpha^{n-1}(a_2 - \beta a_1))$$

From here it is not difficult to notice that $a_n = \frac{1}{\beta-\alpha}(\beta^n(a_2 - \alpha a_1) - \alpha^n(a_2 - \beta a_1))$ and it won't be too difficult to find a_1 and a_2 because they are usually given by the problem.

The second case is when $\alpha = \beta$. This case is actually nicer because we

can simply substitute the values to obtain the following.

$$a_{n+1} - 2\alpha a_n + \alpha^2 a_{n-1} = 0$$
$$a_{n+1} - \alpha a_n = \alpha(a_n - \alpha a_{n-1})$$

Again, because this is a geometric sequence, we can telescope to obtain the following equation.

$$a_{n+1} - \alpha a_n = \alpha^{n-1}(\alpha_2 - \alpha a_1)$$

Continuing, we can divided both sides by α^{n+1} to telescope again and obtain the following.

$$\frac{a_{n+1}}{\alpha^{n+1}} - \frac{a_n}{\alpha^n} = \frac{a_2 - \alpha a_1}{\alpha^2}$$
$$\therefore \frac{a_n}{\alpha^n} = \frac{a_2 - \alpha a_1}{\alpha^2} \cdot (n-1) + \frac{a_1}{\alpha}$$

Therefore, $a_n = \alpha^{n-2}(n-1)(a_2 - \alpha a_1) + \alpha^{n-1} a_1$.

> The extended version of this would be when we have more than three variables like the following.
>
> $$p_0 a_n + p_1 a_{n-1} + \cdots + p_k a_{n-k} = 0$$
>
> In this case, assume that $a_n = r^n$ and solve the **characteristic equation**. If the roots $r_1, \cdots, r_k$ are all distinct, the solution would be $a_n = C_1 r_1^n + C_2 r_2^n + \cdots C_k r_k^n$. If any of the roots repeat, make them distinct by multiplying n. For example, if r_i repeats three times, we can make the roots into k distinct ones by having the $k-3$ roots and $r_i, nr_i, n^2 r_i$. The coefficients C_i should be easy to find by substitution because initial conditions are most likely evident from the problem.

There are few more cases left, and let's continue with Case VII.

VII. $\mathbf{a_n = pa_{n-1} + qb_{n-1}, b_n = ra_{n-1} + sb_{n-1}}$

This is the case when we are introduced two independent variables.

Intuitively, we should eliminate one variable to turn the equations into more simple forms discussed earlier and substitute back into one of the equations to obtain the other. Here is an example.

Exercise 5.2.5: Enoch's Problem

Given $a_n = 8a_{n-1} + 7b_{n-1}$ and $b_n = 2a_{n-1} + 3b_{n-1}$ with $a_1 = 5$ and $b_1 = 4$, find a_n and b_n in terms of n.

Solution.

For this problem, we could technically use linear algebra, but I think that kinda ruins the fun and elegance of the solution, so let's not use matrices. Instead, we could try eliminating b_n. Consider the following equations.

$$\begin{aligned} 7b_n &= a_{n+1} - 8a_n \\ 7b_n &= 14a_{n-1} + 21b_{n-1} = 14a_{n-1} + 3a_n - 24a_{n-1} \\ \therefore a_{n+1} - 8a_n &= 14a_{n-1} + 3a_n - 24a_{n-1} \end{aligned}$$

Rewriting the equation, we obtain $a_{n+1} - 11a_n + 10a_{n-1} = 0$. This is the previous case, but simpler! Let $S_n = a_{n+1} - a_n$. Substituting, we obtain $S_n = 10S_{n-1}$ and telescoping gives $S_n = 10^{n-1}S_1$. Moreover, because $S_1 = a_2 - a_1 = 40 + 28 - 5 = 63$, $a_{n+1} - a_n = 63 \cdot 10^{n-1}$. Telescoping again, we obtain

$$a_n - a_1 = 63 \sum_{k=2}^{n} 10^{n-2} = 63 \cdot \left(\frac{10^{n-1} - 1}{10 - 1} \right) = 7(10^{n-1} - 1)$$

and $a_n = 7 \cdot 10^{n-1} - 2$. Substituting to the first equation,

$$\begin{aligned} 7b_{n-1} = a_n - 8a_{n-1} &= 7 \cdot 10^{n-1} - 2 - 56 \cdot 10^{n-2} + 16 \\ &= 14 \cdot 10^{n-2} + 14 \end{aligned}$$

holds and $b_n = 2 \cdot 10^{n-1} + 2$.

This case was easier than the others! Continuing, here is Case VIII.

VIII. $a_n = \frac{pa_{n-1}+q}{ra_{n-1}+s}$

This is another case where we are introduced a fraction. But unlike Case V, you can notice that finding the reciprocal of both sides does not solve the problem. For this case, clever substitution and introducing a new sequence is the key to a solution. Here is an example to demonstrate it.

Exercise 5.2.6: WinnersKMO

Given a_1 and $a_n = \frac{a_{n-1}+2}{2a_{n-1}+1}$, find a_n [Win26].

Solution.

To solve this problem, let's first simplify the numerator.

$$a_n = \frac{\frac{1}{2}(2a_{n-1}+1)+\frac{3}{2}}{2a_{n-1}+1} = \frac{1}{2} + \frac{\frac{3}{2}}{2a_{n-1}+1}$$
$$2a_n - 1 = \frac{3}{2a_{n-1}+1}$$

Let $b_n = 2a_n + 1$. Substituting, we obtain $b_n - 2 = \frac{3}{b_{n-1}}$. Here is when we introduce new terms c_n. Let $b_n = \frac{c_{n+1}}{c_n}$ where $c_1 = 1$ and $c_2 = 5$. Substituting,

$$\frac{c_{n+1}}{c_n} - 2 = \frac{3c_{n-1}}{c_n}$$

and we obtain $c_{n+1} - 2c_n - 3c_{n-1} = 0$, and this is Case VI.

This case is not too good but not too bad because solving the characteristic equation is clean. Solving $x^2 - 2x - 3 = 0$, we get $x = 3$ and $x = -1$. Therefore, $c_n = A3^n + B(-1)^n$. Because $c_1 = 3A - B$, $c_2 = 9A + B$, and $b_1 = 5$, we can see that $\frac{9A+B}{3A-B} = 5$ and $6B = 6A$. Because $c_1 = 1$, $A = B = \frac{1}{2}$ and $c_n = \frac{3^n+(-1)^n}{2}$. Now we are done!

Substituting back to b_n, we obtain

$$b_n = \frac{\frac{3^{n+1}+(-1)^{n+1}}{2}}{\frac{3^n+(-1)^n}{2}} = \frac{3^{n+1}+(-1)^{n+1}}{3^n+(-1)^n}$$

and substituting it back to a_n gives us the following.

$$a_n = \frac{b_n - 1}{2} = \frac{\frac{3^{n+1}+(-1)^{n+1}}{3^n+(-1)^n} - 1}{2} = \frac{\frac{3^{n+1}+(-1)^{n+1}-3^n-(-1)^n}{3^n+(-1)^n}}{2}$$

Therefore $a_n = \frac{3^n-(-1)^n}{3^n+(-1)^n}$.

Continuing with this case, here is the final case for this note!

IX. Trigonometry in Recursion

I personally think this case is the hardest to see and apply because for some reason I just can't remember trigonometric properties properly. But fortunately, I think just some basic relations like double angle, half angle, and sine and cosine relationship are enough. However, the problem is that I think it is very difficult to see when to use trig for recursion. Here is an example from WinnersKMO [Win26] that I enjoyed.

Exercise 5.2.7: WinnersKMO

Given $a_1 > 0$ and $a_n = \sqrt{2 + a_{n-1}}$, find a_n [Win26].

Solution.

One interesting thing that we can notice from this question unlike the other problems is that we don't have our initial value set. To be honest, I don't think there is a better way to understand the solution presented here than memorizing patterns, so here we go.

First, recall that $\cos^2 \frac{\theta}{2} = \frac{1+\cos\theta}{2} = \frac{2+2\cos\theta}{4}$. Rearranging, we can see that the given equation resembles with the rearrangement $2\cos\theta = \sqrt{2+2\cos 2\theta}$. Following the pattern, let $a_n = 2\cos\theta_n$ where $\theta_n =$

$\frac{1}{2}\theta_{n-1}$. Telescoping, we see that $\theta_n = \frac{1}{2^{n-1}}\theta_1$ and $a_n = 2\cos\frac{\theta_1}{2^{n-1}}$. Substituting $a_1 = 2\cos\theta_1$, we obtain the following equation for $a_1 \leq 2$.

$$a_n = 2\cos\left(\frac{\cos^{-1}\left(\frac{a_1}{2}\right)}{2^{n-1}}\right)$$

Now that we have observed the case when $0 < a_1 \leq 2$, we now need to consider $a_1 > 2$.

For $a_1 > 2$, let $a_1 = \frac{2}{\sin 2\alpha}$. Recall that $\sin 2\alpha = 2\sin\alpha\cos\alpha$. Substituting, we get the following equations.

$$a_2 = \sqrt{2 + \frac{2}{\sin 2\alpha}} = \sqrt{2 + \frac{1}{\sin\alpha\cos\alpha}} = \frac{\sin\alpha + \cos\alpha}{\sqrt{\sin\alpha\cos\alpha}}$$
$$a_3 = \sqrt{2 + \frac{\sin\alpha + \cos\alpha}{\sqrt{\sin\alpha\cos\alpha}}} = \frac{\sqrt{\sin\alpha} + \sqrt{\cos\alpha}}{(\sin\alpha\cos\alpha)^{\frac{1}{4}}}$$

Repeating, we can see that

$$a_n = \frac{(\sin\alpha)^{\frac{1}{2^{n-2}}} + (\cos\alpha)^{\frac{1}{2^{n-2}}}}{(\sin\alpha\cos\alpha)^{\frac{1}{2^{n-1}}}} = \frac{(\sin\alpha)^p + (\cos\alpha)^p}{(\sin\alpha\cos\alpha)^{\frac{p}{2}}}$$
$$= (\tan\alpha)^{\frac{p}{2}} + (\cot\alpha)^{\frac{p}{2}} = (\tan\alpha)^{\frac{1}{2^{n-1}}} + (\cot\alpha)^{\frac{1}{2^{n-1}}}$$

where $p = \frac{1}{2^{n-2}}$. Moreover, because $\tan\alpha = \frac{\sin 2\alpha}{1+\cos 2\alpha}$, the following equations hold.

$$(\tan\alpha)^{\frac{1}{2^{n-1}}} = \left(\frac{\sin 2\alpha}{1+\cos 2\alpha}\right)^{\frac{1}{2^{n-1}}} = \left(\frac{\frac{2}{a_1}}{1+\sqrt{1-\frac{4}{a_1^2}}}\right)^{\frac{1}{2^{n-1}}}$$
$$= \left(\frac{2}{a_1+\sqrt{a_1^2-4}}\right)^{\frac{1}{2^{n-1}}} = \left(\frac{a_1-\sqrt{a_1^2-4}}{2}\right)^{\frac{1}{2^{n-1}}}$$
$$(\cot\alpha)^{\frac{1}{2^{n-1}}} = \left(\frac{1}{\frac{2}{a_1+\sqrt{a_1^2-4}}}\right)^{\frac{1}{2^{n-1}}} = \left(\frac{a_1+\sqrt{a_1^2-4}}{2}\right)^{\frac{1}{2^{n-1}}}$$

Therefore, $a_n = \left(\frac{a_1-\sqrt{a_1^2-4}}{2}\right)^{\frac{1}{2^{n-1}}} + \left(\frac{a_1+\sqrt{a_1^2-4}}{2}\right)^{\frac{1}{2^{n-1}}}$.

This problem had lots of trigonometric properties! Although this section was quite long I think they all follow general pattern of telescoping, using characteristic equations, and using clever substitution to make the problems easier. This is it for this section, and we will discuss about symmetric and cyclic sums in the next section.

Bounding and Monotonicity

§5.3 Symmetric and Cyclic Sums

5.3.1 Elementary Symmetric Polynomials

5.3.2 Newton's Identities

5.3.3 Clearing Denominators

§5.4 Practice Problems

Below are some practice problems with a subjective rating on difficulties. The solutions are included at the end of Math Olympiad part of LibreNotebook.

5.4.1. **2000 APMO Problem 1 (● ● ○)**
Find $S = \sum_{i=0}^{101} \frac{x_i^3}{1-3x_i+3x_i^2}$ for $x_i = \frac{i}{101}$.

Note 6

[G] Euclidean Geometry

Euclidean geometry in my opinion is the hardest section in olympiads despite being so widely used beyond olympiads and competitions. Many problems are solved by angle chasing, finding similar triangles, and only with elementary concepts. However, it is very difficult to utilize such concepts unless you have really good eyes, draws perfectly, or in the best condition. In this note, we will discuss more in-depth about the topics that many of us are familiar with geometry in competitions and gaining better eyes for euclidean geometry.

§6.1 Power of a Point

You might already be very familiar with this topic if you have math competition background. However, I wanted to revisit this topic as it is surprisingly common topic in math olympiad. However, the difference could be finding patterns and more advanced topics from power of a point. With that in mind, let's review some of the topics.

6.1.1 Definitions and Theorems

As always, let's get started with the definition.

Definition 6.1.1

The **power of a point** of P with respect to a circle ω represented as $P_\omega(P) = PO^2 - r^2$ where O and r the center and radius of ω respectively.

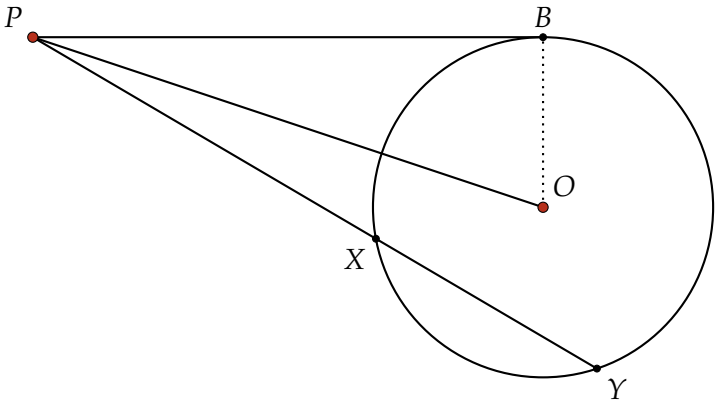
The power of a point is a numeric value that describes the relationship between the circle and the point. Before reviewing the relationship, let's first derive the formula in the definition. The theorem below is the heart of all concepts in this section.

Theorem 6.1.2: Power of a Point Theorem

Let P be a point and ω be a circle on the same space. If lines l_1 and l_2 that passes through P meets ω at A, B and C, D (not necessarily distinct) respectively, then $PA \cdot PB = PC \cdot PD$.

If you read the other notes, you may have noticed that I try my best to include proofs for all theorems included. However, let's skip the proof for now for the purpose of this note.

From the theorem above, we can deduce that $PX \cdot PY$, where X and Y are points on ω such that P, X, Y are collinear, is always consistent. With that in mind, consider the following diagram.



By the theorem, $P_\omega(P) = PX \cdot PY = PB \cdot PB$. Moreover by the defi-

nition of a line tangent to a circle, $PB \perp BO$ and $PO^2 = PB^2 + r^2$ by Pythagorean Theorem. Therefore, $P_\omega(P) = PO^2 - r^2$.

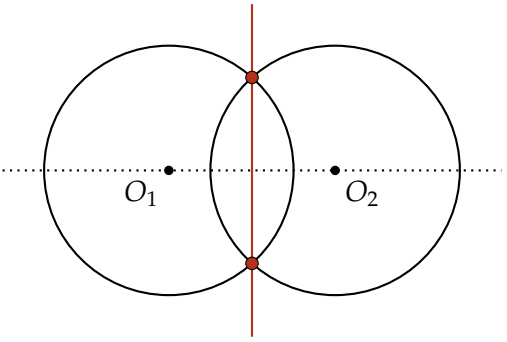
With the definition, we can derive trivial relationship between the power of a point and the location of the point. If $P_\omega(P) < 0$, then P is inside ω. If $P_\omega(P) = 0$, then P is on ω and P is outside ω if and only if $P_\omega(P) > 0$.

Continuing from the definition, we can derive another priceless definition that is one of the foundational concept in olympiad geometry.

Definition 6.1.3

The **radical axis** of two circles ω_1 and ω_2 with different center is the locus of P such that $P_{\omega_1}(P) = P_{\omega_2}(P)$.

Another way of thinking about this is that a point P is on the radical axis if and only if $O_1P^2 - r_1^2 = O_2P^2 - r_2^2$ by definition.



Continuing, there is another very properties of radical axis.

Theorem 6.1.4: Properties of Radical Axis

1. The radical axis is always a straight line.
2. The radical axis of ω_1 and ω_2 is always perpendicular to the line that pass through two origins of ω_1 and ω_2.

Let's first prove the first property.

Proof.

Let ω_1 be a circle defined as $(x-a_1)^2+(y-b_1)^2=r_1^2$. Similarly, let ω_2 be a circle $(x-a_2)^2+(y-b_2)^2=r_1^2$. Let $P(x,y)$ be any points such that $P_{\omega_1}(P)=P_{\omega_2}(P)$. Substituting, the following equation holds by definition.

$$(x-a_1)^2+(y-b_1)^2-r_1^2=(x-a_2)^2+(y-b_2)^2-r_2^2$$

Expanding and rearranging the equation,

$$\begin{aligned}&x^2-2a_1x+a_1^2+y^2-2b_1y+b_1^2-r_1^2\\=\;&x^2-2a_2x+a_2^2+y^2-2b_2y+b_2^2-r_2^2\\&(2a_2-2a_1)x+(2b_2-2b_1)y+C=0\end{aligned}$$

holds where C is the constant values. Notice that the locus of all (x,y) that satisfies the equation above is a linear function. Thus, the radical axis is always linear.

Continuing, we can prove the second property. There are few ways of proving this, but let's take an algebraic approach with our equation above.

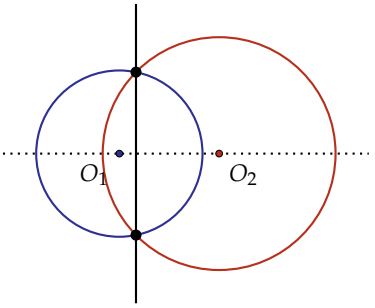
Proof.

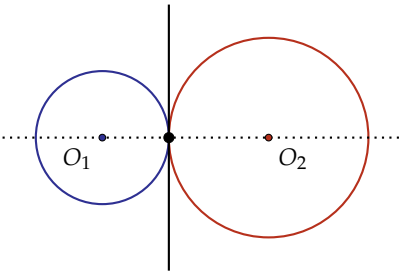
Continuing with the equation

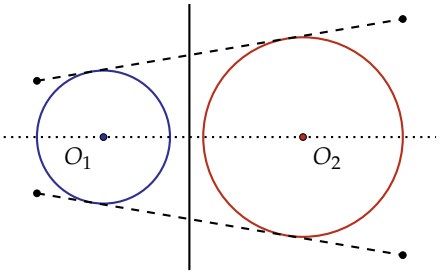
$$(2a_2-2a_1)x+(2b_2-2b_1)y+C=0$$

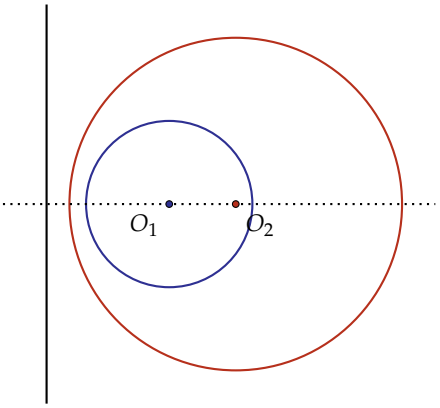
from the previous proof, it is evident that the slope of the line is $-\frac{2a_2-2a_1}{2b_2-2b_1}$. Continuing, the slope of the line that passes through the two centers of ω_1 and ω_2 is $\frac{b_2-b_1}{a_2-a_1}$. Because $-\frac{2a_2-2a_1}{2b_2-2b_1}\cdot\frac{b_2-b_1}{a_2-a_1}=-1$, the two lines are perpendicular to each other.

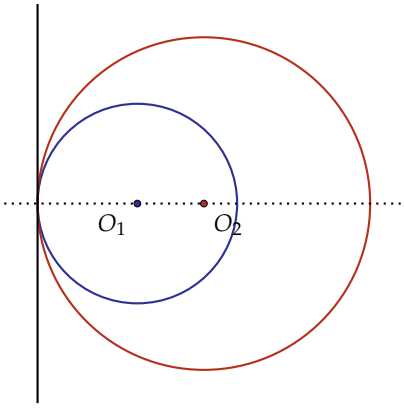
Before we move on to further theorems, lemmas, definitions, and problems, here are five special forms that makes our lives easier when graphing to solve problems.











As you can see from the diagrams above, the fact that the radical axis passes through the intersection(s) of ω_1 and ω_2 will come very handy when we solve problems. Also, because this knowledge is considered trivial and it is part of the definition, I believe the graders won't take points off for not proving why the radical axis passes through the intersection(s).

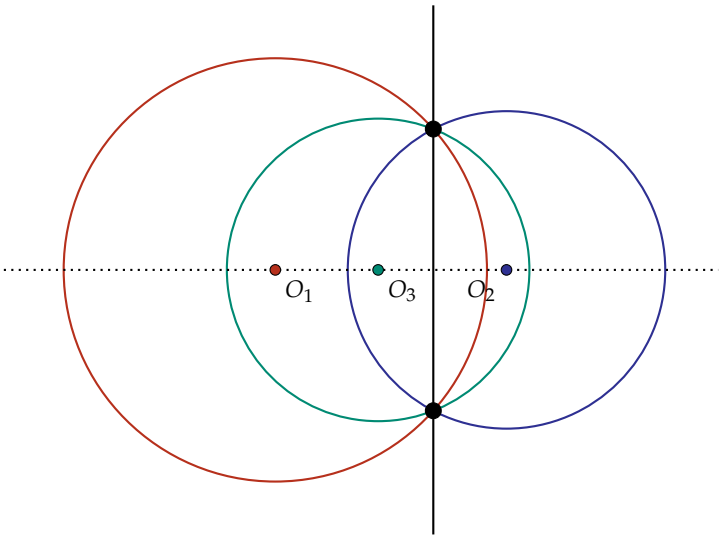
Also for the third diagram, one tip when drawing is that you can find the radical axis by drawing two lines, or segments, that are *tangent* to both circles. Then, the radical axis is the line that connects the centers of the two segments. Before moving on to common patterns and theorems,

here is one more definition to keep in mind.

Definition 6.1.5

Coaxial circles are circles that share the common pairwise radical axis.

Below is an example of three coaxial circles.



The three circles $\omega_1, \omega_2, \omega_3$ are coaxial because the radical axis for ω_1, ω_2, for ω_2, ω_3, and for ω_3, ω_1 are all equal. It is also worth noting that three circles in this case all passes through two common intersections. Coaxial circles are not limited to three circles, but most problems would be on three circles for olympiads.

With the definition in mind, we can observe some key properties of coaxial circles.

Theorem 6.1.6

The centers of coaxial circles $\omega_1, \omega_2, \omega_3$ are collinear.

Proof.

Let the centers of $\omega_1, \omega_2, \omega_3$ be O_1, O_2, O_3 and the radical axis be l. By definition, the radical axis of ω_1 and ω_2 is l and $O_1O_2 \perp l$. Similarly, $O_1O_3 \perp l$ since l is the radical axis of ω_1 and ω_3. Because there exists exactly one line perpendicular to l that passes through O_1, O_1O_2 and O_1O_3 lines are the same line and O_1, O_2, O_3 are collinear.

This theorem can look somewhat trivial. However, finding such conditions and looking for coaxial circles can come very handy.

Never underestimate! Facts like $OA = OB$ for a circle with center O and $P_\omega(A) = P_\omega(B)$ can be the key to the solution. Whenever you see three circles, circles defined by reflection, collinearity of their centers, and/or the intersection of three circles, we should be reminded of coaxial circles and suspect it as the key at least once.

Continuing, we can see the relationship between the three radical axes with three distinct circles. One thing to keep in mind if you are taking Korean Mathematical Olympiad, then the following theorem is known as Monge's Theorem in addition to one that is related to homothety. However let's call this Radical Axis Concurrence Theorem since I think that is more universal.

Theorem 6.1.7: Radical Axis Concurrence Theorem

Given three distinct circles, the pairwise radical axes of the circles are concurrent.

Proof.

Let $\omega_1, \omega_2, \omega_3$ be three distinct circles with the centers O_1, O_2, O_3. Let l_1, l_2, l_3 be the radical axes of the pairs O_1, O_2 and O_2, O_3 and O_3, O_1 respectively. Let K be the intersection of l_1 and l_2. By definition, $P_{\omega_1}(K) = P_{\omega_2}(K)$ and $P_{\omega_2}(K) = P_{\omega_3}(K)$. Therefore, $P_{\omega_1}(K) = P_{\omega_3}(K)$ and $K \in l_3$. Therefore, l_1, l_2, l_3 are concurrent.

This is it! Continuing from the theorem, we can define the following

term.

Definition 6.1.8

The **radical center** of three distinct circles is the concurrent point of pairwise radical axes.

Last, but not least, here is a very powerful theorem on coaxial circles. Honestly, this theorem made me suspect power of a point every time I see more than two circles on a problem.

Theorem 6.1.9

Let ω_1 and ω_2 be two distinct circles and A, B, C be three distinct points. If

$$\frac{P_{\omega_1}(A)}{P_{\omega_2}(A)} = \frac{P_{\omega_1}(B)}{P_{\omega_2}(B)} = \frac{P_{\omega_1}(C)}{P_{\omega_2}(C)}$$

holds, then either the circumcircle of $\triangle ABC$, ω_1, and ω_2 are coaxial circles or A, B, C is the radical axis of ω_1 and ω_2.

Proof.

First, notice that for any circles $x^2 + y^2 + ax + by + c = 0$ and a point $A(x_1, y_1)$, $P_\omega(A) = (x_1 - p)^2 + (y_2 - q)^2 - r^2 = x_1^2 + y_1^2 + ax_1 + by_1 + c$. Moreover, if $\frac{P_{\omega_1}(A)}{P_{\omega_2}(A)} = \frac{P_{\omega_1}(B)}{P_{\omega_2}(B)} = \frac{P_{\omega_1}(C)}{P_{\omega_2}(C)} = k$, then $P_{\omega_1}(X) - kP_{\omega_2}(X) = 0$ for $X = A, B, C$.

Let ω_1 be defined as $x^2 + y^2 + a_1x + b_1y + c_1 = 0$ and ω_2 as $x^2 + y^2 + a_2x + b_2y + c_2 = 0$. Rewriting the expression $P_{\omega_1}(X) - kP_{\omega_2}(X)$, the following equations are obtained.

$$\begin{aligned} P_{\omega_1}(X) - kP_{\omega_2}(X) &= x^2 + y^2 + a_1x + b_1y + c_1 \\ &\qquad - k(x^2 + y^2 + a_2x + b_2y + c_2) \\ &= (1-k)x^2 + (1-k)y^2 + (a_1 - ka_2)x \\ &\qquad + (b_1 - kb_2)y + (c_1 - kc_2) \end{aligned}$$

Define ω_3 as $(1-k)x^2 + (1-k)y^2 + (a_1 - ka_2)x + (b_1 - kb_2)y +$

$(c_1 - kc_2) = 0$ for $k \neq 1$. Then, $A, B, C \in \omega_3$ and ω_3 is the circumcircle of $\triangle ABC$. Moreover by construction, the power of any point X satisfy the following equation.

$$P_{\omega_3}(X) = \frac{P_{\omega_1}(X) - kP_{\omega_2}(X)}{1-k}$$

Therefore, if X lies on the radical axis of ω_3 and ω_1, i.e. $P_{\omega_3}(X) = P_{\omega_1}(X)$, then $(1-k)P_{\omega_1}(X) = P_{\omega_1}(X) - kP_{\omega_2}(X)$ and $P_{\omega_1}(X) = P_{\omega_2}(X)$. In other words, $P_{\omega_3}(X) = P_{\omega_1}(X)$ if and only if $P_{\omega_1}(X) = P_{\omega_2}(X)$ and $\omega_1, \omega_2, \omega_3$ are coaxial circles.

If $k = 1$, then $P_{\omega_1}(A) = P_{\omega_2}(A)$, $P_{\omega_1}(B) = P_{\omega_2}(B)$, and $P_{\omega_1}(C) = P_{\omega_2}(C)$ holds and A, B, C lies on the radical axis of ω_1 and ω_2. Because radical axis is a straight line, A, B, C are collinear and the points form the radical axis.

With the definition and theorems in mind, let's solve some practice problems to see how they are hidden.

6.1.2 Brain Teasers

The first problem is quite nostalgic and elegant problem that I wished to share.

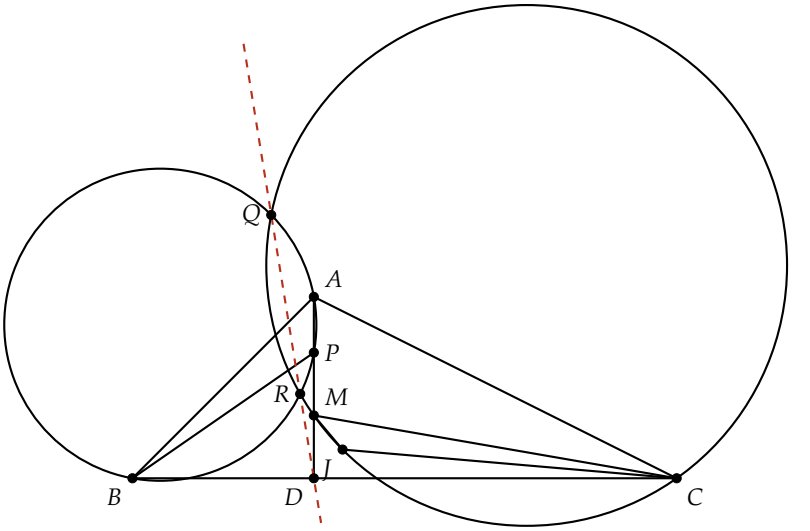
Exercise 6.1.10: 2025 Korean MO Problem 5 (● ○ ○)

Let D be a point on the segment BC of triangle ABC such that $\overline{AD} = \overline{BD} = \frac{1}{2}\overline{CD}$. Let $P(\neq A, D)$ be a point on segment AD, M be the midpoint of PD, and J be the incenter of triangle MDC. If the circumcircles of $\triangle MJC$ and $\triangle ABP$ intersect at two distinct points Q and R, show that D, Q, R are collinear.

(Video Solution)

Solution.

First, let's draw the diagram.

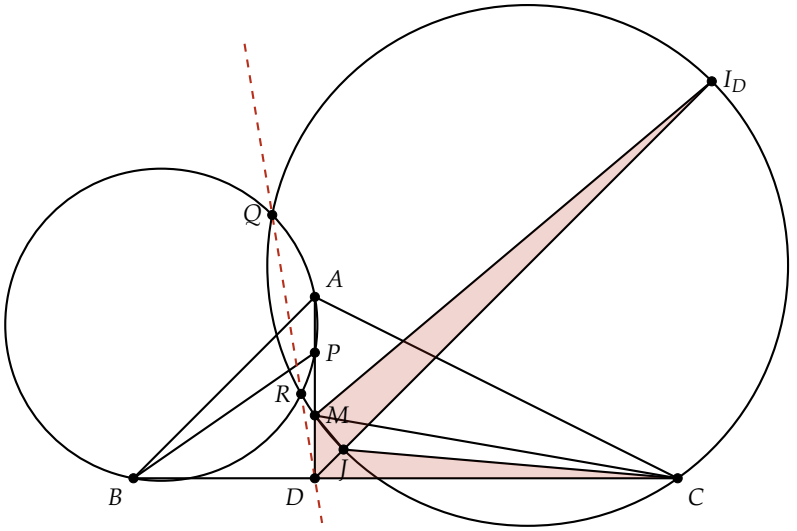


Notice that because Q and R both lies of the circles ω_1 (left) and ω_2 (right), it suffices to show that D lies on the radical axis of ω_1 and ω_2. In other words, we must show that $P_{\omega_1}(D) = P_{\omega_2}(D)$. Consider the following equation.

$$P_{\omega_1}(D) = \overline{AD} \cdot \overline{PD} = \overline{BD} \cdot (2 \cdot \overline{MD}) = 2\overline{BD} \cdot \overline{MD} = \overline{DC} \cdot \overline{MD}$$

Therefore, it suffices to show that $P_{\omega_2}(D) = \overline{DC} \cdot \overline{MD}$.

To find P_{ω_2}, we can use another classic pattern that we can see from the problem. We see the point D, the incenter J and the circumcircle of $\triangle MJC$. Naturally trying to implement incenter/excenter lemma here, we can draw and find excenter of $\triangle MDC$ at D.



Notice that $\angle MDJ = \angle JDC$ by construction. Similarly, $\angle MI_DJ = \angle MCJ = \angle JCD$. Thus $\triangle MDI_D$ and $\triangle JDC$ are similar. Because $MD : JD = DI_D : DC$, $P_{\omega_2}(D) = DJ \cdot DI_D = MD \cdot DC$.

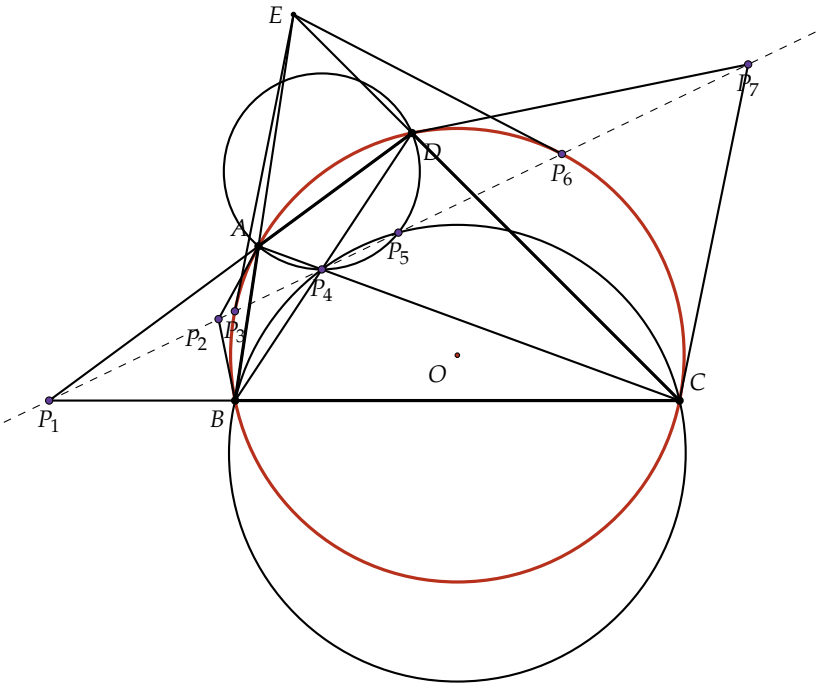
Never overestimate! I remember trying my best not to use power of a point during the exam because the morning problems were hard that day, and ended up wasting lots of time in the afternoon exam. I panicked, and learned a lesson.

§6.2 Six Collinear Points

Six collinear points, also known as seven collinear points, is one of the fundamental topics to study in euclidean geometry for olympiads. I don't think this topic is popular internationally, but I can assure that every Korean students preparing for KMO knows this diagram.

This diagram is quite special because seven points lie on the same line, and that is definitely not common. There are also many different ways to prove this, but let's not use Pascal and other topics, but only foundational topics because it feels intuitive and more like an olympiad problem. But before proving, here is the diagram. Also, if you prefer video explanation

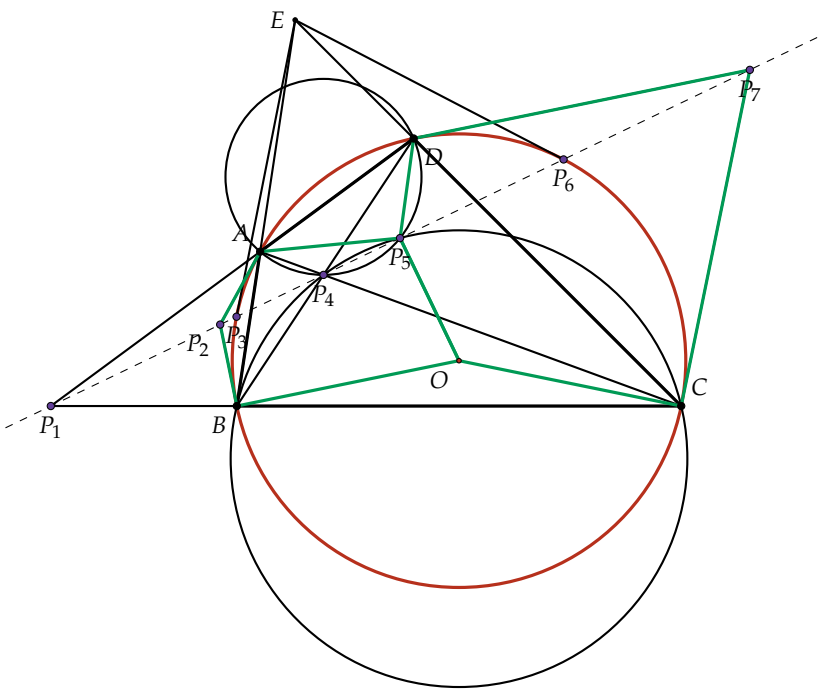
over text, the video is available as well.



Here, quadrilateral $ABCD$ is inscribed in the circle with center O, and E and P_1 are the points obtained by finding the intersection between AB and CD and intersection between AD and BC respectively. P_2 and P_7 are obtained by finding the intersection between the lines tangent to the circle with center O from A and B and C and D respectively. Finally, P_4 is the intersection of AC and BD and P_5 is the intersection of circumcircles of $\triangle P_4AD$ and $\triangle P_4BC$.

I know this sounds a lot at first, but it is actually quite easy to memorize and notice in problems involving cyclic quadrilaterals. Now let's prove this thing!

> *Proof.*
>
> Consider the annotated diagram below.



First, notice that P_1 is the radical center of three circles. Therefore, P_1, P_4, P_5 are collinear. Continuing, notice that quadrilaterals AP_2BO and $DOCP_7$ are cyclic by construction. Moreover, the following equations hold.

$$\begin{aligned}
\angle AP_3B &= \angle AP_3P_4 + \angle P_4P_3B \\
&= \angle ADB + \angle ABC = \angle AOB \\
\angle DP_5C &= 360 - \angle DP_5P_4 - \angle P_4P_5C \\
&= 180 - \angle DP_5P_4 + 180 - \angle P_4P_5C \\
&= \angle DAC + \angle DBC = \angle DOC
\end{aligned}$$

Therefore, (AP_2BOP_5) and (DP_5OCP_7).

Continuing, consider the following equations.

$$\angle P_4P_5O = \angle P_4P_5B + \angle BP_5O \qquad = \angle ACB + \angle BAO = 90^\circ$$

Therefore, $P_4P_5 \perp P_5O$ and E, P_5, O are collinear since E is the radical center of $\omega_1, \omega_4, \omega_5$ where ω_4 and ω_5 are the circle that circumscribe the pentagons. Because OP_2 and OP_7 are the diameter of ω_4 and ω_5, $P_1, P_2, P_4, P_5, P_7 \in l$.

Using power of a point theorem, notice that the following equations hold.

$$\begin{aligned} EP_3^2 &= ED \cdot EC = EP_5 \cdot EO \\ \therefore EP_3 : EO &= EP_5 : EP_3 \\ EP_6^2 &= EA \cdot EB = EP_5 \cdot EO \\ \therefore EP_6 : EO &= EP_5 : EP_6 \end{aligned}$$

Therefore, $\triangle EP_5P_3 \sim \triangle EP_3O$ and $\triangle EP_5P_6 \sim \triangle EP_6O$. In other words, $P_3P_5 \perp P_5O$ and $P_6P_5 \perp P_5O$. Thus, the seven points are collinear.

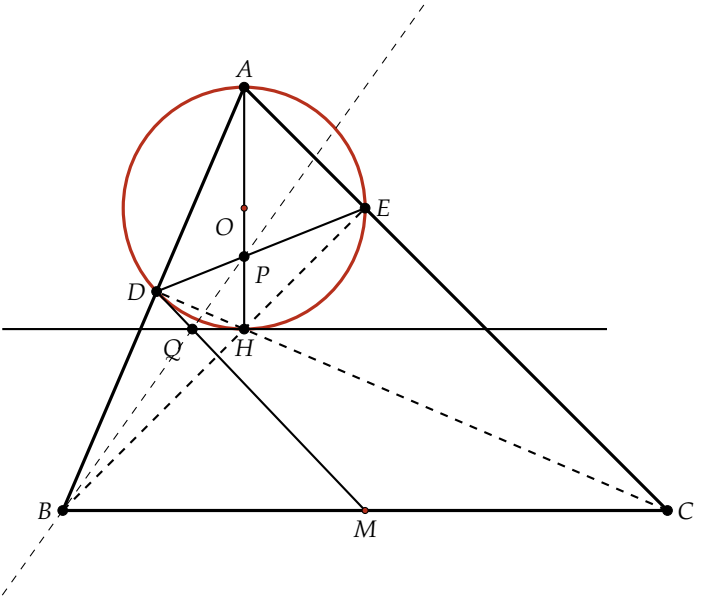
This could look a lot, but when you think about it, you can observe this in most of the problems involving cyclic quadrilaterals. Many times, three or four of these seven points act as a breakthrough of the problems. Here is a quick example.

Exercise 6.2.1: 2020 KMO Problem 2

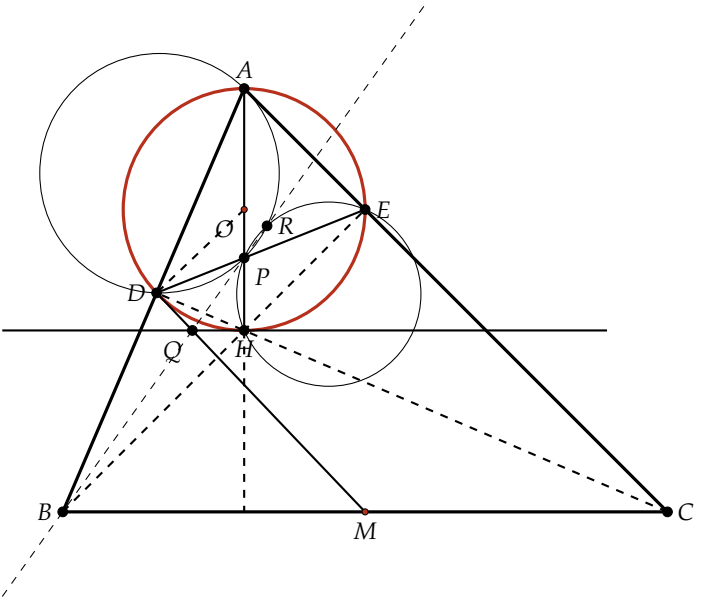
Given an acute triangle ABC, let H and M be the orthocenter and the midpoint of BC respectively. Let D and E be the intersection between the circle with diameter AH and sides AB and AC respectively. Let P be the intersection of AH and DE and let Q be the intersection of DM and the line perpendicular to AH that passes through H. Show that P, Q, B are collinear.

Solution.

First, consider the following diagram.



Looking at the diagram with respect to cyclic quadrilateral $ADHE$, the six collinear points can be found. Consider the annotated diagram.



From here, notice that with $\angle ADH = \alpha$, we have $\alpha = \angle DEH = \angle DHQ = \angle DCB$. Moreover, $\angle DMB = 2\alpha = \angle DOH$. In other words, $OD \perp DM$ and B, P, Q are three of the seven points.

Continuing, consider the new point R. Notice that because $\angle DRP + \angle PRH = \angle DAH + \angle DEH = \angle DOH$, $(DORHQ)$. Moreover, notice that $\angle ORP = \angle ORD + \angle DRP = \angle OHD + \angle DAH = 90^\circ$. Because $\angle QRO = 90^\circ$, Q, P, R are collinear and B, Q, P, R are also collinear by radical axis. Therefore, P, Q, B are collinear.

This problem is now easy with six collinear points! There are many different ways to solve this problem such as using homothety and pole and polar, but I think this method is the most straightforward. This is it for this section! In the next section, we will discuss another very important topic isogonal conjugate.

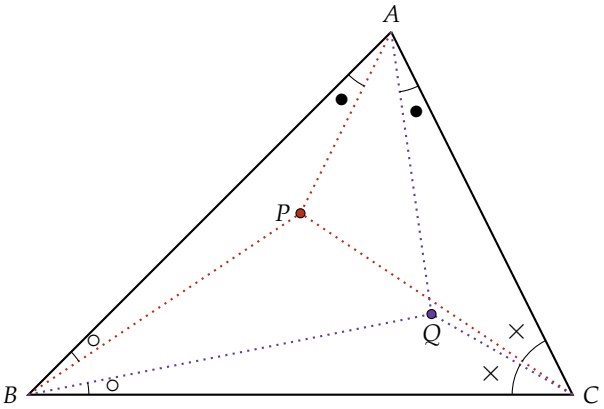
§6.3 Isogonal Conjugate

The isogonal conjugate of a point P with respect to a triangle is an interesting well-defined point that can be expanded to different techniques and patterns. Although the name suggests the definition of the point, here is the definition.

Definition 6.3.1

The **isogonal conjugate** of a point P with respect to a triangle ABC is the intersection of the reflections of AP, BP, CP with respect to the angle bisectors at A, B, and C.

To better understand this topic, consider the following diagram.



Note that the lines AQ, BQ, CQ are the **isogonal lines** of AP, BP, CP respectively. The point Q is the isogonal conjugate of P with respect to $\triangle ABC$. Now this point Q is well-defined. To show that the isogonal conjugate is always a well-defined point, we can use many different methods, but here are two of my favorite ones.

For the first proof, we will use Angle-Ceva Theorem. I honestly am not sure the standard term for the theorem because it is not widely known, but in Korea we call it as Angle-Ceva Theorem and I also saw some English forms like the angle form of Ceva' Theorem and trigonometric form of Ceva's Theorem.

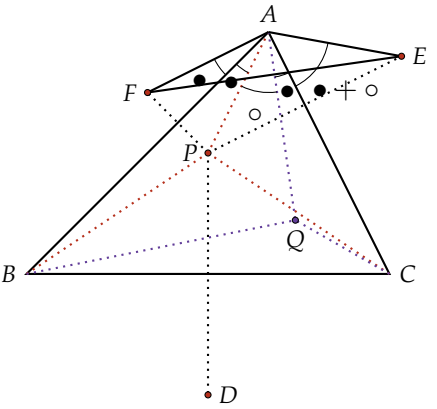
> The theorem states that the following equation holds for some point P in triangle ABC.
>
> $$\frac{\sin\angle ABP}{\sin\angle PBC}\cdot\frac{\sin\angle CAP}{\sin\angle PAB}\cdot\frac{\sin\angle BCP}{\sin\angle PCA}=1$$

Note that by construction, the following equation holds.

$$\begin{aligned}&\frac{\sin\angle CBQ}{\sin\angle QBA}\cdot\frac{\sin\angle ACQ}{\sin\angle QCB}\cdot\frac{\sin\angle BAQ}{\sin\angle QAC}\\=&\frac{\sin\angle ABP}{\sin\angle PBC}\cdot\frac{\sin\angle BCP}{\sin\angle PCA}\cdot\frac{\sin\angle CAP}{\sin\angle PAB}=1\end{aligned}$$

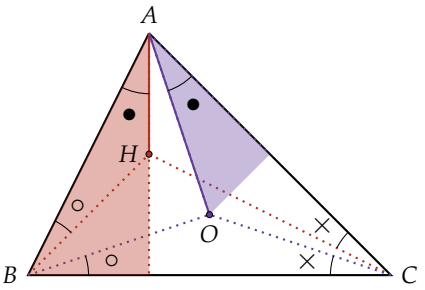
Therefore by reverse Angle-Ceva, Q is well-defined.

Another way of proving the existence is by using reflections.



Because we do not yet know if there exists point Q, let l_A, l_B, l_C be the respective isogonal lines. Notice that by construction, because $AF = AP = AE$, l_A is the perpendicular bisector of FE. Analogously, it is self-evident that l_B and l_C are perpendicular bisector of FD and ED respectively. In other words, the point of intersection of l_A, l_B, l_C is the center of the circumcircle of $\triangle DEF$ and the center of such circle and the points D, E, F always exist. Thus, Q is well-defined.

I personally find this method more interesting because we can still prove it without prior knowledge. Continuing with the proofs, consider the following diagram for a special case.



Notice that because $\angle AEF = \angle ABC$, $\angle BAH = \angle OAC$ by AA similarity. Similarly, the analogous argument applies to the other angles.

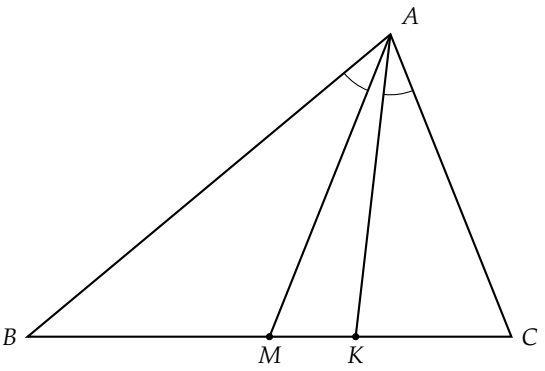
Points H and O are isogonal conjugates.

Before we wrap up this section, let's discuss about a special case called symmedian.

Definition 6.3.2

The **symmedian** from a vertex of a triangle is the isogonal line of the line that passes through the vertex and the midpoint of its opposite side.

Consider the following the digram.



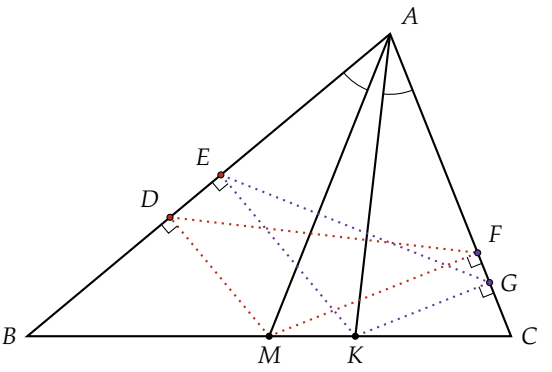
Here, M is the midpoint of side BC and AK is the symmedian from A. With this definition, we can construct the following theorem.

Theorem 6.3.3

Given a triangle ABC with midpoint M of side BC, AK is the symmedian if and only if $\frac{CK}{BK} = \frac{AC^2}{AB^2}$.

Proof.

First, consider the following diagram where D, E are points on AB drawn perpendicularly from M and K and F, G are point on AC drawn perpendicularly from M and K.



Notice that because $\square ADMF$ and $\square AEKG$ are cyclic, $\angle DMF = \angle EKG$. Moreover by construction, because AK is the symmedian from A, $\angle DFM = \angle KEG$. Therefore, $\triangle DMF \sim \triangle GKE$ and $\frac{GK}{KE} = \frac{DM}{MF}$.

Continuing, notice that because $\frac{1}{2}MD \cdot AB = \frac{1}{2}MF \cdot AC$, $\frac{MD}{MG} = \frac{AC}{AB}$. In other words, $\frac{GK}{KE} = \frac{AC}{AB}$ and $\frac{[\triangle AKC]}{[\triangle ABK]} = \frac{AC^2}{AB^2}$. Moreover, because the entire process is reversible, the theorem holds.

The concept of isogonal conjugate appears surprisingly a lot when we discuss more advanced topics like harmonic range and spiral similarity. This is it for this section! Let's discuss homothety in the next section before getting into exciting topics like harmonic range and spiral similarity.

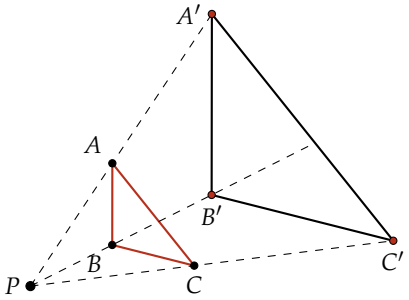
§6.4 Homothety

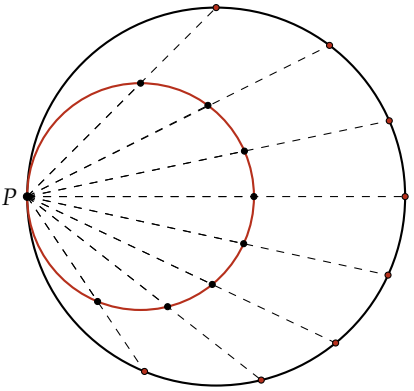
Homothety is a type of linear transformation that concerns dilation. Although it is a projective transformation, we consider this as a topic in Euclidean geometry. In this section, we are going to review important topics and properties in homothety.

Definition 6.4.1

Homothety is a linear dilation of a point A from a point P with factor k such that $k\vec{AP} = \vec{A'P}$ where A' is the **image** of A.

For a more visual explanation, consider the following diagrams that are both homothetic to center P.





As you can see from the diagram above, the homothetic center, a point, and its image are collinear. Also, a figure and its image is also similar.

> This can come especially handy when we have to prove collinearity of three points because if a triangle is an image of the other triangle, then the homothetic center and the circumcenters of the triangles are collinear.

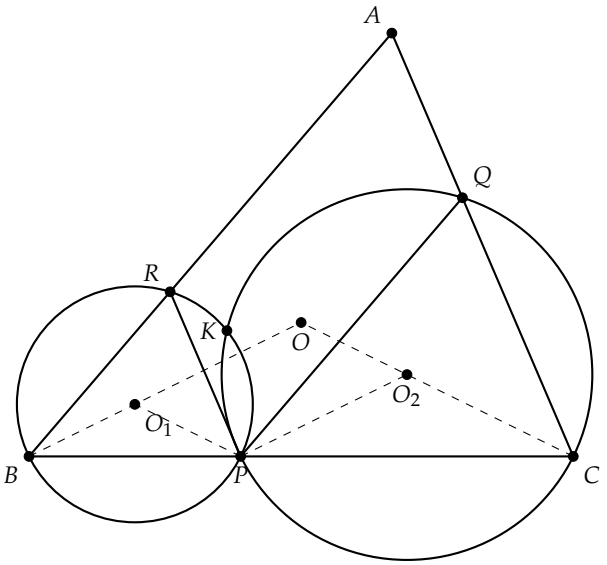
I will skip the proof since it is quite straightforward using similar triangles. Here is an example problem using the collinearity of the three points.

Exercise 6.4.2: 2013 KMO Problem 1

Given a triangle ABC, let P be a point on the side BC and define points Q and R as the intersections of the lines parallel to AC and AB that passes through P with AB and AC respectively. Let O, O_1, O_2 be the circumcenters of triangles ABC, BPR, PCQ respectively. If $K \neq P$ is the intersection of the circumcircles of $\triangle BPR$ and $\triangle PCQ$, show that $OO_1 = KO_2$.

Solution.

First, consider the following diagram.



First, notice that because $KO_2 = PO_2$, it suffices to show that quadrilateral OO_1PO_2 is a parallelogram. Let $\angle BAC = \alpha$. Notice that $\angle BO_1P = \angle PO_2C = \angle BOC = 2\alpha$. Therefore, it suffices to show that B, O_1, O and C, O_2, O are collinear. By construction, $\triangle BRP$ and $\triangle BAC$ and $\triangle CQP$ and $\triangle CAB$ are homothetic from B and C respectively. Thus, $OO_1 = KO_2$.

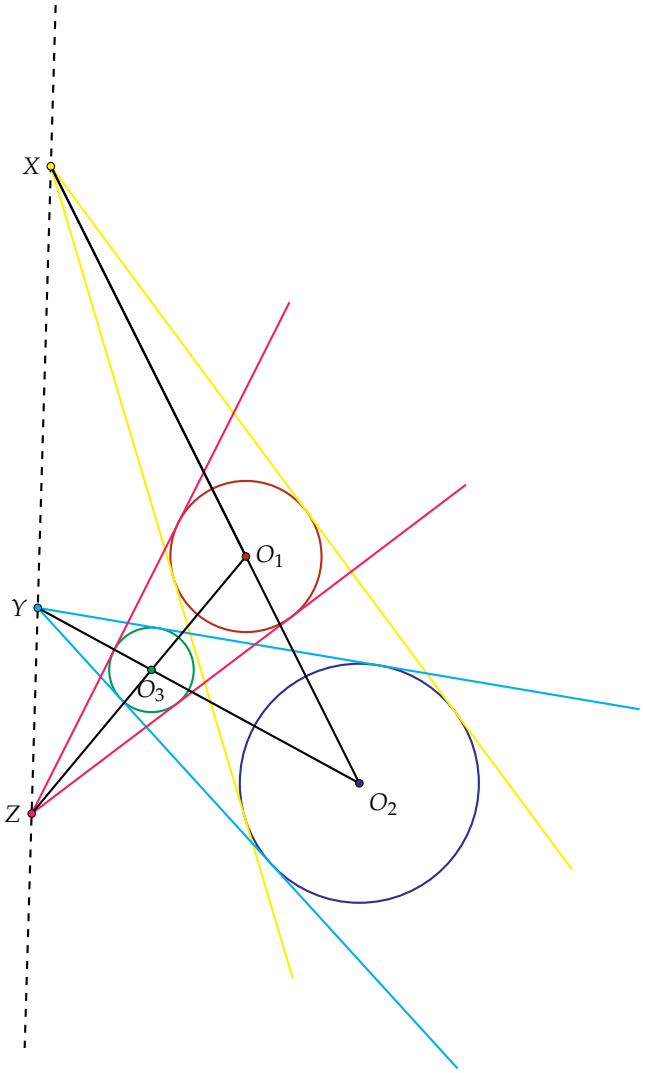
With this basic idea in mind, let's discuss an important theorem.

Theorem 6.4.3: Monge's Theorem

The pairwise external similitude centers of three distinct circles are collinear.

Proof.

First, consider the following diagram.



From the diagram above, it is evident that X, Y, Z are collinear if and only if Menelaus' Theorem holds for $\triangle O_1O_2O_3$. In other words, it suffices to show that the following equation holds.

$$\frac{O_2X}{O_1X} \cdot \frac{O_3Y}{O_2Y} \cdot \frac{O_1Z}{O_3Z} = 1$$

Note that $\frac{O_2X}{O_1X} = \frac{R_2}{R_1}$ by similar triangle. Similarly, $\frac{O_3Y}{O_2Y} = \frac{R_3}{R_2}$ and $\frac{O_1Z}{O_3Z} = \frac{R_1}{R_3}$. Because $\frac{R_2}{R_1} \cdot \frac{R_3}{R_2} \cdot \frac{R_1}{R_3} = 1$, the theorem holds.

Homothety in my opinion very popular and powerful tool because it gives the best conditions like parallel sides and similarity. This is it for this section, and we will discuss more on different points and circles of a triangle in the next section.

§6.5 Points and Circles of a Triangle

There are so many important points circles and circles in triangles. Knowing relationship between them can significantly help in solving many problems. Before we get into the first content, let's clear up some symbols. To be honest, I just don't think it would be efficient to define each points, so I will just mention them here. We will be using O for circumcenter, I for incenter, H for orthocenter, G for centroid, and I_A, I_B, I_C for excenters unless otherwise stated. With that in mind, let's get started with Euler line and related theorem!

6.5.1 Servoir's Theorem and Euler Line

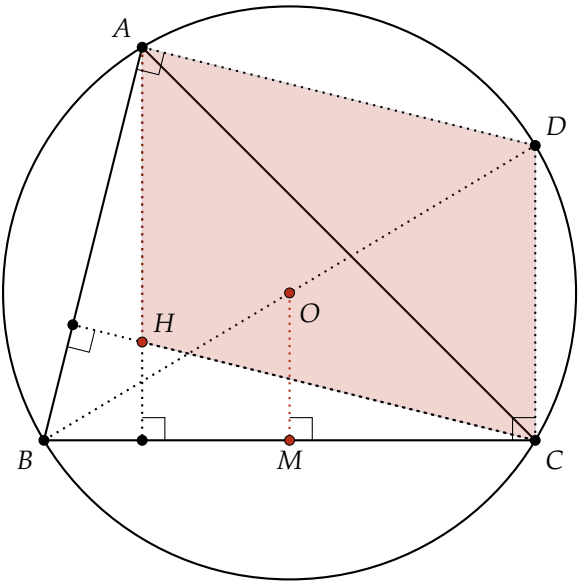
As a warm-up, let's start with Servoir's Theorem and Euler Line. Fun fact, when you search Servoir's Theorem online, you won't probably get the theorem that we will discuss. I am not sure if we have an English name for this theorem or if it is even named, but this is actually very useful and Koreans call this Servoir's Theorem.

Theorem 6.5.1: Servoir's Theorem

For triangle ABC and midpoint M on side BC, $AH = 2OM$.

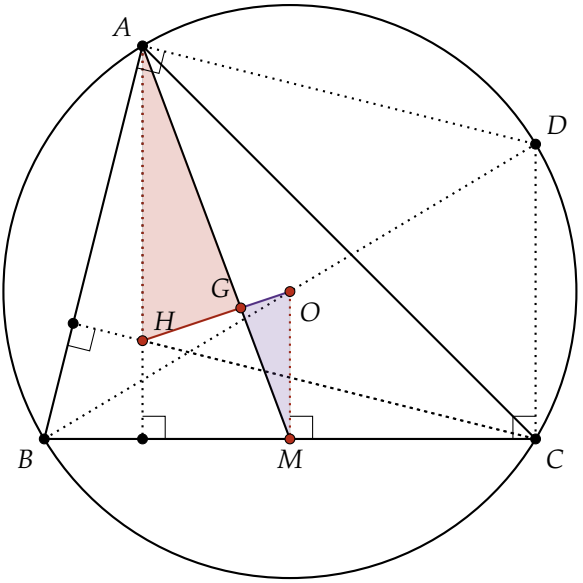
Proof.

First, consider the following diagram.



From the diagram above, it is evident that lines $\overline{AH}$ and $\overline{DC}$ are parallel. Similarly, $\overline{CH}$ and $\overline{AD}$ are parallel. Therefore, quadrilateral $AHCD$ is a parallelogram and $AH = CD = 2OM$.

Continuing from the diagram above, we could also consider G of $\triangle ABC$.



Consider triangles AHG and MOG. Notice that $\angle(HAG) = \angle(OMG)$ since $\overline{AH} \parallel \overline{OM}$. Moreover by Servoir's Theorem, $AH : OM = 2 : 1$ holds and $AG : GM = 2 : 1$ is true by definition. Therefore, $\triangle AHG$ and $\triangle MOG$ are similar and $\angle(AGH) = \angle(MGO)$. In other words, H, G, O are collinear, and we have a special name for the line!

Definition 6.5.2

The **Euler Line** of a triangle ABC is the line that contains H, G, O.

As you can see, Euler line exists for all triangle except equilateral triangle. The fact that H, G, O is collinear will come very handy when solving problems!

6.5.2 Nine-Point Circle

Note 7

[G] Projective Geometry

Projective geometry, unlike euclidean geometry, relies on the relationship between objects and their projections. One way I like to think about it is the study of shades. There are important topics that cannot be solved without prior knowledge, so let's dive into it!

§7.1 Foundations

Before we get into some important theorems and topics, lets discuss some of its foundational properties to get the gut feeling for future topics.

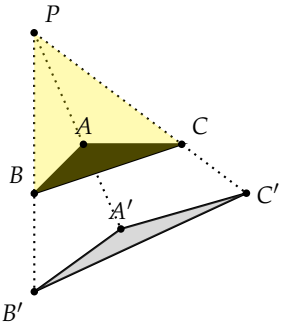


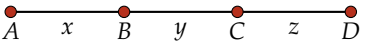
Figure 7.1

First, consider the two, not necessarily similar, triangles and a point P. From the diagram on the left, we can think of it as having a light source P that shines $\triangle ABC$. There, we get our outcome $\triangle A'B'C'$, and we say that A', B', C' are corresponding points of A, B, C respectively. Similarly, we could state that $\overline{A'B'}, \overline{B'C'}, \overline{C'A'}$ are

corresponding sides of $\overline{AB}, \overline{BC}, \overline{CA}$ respectively.

7.1.1 Cross Ratio

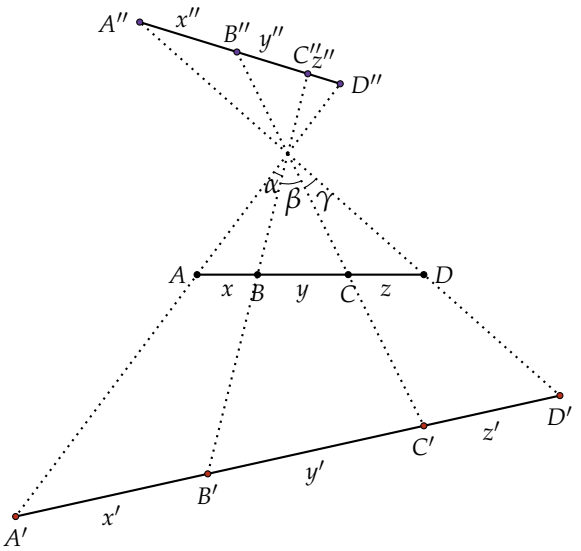
Cross ratio is a very important topic that lays foundations to many contents that we are going to discuss from now so it is very important to fully understand them! First, consider the following diagram with four collinear points A, B, C, D.



From the diagram above, we could define a new term.

> The **cross-ratio** of A, B, C, D is defined as the following.
>
> $$(A, B; C, D) = \frac{(x+y)(y+z)}{y(x+y+z)}$$

This is very important because if we project these points with respect to another line by a point say P, then the cross ratio of the resulting points is the same as the original value. Consider the following diagram for an illustration.



In this diagram, we can see that the following equation holds.

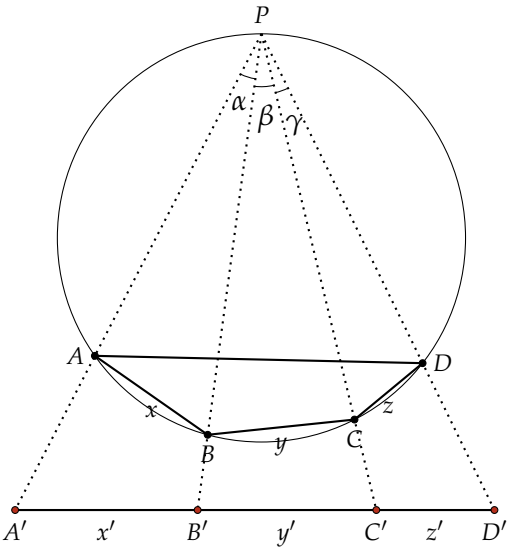
$$(A, B; C, D) = (A', B'; C', D') = (A'', B''; C'', D'')$$

To prove this, let's use length area ratio. Consider the following equations.

$$\begin{aligned}(A, B; C, D) &= \frac{(x+y)(y+z)}{y(x+y+z)} = \frac{[\triangle PAC] \cdot [\triangle PBD]}{[\triangle PBC] \cdot [\triangle PAD]} \\ &= \frac{PA \cdot PC \cdot \sin(\alpha+\beta) \cdot PB \cdot PD \cdot \sin(\beta+\gamma)}{PB \cdot PC \cdot \sin\beta \cdot PA \cdot PD \cdot \sin(\alpha+\beta+\gamma)} \\ &= \frac{\sin(\alpha+\beta) \cdot \sin(\beta+\gamma)}{\sin\beta \cdot \sin(\alpha+\beta+\gamma)} \\ &= \frac{[\triangle PA'C'] \cdot [\triangle PB'D']}{[\triangle PB'C'] \cdot [\triangle PA'D']} = (A', B'; C', D')\end{aligned}$$

The same argument applies for the top one, or you could just think of it as the rotated version.

Continuing with this, we can implement this idea to four points on a circle. I think this is one of the foundational topic that could act as the breakthrough.



From this diagram, we will use the Law of Sines to show that $(A, B; C, D) = (A', B'; C', D')$. First, notice that $(A', B'; C', D') = \frac{\sin(\alpha+\beta)\cdot\sin(\beta+\gamma)}{\sin\beta\cdot\sin(\alpha+\beta+\gamma)}$ from our previous observation. Moreover, the following equations hold.

$$\begin{aligned}
\frac{\sin(\alpha+\beta)\cdot\sin(\beta+\gamma)}{\sin\beta\cdot\sin(\alpha+\beta+\gamma)} &= \frac{\frac{\sin(\alpha+\beta)}{AC}\cdot\frac{\sin(\beta+\gamma)}{BD}}{\frac{\sin\beta}{BC}\cdot\frac{\sin(\alpha+\beta+\gamma)}{AD}}\cdot\frac{AC\cdot BD}{BC\cdot AD} \\
&= \frac{\frac{\sin\angle PDA}{PA}\cdot\frac{\sin\angle PAD}{PD}}{\frac{\sin\alpha}{AB}\cdot\frac{\sin\angle PAD}{PD}}\cdot\frac{AC\cdot BD}{BC\cdot AD} \\
&= \frac{\frac{\sin\angle PDA}{PA}}{\frac{\sin\alpha}{AB}}\cdot\frac{AC\cdot BD}{BC\cdot AD} \\
&= \frac{AC\cdot BD}{BC\cdot AD} = (A, B; C, D)
\end{aligned}$$

Therefore, the cross ratio is consistent if A, B, C, D lies on a circle any they are projected to a line from another point on the circle. Continuing, we can define a new term.

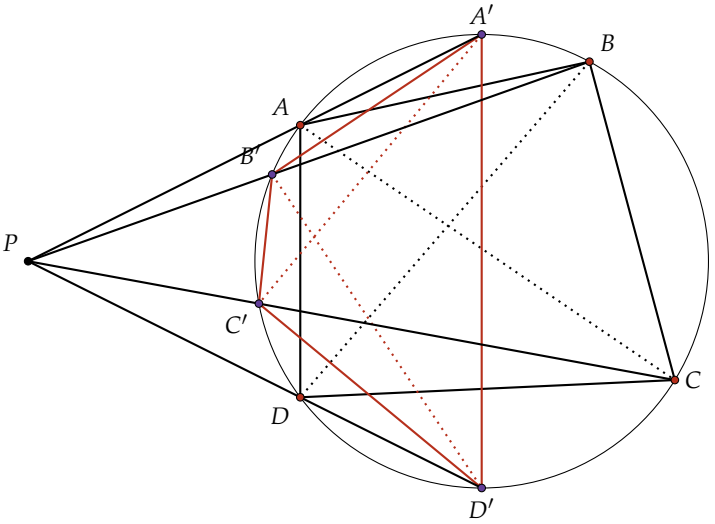
Definition 7.1.1

A **harmonic quadrilateral** $ABCD$ is a cyclic quadrilateral such that $(A, B; C, D) = -1$.

Another way of understanding harmonic quadrilateral is a cyclic quadrilateral with same product of the lengths of opposite edges are equal. It is pretty obvious using Ptolemy and the definition of cross ratio.

Next one is even more interesting. It is about projection of of a cyclic quadrilateral on to the inscribing circle.

In the diagram below, $(A, B; C, D) = (A', B'; C', D')$.



To prove this, we can use similar triangles. First, through simple angle chasing, it is evident that the following triangles are similar through AA similarity.

$$\triangle PAC \sim \triangle PC'A'$$
$$\triangle PDB \sim \triangle PB'D'$$
$$\triangle PAD \sim \triangle PD'A'$$
$$\triangle PB'C' \sim \triangle PCB$$

Using the length ratio of the similar triangles, we obtain following equations.

$$\frac{AC}{A'C'} = \frac{PA}{PC'} = \frac{PC}{PA'}$$
$$\frac{BD}{B'D'} = \frac{PD}{PB'} = \frac{PB}{PD'}$$
$$\frac{A'D'}{AD} = \frac{PD'}{PA} = \frac{PA'}{PD}$$
$$\frac{B'C'}{BC} = \frac{PB'}{PC} = \frac{PC'}{PB}$$

Multiplying the equations,

$$\frac{AC}{A'C'} \cdot \frac{BD}{B'D'} \cdot \frac{A'D'}{AD} \cdot \frac{B'C'}{BC} = \sqrt{\frac{PA}{PC'} \cdot \frac{PC}{PA'} \cdot \frac{PD}{PB'} \cdot \frac{PB}{PD'} \cdot \frac{PD'}{PA} \cdot \frac{PA'}{PD} \cdot \frac{PB'}{PC} \cdot \frac{PC'}{PB}} = 1$$

is obtained with $\frac{AC \cdot BD}{AD \cdot BC} = \frac{A'C' \cdot B'D'}{A'D' \cdot B'C'}$, and $(A, B; C, D) = (A', B'; C', D')$.

Before we wrap-up this section, consider the following theorem. This theorem comes especially handy when you have to draw a symmedian because many olympiads don't allow protractor.

Theorem 7.1.2

Given a harmonic quadrilateral $ABCD$, BD is the symmedian from B of $\triangle ABC$.

Proof.

First, notice that by Theorem 6.3.3, it suffices to show that $\frac{AK}{CK} = \frac{BA^2}{BC^2}$. Consider the following equations.

$$\frac{AK}{CK} = \frac{[\triangle ABD}{[\triangle BCD]} = \frac{\frac{1}{2}AB \cdot AD \cdot \sin \angle BAD}{\frac{1}{2}BC \cdot CD \cdot \sin \angle BCD} = \frac{AB \cdot AD}{BC \cdot CD} = \frac{AB}{BC} \cdot \frac{AB}{BC} = \frac{BA^2}{BC^2}$$

Therefore, the theorem holds.

With these concepts in mind, let's discuss some fundamental theorems in projective geometry.

§7.2 Fundamental Theorems

There are many foundational topics and theorems to discuss. Before we get into more advanced topics, let's discuss key theorems like Pascal's theorem and Desargues' theorem.

7.2.1 Pascal's Theorem

Let's first get started with Pascal's theorem! Surprisingly, Pascal's theorem is considered a topic in projective geometry even if it holds in Euclidean plane for some case because parallel pairs deal with point at infinity. Here is the statement.

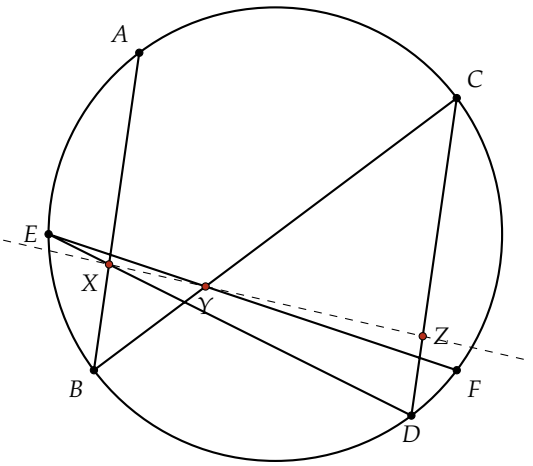
Theorem 7.2.1: Pascal's Theorem

Given six points A, B, C, D, E, F on a conic section, if $X = AB \cap DE$, $Y = CD \cap FA$, and $Z = EF \cap BC$, then X, Y, Z are collinear, whose line is known as **Pascal line**.

We stated here as conic section, but we can just consider it as a circle because majority of the problems deal with circle. Also, one tip for memorizing the definition of X, Y, Z is the following.

$$\begin{array}{cc|cc|cc} A & B & C & D & E & F \\ D & E & F & A & B & C \end{array}$$

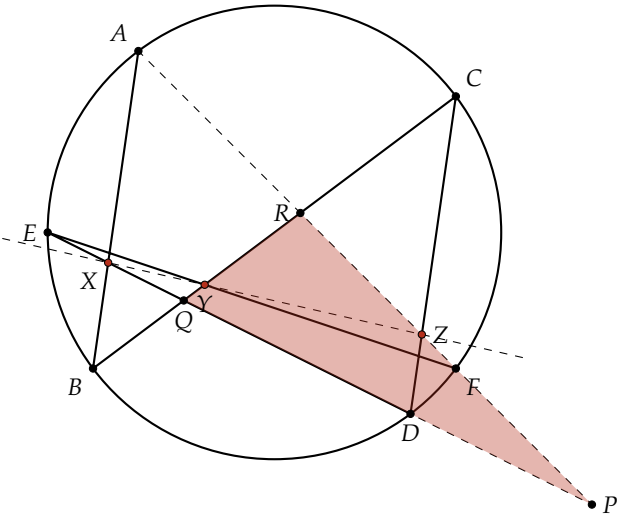
Sometimes it is easy to mess up like $AD \cap BE$, but remembering AB helped for me. Also, there are few different diagrams when you search Pascal's theorem depending on the orientation of the hexagon, but here is the famous and clean diagram.



There are many different ways to prove this theorem, and I think the most popular method is using Menelaus. However, there are also many different interesting ways of proving this, so let's prove Pascal's theorem using two different ways.

Proof.

The first proof involves using Menelaus three times. First, consider the new points P, Q, R.



From the triangle PQR, the following equations can be obtained with Menelaus' theorem.

$$\frac{PA}{AR} \cdot \frac{RB}{BQ} \cdot \frac{QX}{XP} = 1$$
$$\frac{PZ}{ZR} \cdot \frac{RC}{CQ} \cdot \frac{QD}{DP} = 1$$
$$\frac{EQ}{PE} \cdot \frac{YR}{QY} \cdot \frac{FP}{RF} = 1$$

Moreover by Power of a Point theorem, the following equations

hold.

$$PD \cdot PE = PF \cdot PA$$
$$BQ \cdot QC = EQ \cdot QD$$
$$BR \cdot RC = AR \cdot RF$$

Multiplying the equations,

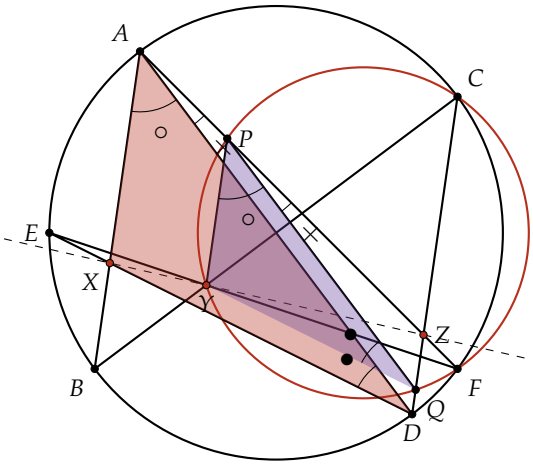
$$\frac{QX}{XP} \cdot \frac{PZ}{ZR} \cdot \frac{YR}{QY} = 1$$

is obtained and X, Y, Z are collinear by Menelaus' theorem.

The next one is less popular, but also an interesting proof.

Proof.

First, consider the following annotated diagram.



First, notice that $\angle FPQ = \angle DCF = \angle DAF$. Therefore, $AD \parallel PQ$. Similarly, $\angle XAD = \angle BCD = \angle YPQ$ and $\angle ADX \angle AFE = \angle PTQ$. Therefore, $\triangle AXD \sim \triangle PYQ$. Moreover, because Z is the center of homothety, X, Y, Z lie on the same line.

Pascal's theorem is arguably one of more popular and foundational

theorem in olympiad geometry. With the idea in mind, here is a practice problem.

Exercise 7.2.2

PASCAL PROBLEM

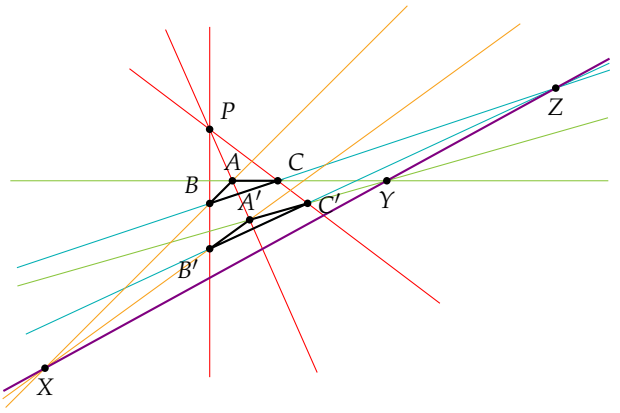
7.2.2 Desargues' Theorem

Continuing from Pascal's theorem, here is Desargues' theorem.

Theorem 7.2.3: Desargues' Theorem

Two triangles are axially perspective if and only if they are centrally perspective. Two triangles are considered axially perspective if there exists the axis of perspectivity and they are centrally perspective if there exists the center of perspectivity.

For simpler explanation, consider the diagram below, which is simply the extension of Figure 7.1.



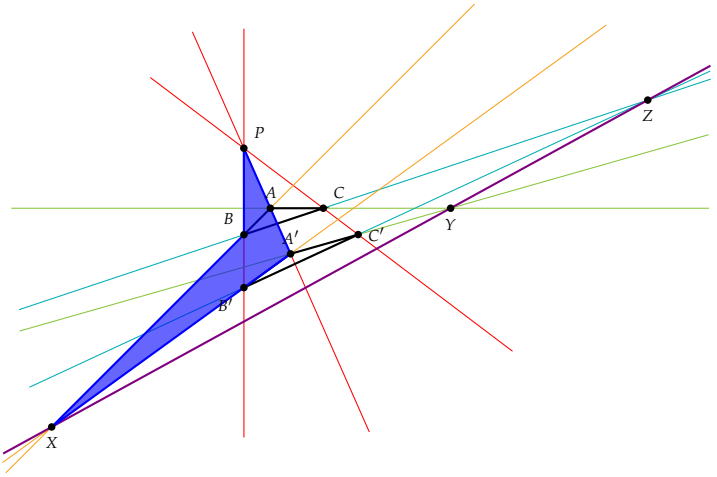
Point P, where AA', BB', CC' concur, is the center of perspectivity. Line XZ, where X, Y, Z are collinear, is the axis of perspectivity in the diagram above. If the pairs of corresponding lines are all parallel, it is said that the lines meet at a point of infinity. However, we normally don't discuss about that case.

The theorem is stating that if there exists a point P such that $P = AA' \cap BB' \cap CC'$, then the points $X = AB \cap A'B'$, $Y = BC \cap B'C'$, and $Z = CA \cap C'A'$ are collinear. With this in mind, let's try to prove the theorem.

Proof.

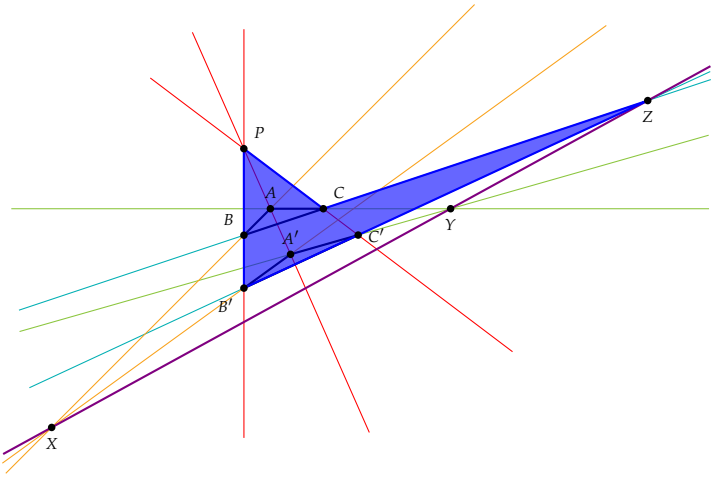
The proof of the theorem involves the use of Menelaus' theorem three times [Pam23]. First, let's prove that two triangles are axially perspective if they are centrally perspective.

Assuming that $\triangle ABC$ and $\triangle A'B'C'$ are perspective from point P, Menelaus' Theorem can be applied.



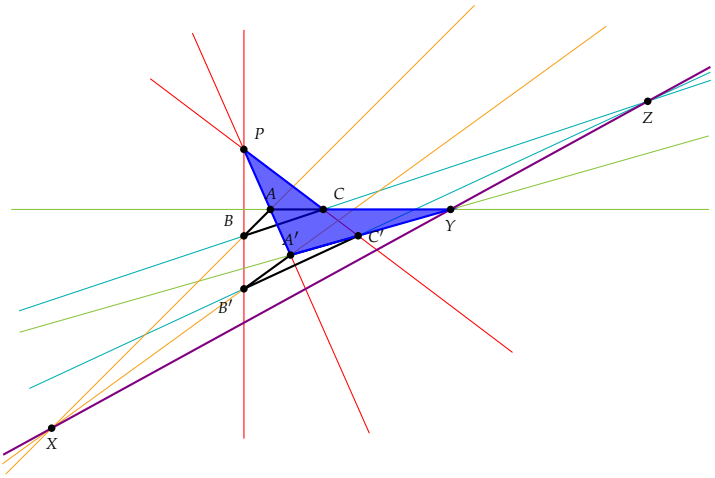
Consider $\triangle PA'B'$ with transverse line XB. Using Menelaus' Theorem, the following equation could be derived.

$$\frac{XA'}{XB'} \cdot \frac{BB'}{BP} \cdot \frac{AP}{AA'} = 1$$



Similarly, consider $\triangle PB'C'$ and transverse line ZC.

$$\frac{ZB'}{ZC'} \cdot \frac{CC'}{CP} \cdot \frac{BP}{BB'} = 1$$



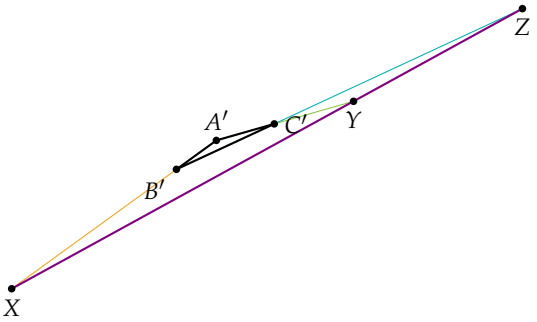
With $\triangle PC'A'$ and line YC,

$$\frac{YA'}{YC'} \cdot \frac{CC'}{CP} \cdot \frac{AP}{AA'} = 1$$
$$\frac{YC'}{YA'} \cdot \frac{CP}{CC'} \cdot \frac{AA'}{AP} = 1$$

Multiplying the equations,

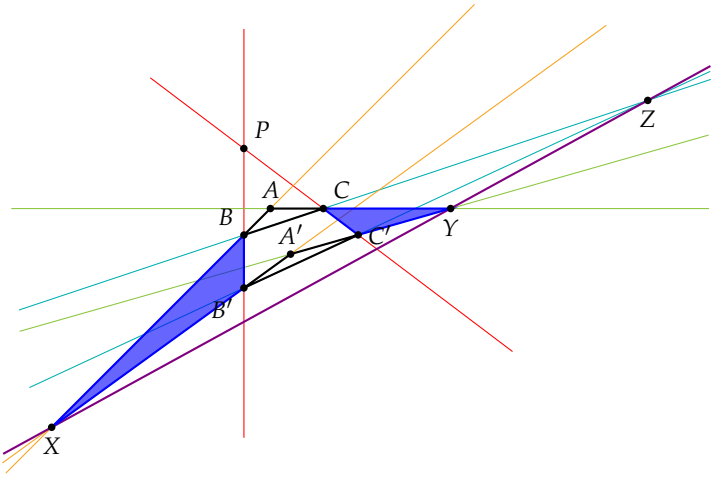
$$\left(\frac{XA'}{XB'}\cdot\frac{BB'}{BP}\cdot\frac{AP}{AA'}\right)\left(\frac{ZB'}{ZC'}\cdot\frac{CC'}{CP}\cdot\frac{BP}{BB'}\right)\left(\frac{YC'}{YA'}\cdot\frac{CP}{CC'}\cdot\frac{AA'}{AP}\right)=1$$
$$\frac{XA'}{XB'}\cdot\frac{ZB'}{ZC'}\cdot\frac{YC'}{YA'}=1$$

is obtained. Continuing, consider the following diagrams.



By converse of Menelaus' Theorem, it is evident that X, Y, and Z are collinear.

Now that the first statement for if and only if condition is proven, the converse must also be proven. Consider the diagram below where the same construction method is used. Note that if AA', BB', and CC' concur is yet to be known.



Notice that $\triangle XBB'$ and $\triangle YCC'$ are in perspective from point Z. Moreover, the first half of the proof manifests that P, A, and A' must be collinear. By construction, P is on line BB' and CC'. Moreover, it is also on the line AA'. Thus, $\triangle ABC$ and $\triangle A'B'C'$ are centrally perspective if they are axially perspective.

This is it for the proof! Before we move on to the next problem, let's solve a quick example problem.

Exercise 7.2.4

D is a point on $\overline{BC}$ in $\triangle ABC$. Let I_1, I_2 be the incenter of $\triangle ABD$ and $\triangle ACD$ respectively. Moreover, let I_3 and I_4 be ex-centers in respect to $\angle BAD$ and $\angle CAD$ respectively. Show that $\overline{I_1I_2}$, $\overline{I_3I_4}$, and $\overline{BC}$ intersect at one point.

(Video Solution)

Proof.

First, because I_1 and I_2 are the angle bisectors of $\angle ABD$ and $\angle ACD$ respectively, $I = BI_1 \cap CI_2$ is the incenter of $\triangle ABC$. Similarly, we know that $I' = BI_3 \cap CI_4$ is the ex-center of $\triangle ABC$ in respect that $\angle BAC$. Therefore A, I, and I' are collinear.

With similar idea, we could prove that A, I_1, I_3 are collinear as well as A, I_2, I_4. In other words, I_1I_3 and I_2I_4 intersect at point A. Therefore, $BI_1 \cap CI_2$, $I_1I_3 \cap I_2I_4$, and $I_3B \cap I_4C$ are collinear in $\triangle I_1BI_3$ and $\triangle I_2CI_4$. By Desargues' theorem, $\overline{I_1I_2}$, $\overline{I_3I_4}$, and $\overline{BC}$ concurs.

Note 8

[N] Diophantine Equations

Diophantine equation refers to equations with integer coefficient whose interests are integer solutions. For instance, equations like $ax + by = c$, $x^2 - ny^2 = 1$, and $x^n + y^n = z^n$ whose interests lie only on integer solutions are Diophantine equations. Diophantine equation has very wide range in difficulties starting from finding Pythagorean triples to famous theorems like Fermat's Last Theorem. In this note, we will discuss foundational topics on Diophantine equations to common techniques like proof by infinite descent and Vieta jumping.

§8.1 Primitive Pythagorean Triples

Problems on Pythagorean triples are probably one of the first problems that most of us solve in number theory, and yet remains as one of the dominant theorem. Maybe this is slightly biased towards Korean MO, but Pythagorean triples are very common and worth noting. As always, let's start with foundational properties.

Theorem 8.1.1

For a primitive Pythagorean triple (a, b, c), there exists a pair of positive integers $m > n$ such that the following equations hold.

$$\{a, b\} = \{m^2 - n^2, 2mn\}, \quad c = m^2 + n^2$$

Before we prove this theorem, let's clarify few terms. A primitive Pythagorean triple (a, b, c) means triples (a, b, c) such that $(a, b) = (b, c) = (c, a) = 1$. Also, we cannot definitively say that $a = m^2 - n^2$ and $b = 2mn$ because it depends on which term is odd and even. Indeed, we have opposite parities for a and b. I mean we can technically say $a = m^2 - n^2$ and $b = 2mn$ without loss of generality, but let's keep our theorem this way to highlight the parity. Now here is the famous proof!

Proof.

Without loss of generality, let $2 \nmid a$. By construction, the following congruences hold.

$$a^2 \equiv 1 \pmod 4, \quad b^2 \equiv 0 \pmod 4$$
$$c^2 \equiv a^2 + b^2 \equiv 1 \pmod 4$$

Therefore, c is odd. Let $a = 2a' + 1$ and $c = 2c' + 1$. Then, the following congruences hold.

$$(2a' + 1)^2 + b^2 \equiv (2c' + 1)$$
$$4a'^2 + 4a' + 1 + b^2 \equiv 4c'^2 + 4c' + 1 \pmod 8$$
$$b^2 \equiv 4c'^2 + 4c' - 4a'^2 - 4a' \equiv 4(c'^2 + c' - a'^2 - a') \equiv 0 \pmod 8$$

Thus $4 \mid b$. Continuing, the following equation holds by definition.

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

Let $d = (c + a, c - a)$. Note that $2 \mid d$ since $2 \nmid a, c$. Moreover, $d \mid (c + a) + (c - a) = 2c$ and $d \mid (c + a) - (c - a) = 2a$. Because

$(a, c) = 1$, $d = 2$. In other words, $c + a = 2m^2$ and $c - a = 2n^2$ for some coprime positive integers m, n. Substituting, $b = 2mn$. Thus, there exists a pair of positive integers $m > n$ such that $\{a, b\} = \{m^2 - n^2, 2mn\}$ and $c = m^2 + n^2$.

Honestly, this is pretty much it for the section. There isn't lots of topics to cover for primitive Pythagorean triple, but its applications are very worth noting. We will discuss more on the application of the theorem as we discuss infinite descent, but before we conclude this section, let's solve a practice problem to see how it is used in problems.

When using this theorem, it is very important to consider the condition stated that $(a, b) = (b, c) = (c, a) = 1$. Moreover, it is important to consider the parity of a and b before assigning respective values $m^2 - n^2$ and $2mn$. Lastly, but most importantly, remember that $(m, n) = 1$ and they are opposite parity! The last condition is very easy to miss (at least for me) and used very often when we apply this theorem recursively.

Below is an application that seems obvious, but is actually way difficult than it looks. For some reasons, I got lost very easily when learning applications of the theorem and any topics in NT, and I keeping the conditions of given and new variables in mind really helped me.

Exercise 8.1.2: 2020 Korean MO Problem 4

Find a pair of positive coprime integers (a, b) other than $(41, 12)$ such that $a^2 - 5b^2$ and $a^2 + 5b^2$ are both perfect squares.

This solution is largely based on a solution by WinnersKMO.

Solution.

First, let (x, y) be another coprime pair such that $x^2 - 5y^2 = m^2$ and $x^2 + 5y^2 = n^2$ for some integers m and n. Rearranging the equation, $2x^2 = m^2 + n^2$ and $10y^2 = n^2 - m^2$ hold. Starting with the first

equation, the following equation is obtained.

$$x^2 = \frac{m^2+n^2}{2} = \left(\frac{m+n}{2}\right)^2 + \left(\frac{m-n}{2}\right)^2$$

From the equation, assume that x, $\frac{m+n}{2}$, and $\frac{m-n}{2}$ are pairwise coprime. Note that we are allowed to set this condition since we just need to find any x and y. By Theorem 8.1.1, there exists coprime integers u, v with opposite parity such that $x = u^2 + v^2$, $\frac{m-n}{2} = u^2 - v^2$, and $\frac{m+n}{2} = 2uv$. Again, we are allowed to set parities since we are finding any x and y.

Continuing, the second equation can be modified.

$$\begin{aligned} 10y^2 &= 4 \cdot \frac{n+m}{2} \cdot \frac{n-m}{2} \\ \frac{5}{2}y^2 &= (u^2-v^2)2uv \\ \therefore 5\left(\frac{y}{2}\right)^2 &= (u+v)(u-v)uv \end{aligned}$$

By construction, u, v are coprime to each other and $(u+v)(u-v)$ and uv are also coprime. Let $u = 41^2$ and $v = 5 \cdot 12^2$. Notice that the conditions that we have set while defining the variables are met. Moreover as given by the problem, it is evident that $(41^2 + 5 \cdot 12^2)(41^2 - 5 \cdot 12^2)$ are perfect squares, returning an integer y. Hence, the following equations are obtained.

$$\begin{aligned} 5\left(\frac{y}{2}\right)^2 &= (41^2+5\cdot 12^2)(41^2-5\cdot 12^2)\cdot 41^2 \cdot 5 \cdot 12^2 \\ \left(\frac{y}{2}\right)^2 &= (41^2+5\cdot 12^2)(41^2-5\cdot 12^2)\cdot 41^2 \cdot 12^2 \\ \therefore y &= 2\cdot 12 \cdot 41 \cdot \sqrt{(41^2+5\cdot 12^2)(41^2-5\cdot 12^2)} = 1494696 \end{aligned}$$

Moreover, $x = u^2 + v^2 = \left(41^2\right)^2 + \left(5 \cdot 12^2\right)^2 = 3344161$. Therefore, $(3344161, 1494696)$ is another pair.

With this in mind, let's discuss Pell's equation, another fundamental

topic from Diophantine equations.

§8.2 Infinite Descent

Infinite descent is a powerful proving tool in number theory that is used very frequently. Many, if not most, problems on Diophantine equations are proven with infinite descent and it has wide range of applications from showing that there are no solutions under a certain condition and finding solutions to advance techniques like Vieta jumping.

Before we get started with how this technique works, we need to discuss the foundational principle that infinite descent builds up on. Consider the following principle.

Theorem 8.2.1

The **well-ordering principle** states that there always exists a least element in an non-empty set of positive integers.

In olympiad, we take the principle above as an axiom and does not require a proof. Meaning, we can just use "by well-ordering principle..." when we prove by infinite descent.

The well-ordering principle is foundational for infinite descent because it allows us to have contradiction. Here is what I mean. When we prove with infinite descent, we assume that there exists a minimum value in a set. The minimum value could be a number, sum, and much more depending on the problem. Also, note that we can assume this because our interests only lies on positive integers and there cannot be a "least element" for rational and reals. Then, we will infinitely descend, eventually getting contradiction with our assumption. This is quite hard to understand without an example, so let's take a look at very famous example before formally stating the method described.

Exercise 8.2.2: Classic Problem

Show that the equation $x^3 + 2y^3 = 4z^3$ does not have any solutions in positive integers.

(Video Solution)

Solution.

For the sake of contradiction, assume that there exists positive integer solutions to the equation and let (a, b, c) be the solution with the least a by well-ordering principle. Consider the following congruence.

$$x^3 + 2y^3 \equiv 4z^3 \equiv 0 \pmod 2$$

Therefore, $2 \mid x$. Let x' be an integer such that $x = 2x'$. Substituting, $8x'^3 + 2y^3 = 4z^3$ and $4x'^3 + y^3 = 2z^3$ are obtained. By the similar argument, it is evident that $2 \mid y$ and $y = 2y'$ for some integer y'. Substituting again, $4x'^3 + 8y'^3 = 2z^3$ and $2x'^3 + 4y'^3 = z^3$. Therefore, $2 \mid z$ and there exists some positive integer z' such that $z = 2z'$. Continuing with the final substitution, $2x'^3 + 4y'^3 = 8z'^3$ and $x'^3 + 2y'^3 = 4z'^3$ are obtained. In other words, if (x, y, z) is a solution, (x', y', z') is also a solution to the equation.

Note that (a, b, c) is assumed to be a solution with the least a. Therefore, $(\frac{a}{2}, \frac{b}{2}, \frac{c}{2})$ is also a solution to the equation. Because $\frac{a}{2} < a$, there is no positive integer solutions to the equation.

As you can see from the example above, we assume the minimum and show that the infinite descent contradicts with our assumption.

Definition 8.2.3

The **proof by infinite descent** is a technique that shows contradiction through infinitely descending sequence. Assume that there exists a solution with minimum condition by well-ordering principle. Then, construct a strictly and infinitely descending sequence. The proof completes by contradiction.

We can expand our idea to see how we can use Pythagorean triples discussed earlier and infinite descent together to solve problems.

Exercise 8.2.4

Show that there are no integer triples (x, y, z) where $x \neq 0$ that satisfies $2x^4 + 2x^2y^2 + y^4 = z^2$.

~Mathematics Stack Exchange

Solution.

First, let $(x, y) = d$. Notice that dividing both sides by d^4 gives $\frac{z}{d^2} \in \mathbb{Z}$ and $d^2 \mid z$. Therefore, let $a = \frac{x}{d}$, $b = \frac{y}{d}$, and $c = \frac{z}{d^2}$ with minimum $a + b$ by well-ordering principle. Substituting to the original equation divided by d^2, $2a^4 + 2a^2b^2 + b^4 = c^2$ and $(a^2 + b^2)^2 + (a^2)^2 = c^2$. Continuing, m and n can be used to satisfy the Pythagorean triple.

Notice that if $a^2 = m^2 - n^2$, then $a^2 + b^2 = 2mn$. However, $a^2 + b^2 \equiv 2 \pmod 4$ or $a^2 + b^2 \equiv 1 \pmod 4$ since $(a, b) = 1$. In other words, $a^2 = 2mn$, $a^2 + b^2 = m^2 - n^2$ and $c = m^2 + n^2$ where $(m, n) = 1$.

Continuing, notice that $m^2 = a^2 + b^2 + n^2 \equiv n^2 + 1 \pmod 4$. If $2 \mid m$, then $m^2 \equiv 2 \pmod 4$, which is not possible. Therefore, $2 \mid n$. Let $n = 2p^2$ and $m = q^2$.

Note that $b^2 = m^2 - n^2 - a^2 = (m - n)^2 - 2n^2$ and $2n^2 = (m - n)^2 - b^2 = 8p^4$. Continuing, let $p = rs$ where $s \leq r$. Rearranging, $8p^4 = (q^2 - 2p^2 + b^2)(q^2 - 2p^2 - b^2)$ and $2q^2 - 4p^2 = 2r^4 + 4s^2$ or $2q^2 - 4p^2 = 4r^4 + 2s^2$. Thus, $q^2 - 2r^2s^2 = r^4 + 2s^4$ and $q^2 = 2s^4 + 2r^2s^2 + r^4$ or $q^2 = 2r^4 + 2r^2s^2 + s^4$. Notice that this is the same as the original equation where $(x, y, z) = (s, r, q)$ or $(x, y, z) = (r, s, q)$. Moreover, because $s + r \leq 2p < 2pq = a < a + b$, there exists no integer triples (x, y, z) that satisfies the equation.

Infinite descent have wide range of applications, and we can extend its application with Vieta's formulas.

8.2.1 Vieta Jumping

Vieta jumping is a technique that has been formally introduced with the famous problem 1988 IMO Problem 6. Now it is a very standard and common technique, but back then, it was one of the revolutionary technique.

The basic idea is to write a given equation as a quadratic equation with respect to a specific variable, then use the Vieta's formula to apply infinite descent. To see what I mean, let's actually solve 1988 IMO Problem 6 together to see how it works. Also, don't worry! It is often framed as one of the hardest IMO problems ever, but it is not too difficult when you understand the solution. Of course the problem is very difficult because Vieta jumping wasn't really a thing back then, if you understood the contents in this section, you would be surprised to see how satisfying this problem is!

Exercise 8.2.5: 1988 IMO Problem 6

Show that for positive integers a and b, if $\frac{a^2+b^2}{ab+1}$ is an integer, then it is a perfect square.

(Video Solution)

Solution.

Let $\frac{a^2+b^2}{ab+1} = k$ for some integer k and let $a > b$ without loss of generality. Assume for the sake of contradiction that k is not the square of an integer. Rearranging the equation, $a^2 + b^2 = abk + k$ holds and the following quadratic equation can be written with respect to a.

$$a^2 - bka + b^2 - k = 0$$

Using Vieta's formulas, the other root a' of the equation satisfy the

following equation if a is a root of the equation.

$$a' = bk - a = \frac{b^2 - k}{a}$$

Assume by well-ordering principle that (a, b) is a root of the equation with least $a + b$. Therefore, it suffices to show that $a' + b < a + b$ and that a' is a positive integer.

First, consider the following inequality for showing that $a' + b < a + b$.

$$a' = \frac{b^2 - k}{a} < \frac{b^2}{a} < \frac{b^2}{a} = a$$

Therefore, $a' < a$ and $a' + b < a + b$. Continuing, we can show that a' is always a positive integer. By construction, the following equation holds.

$$a'^2 - bka' + b^2 - k = 0$$

Rearranging the equation, $a'^2 + b^2 = bka' + k$. Assume for the sake of contradiction that $a' < 0$. Notice that the left-hand side is positive while the right-hand side $bka' + k = k(ba' + 1) \leq k(1 - b) \leq 0$. The inequality contradicts with the condition $bka' + k > 0$ given by the left-hand side. Continuing, if $a' = 0$, then $b^2 = k$ and this contradicts with the fact that k is not a perfect square. Thus, $\frac{a^2+b^2}{ab+1}$ is always a perfect square if the value is an integer for positive integers a and b.

As you can see from the method above, this is pretty simple when we know what Vieta jumping is! Honestly I am not a huge fan of the term Vieta jumping because it is infinite descent plus Vieta's formulas, but I think the term Vieta jumping made it more intuitive to be used in other type of problems because other problems can also require jumping from one root to the another using Vieta's formulas.

This is it for this section! Infinite descent is vary popular specially in Diophantine equations, and whenever you see lots of squares in the given equation, it is worth considering Vieta jumping. In the next

section, we will be discussing another very popular topic in Diophantine equations Pell's equations.

§8.3 Pell's Equations

§8.4 Practice Problems

Below are some practice problems with a subjective rating on difficulties. The solutions are included at the end of Math Olympiad part of LibreNotebook.

8.4.1. **Classic (● ○ ○)**
Find all natural number solutions to $x^3 + 2y^3 + 4z^3 - 6xyz = 0$.

Note 9

[N] Modular Arithmetic

The two major topics from olympiad Number Theory should be Diophantine equations and modular arithmetic. Having two major parts, there are many contents and techniques to discuss in modular arithmetic just as we did with Diophantine equations. In this note, we will discuss more fundamental topics like primitive roots and quadratic residue and will further discuss different techniques used in problems.

§9.1 Primitive Root Modulo n

Note 10

[C] Combinatorics

§10.1 Foundations

Note 11

Miscellaneous Goodies

This note discusses some of the miscellaneous Number Theory topics that are not necessarily the major part in olympiad NT, but would be very useful when solving problems. At least for KMO, I am aware that it is generally required to prove many of the contents here during the write-ups, but some problems are virtually impossible to solve without previous knowledge on these topics!

§11.1 p-adic Valuation & LTE Lemma

LTE Lemma, which stands for Lifting the Exponent Lemma, concerns about p-adic valuation of two integers. Before we get into the lemma, let's discuss about p-adic valuation. If you prefer video over text, here is also a video explanation!

Definition 11.1.1

The **p-adic valuation** of an integer n and prime p, denoted as $\nu_p(n)$, is the largest possible integer a such that $p^a \mid n$.

Some examples could be $\nu_2(48) = 3$ and $\nu_3(54) = 3$. It is worth noting

that some sources write this as $V_p(n)$, so they most likely refer to the same thing.

Considering this notation as a function, we can notice that the following equation holds for all positive integers m, n and some prime p.

$$v_p(mn) = v_p(m) + v_p(n)$$

This is pretty self evident because it simply counts the number of p in m and n. Another very interesting property is obtained when we add m and n instead of multiplying them.

$$v_p(m+n) \geq \min(v_p(m), v_p(n))$$

This could be pretty hard to see at first, but we can see it instantly if we factorize m and n. Say $m = p^x a$ and $n = p^y b$ where $(p, a) = 1$ and $(p, b) = 1$. Without loss of generality, assume that $x \geq y$. Then, $m + n = p^y(p^{x-y}a + b)$ and the inequality holds. To see when the equality holds, we simply just have to make $p^{x-y}a + b$ not divisible by p. In other words, as long as $x \neq y$, the equality holds.

This conclusion can lead to a more clean form. Given m and n such that $v_p(m) < v_p(n)$, the following equation holds.

$$v_p(m+n) = v_p(m)$$

I think this is pretty self-evident continuing from our conclusion.

> Another very famous example would be Legendre's formula, which states the following equation for positive n and prime p.
>
> $$v_p(n!) = \sum_{k=1}^{\infty} \left\lfloor \frac{n}{p^k} \right\rfloor$$

This is actually the same as the classic competition style problems that asks you the number of p in $n!$. I will skip the proof because I think it is pretty self-evident.

Another very famous application is understanding why $\binom{n}{r}$ is an integer. Obviously combinatoric-wise, its definition makes it an integer, but algebraic-wise, it is pretty interesting considering the fact that $\binom{n}{r} = \frac{n!}{r!(n-r)!}$. To show this, it suffices to prove the following inequality for any prime p.

$$\nu_p(n!) \geq \nu_p(r!) + \nu_p((n-r)!)$$

By Legendre's formula, the inequality converts as

$$\sum_{k=1}^{\infty} \left\lfloor \frac{n}{p^k} \right\rfloor \geq \sum_{k=1}^{\infty} \left(\left\lfloor \frac{r}{p^k} \right\rfloor + \left\lfloor \frac{n-r}{p^k} \right\rfloor \right)$$

and this is true because $\lfloor a+b \rfloor \geq \lfloor a \rfloor + \lfloor b \rfloor$. With this notation and concept in mind, lets get into LTE Lemma!

Theorem 11.1.2: Lifting the Exponent Lemma

Let p be a prime number and x, y be integers such that $p \mid x-y$, $p \nmid x$, and $p \nmid y$. Then, the respective equations hold for the three cases.

I. $\mathbf{p > 2}$

$$\nu_p(x^n - y^n) = \nu_p(x-y) + \nu_p(n)$$

II. $\mathbf{p = 2}$ **and** $\mathbf{2 \mid n}$

$$\nu_2(x^n - y^n) = \nu_2(x^2 - y^2) + \nu_2(n) - 1$$

III. $\mathbf{p = 2}$ **and** $\mathbf{2 \nmid n}$

$$\nu_2(x^n - y^n) = \nu_2(x-y)$$

Before we prove this, an interesting extension that we can notice for odd n and $p > 2$ is that $\nu_p(x^n + y^n) = \nu_p(x+y) + \nu_p(n)$. This can be obtained by substituting $-y$ for y.

Proof.

To prove the lemma, first separate p into two cases where $p > 2$ and $p = 2$. The first case $p > 2$ can be proven with induction.

First, consider the case when $\nu_p(n) = 0$. Notice that because $x - y \equiv 0 \pmod{p}$, $x \equiv y \pmod{p}$. Moreover,

$$x^n - y^n = (x - y)\left(\sum_{k=1}^{n} x^{n-k}y^{k-1}\right)$$

holds and $\sum_{k=1}^{n} x^{n-k}y^{k-1} \equiv \sum_{k=1}^{n} x^{n-1} \not\equiv 0 \pmod{p}$. In other words, $\nu_p(x^n - y^n) = \nu_p(x - y) + \nu_p(n)$.

Similarly, consider the base case when $\nu_p(n) = 1$. Let $n = kp$ where $p \nmid k$ and consider the following equations.

$$x^{kp} - y^{kp} = (x^p - y^p)\left(\sum_{i=1}^{k} x^{pk-ip}y^{ip-p}\right)$$

Notice that because

$$\sum_{i=1}^{k} x^{pk-ip}y^{ip-p} \equiv \sum_{i=1}^{k} x^{pk-p} \equiv kx^{pk-p} \not\equiv 0 \pmod{p}$$

holds, $\nu_p(x^n - y^n) = \nu_p(x^p - y^p)$. Continuing,

$$x^p - y^p = (x - y)\left(\sum_{k=1}^{p} x^{p-k}y^{k-1}\right)$$

and $\sum_{k=1}^{p} x^{p-k}y^{k-1} \equiv px^{p-1} \equiv 0 \pmod{p}$. Substituting $y = x + pt$ to the expression constructed by definition,

$$x^{p-k}y^{k-1} = x^{p-k}(x + pt)^{k-1} + x^{p-k}\left(x^{k-1} + (k-1)x^{k-2}pt + \alpha\right)$$

where $p^2 \mid \alpha$. Therefore, the following congruences hold.

$$\begin{aligned}\sum_{k=1}^{p} x^{p-k}y^{k-1} &\equiv \sum_{k=1}^{p} x^{p-k}x^{k-1} + x^{p-k}(k-1)x^{k-2}pt \\ &\equiv \sum_{k=1}^{p} x^{p-1} + x^{p-2}(k-1)pt\end{aligned}$$

$$\begin{aligned}
&\equiv px^{p-1} + \frac{p(p-1)}{2}x^{p-2}pt \equiv px^{p-1} \\
&\not\equiv 0 \pmod{p^2}
\end{aligned}$$

In other words, $\nu_p(x^p - y^p) = \nu_p(x-y) + 1 = \nu_p(x-y) + \nu_p(n)$ and the base case holds.

Continuing, it suffices to show that if the equation holds for $\nu_p(n) = a$, then it also holds for $\nu_p(n) = a+1$. Let $n = p^a b$ where $p \nmid b$, and consider the following equations.

$$\begin{aligned}
\nu_p\left(x^{p^{a+1}b} + y^{p^{a+1}b}\right) &= \nu_p\left(\left(x^{pb}\right)^{p^a} + \left(y^{pb}\right)^{p^a}\right) \\
&= \nu_p\left(x^{pb} - y^{pb}\right) + a \\
&= \nu_p(x-y) + \nu_p(pb) + a = \nu_p(x-y) + a + 1 \\
&= \nu_p(x-y) + \nu_p(n)
\end{aligned}$$

Therefore by induction, the first case holds.

To prove the second case, consider the following equations with $n = 2^a b$ where $2 \nmid b$.

$$\begin{aligned}
\nu_2\left(x^n - y^n\right) &= \nu_2\left(\left(x^{2^a}\right)^b - \left(y^{2^a}\right)^b\right) \\
&= \nu_2\left(\left(x^{2^a} - y^{2^a}\right)\left(\sum_{k=1}^{b}\left(x^{2^a}\right)^{b-k}\left(y^{2^a}\right)^{k-1}\right)\right) \\
&= \nu_2\left(x^{2^a} - y^{2^a}\right)
\end{aligned}$$

Moreover, because

$$\begin{aligned}
x^{2^a} - y^{2^a} &= \prod_{k=1}^{a}\left(x^{2^{a-k}} + y^{2^{a-k}}\right)\cdot(x-y) \\
&= \prod_{k=1}^{a-1}\left(x^{2^{a-k}} + y^{2^{a-k}}\right)\cdot(x+y)(x-y)
\end{aligned}$$

holds and $x^{2^k} + y^{2^k} \equiv 2 \pmod 4$ for $k > 1$, it is evident that

$$v_p\left(x^{2^a} - y^{2^a}\right) = v_p(x-y) + v_p(x+y) + a - 1$$

holds and the second case holds.

Finally, the last case can be simply proven with the following equation.

$$x^n - y^n = (x-y)\left(\sum_{k=1}^{n} x^{n-k}y^{k-1}\right)$$

Notice that because $\sum_{k=1}^{n} x^{n-k}y^{k-1} \equiv \sum_{k=1}^{n} x^{n-1} \equiv nx^{n-1} \not\equiv 0 \pmod p$, $v_2(x^n - y^n) = v_2(x-y)$.

This proof was quite long, but absolutely worth it! Before we move on to a more advanced usage, here is a simple example.

$$v_3\left(4^3 + 5^3\right) = v_3\left(4^3 - (-5)^3\right) = v_3(4+5) + v_3(3) = 2 + 1 = 3$$

To check this, notice that $v_3(4^3 + 5^3) = v_3(64 + 125) = v_3(189) = 3$. Continuing, here is another problem.

Exercise 11.1.3: Enoch's Problem

Find the greatest integer a such that $2^a \mid 7^{2026} - 1$.

Solution.

Note that it suffices to find $v_2(7^{2026} - 1)$. Using LTE, we can see that $v_2(7^2 - 1) + v_2(2026) - 1 = 4 + 1 - 1 = 4$. Therefore, the greatest integer a is 4.

With this idea in mind, let's take a look at how LTE can be used to solve a more complicated problem.

Exercise 11.1.4

For all positive integers k, show that 2 is a primitive root modulo 5^k.

Solution.

First, recall that 2 is a primitive root mod 5^k if $\min(2) = \phi(5^k)$ for $2^n \equiv 1 \pmod{5^k}$. Notice that for 2^n to be congruent to 1 mod 5, n must be a multiple of 4 since cycles $2, -1, -2, 1$. Therefore, let $n = 4N$.

Rewriting the congruence, $16^N - 1 \equiv 0 \pmod{5^k}$ and the following inequality holds.

$$\nu_5\left(16^N - 1\right) = \nu_5\left(a \cdot 5^k\right) \geq k$$

Using LTE, it is evident that $\nu_5\left(16^N - 1\right) = \nu_5(16 - 1) + \nu_5(N) \geq k$ and $\nu_5(N) \geq k - 1$. Therefore, $N \geq 5^{k-1}$ and $n \geq 4 \cdot 5^{k-1} = \phi(5^k)$ holds and 2 is a primitive root mod 5^k.

This is much simpler than induction, and is it for this section! I think LTE is one of those very powerful technique that often breaks the problem just by applying it. In the next section, we will discuss about combinatoric approach to NT problems.

§11.2 Proving and Problem Solving Tips

Throughout studying and solving olympiad style problems, I noticed and came up with some patterns and tips that I think could be helpful. I am not a olympiad coach or anything, but maybe I could share some of my thoughts and topics that I heard that quite surprised me.

11.2.1 Proving Inequalities

Inequality is definitely one of my favorite section, and I think there are many components that makes a solution elegant. Particularly, I think solutions that makes inequality feel natural are very elegant. Here is what I mean. Instead of trying to "solve the inequality" by making the best out of the conditions, I think "proving the inequality" in a natural

way is more elegant. Say we have a constant on the right-hand side and we want to prove it. Some "bad proof" could include brute forcing and trying to figure out the meaning of the inequality just by looking at the constant.

It is, however, very important to deduce hints from the constants because they could reveal something very important like 2025 KMO Problem 6, which is included in Section 2.4. However that problem is also elegant in a way that it naturally uses the condition to prove the inequality without brute forcing. In short, I think solutions are elegant when we could deduce like this is true and that is true, then this gives the constant. Here is an example to manifest my point.

Exercise 11.2.1: 1997 KMO Problem 4 (● ○ ○)

Given positive a, b, c such that $a + b + c = abc$, show that $\sum_{\text{cyc}} \frac{1}{\sqrt{1+a^2}} \leq \frac{3}{2}$.

Solution.

I particularly like this problem because there are few ways of solving this, but the standard and famous solution deduces the constant from the left-hand side.

Consider the following equation for $\triangle ABC$.

$$\tan C = \tan 180^\circ - (A + B) = -\tan(A + B) = -\frac{\tan A + \tan B}{1 - \tan A \tan B}$$

In other words, $\tan A + \tan B + \tan C = \tan A \tan B \tan C$. Without loss of generality, let $0^\circ < A, B < 90^\circ$. Notice that because c depends on a and b and $\tan A$ and $\tan B$ spans positive reals, we can let $a = \tan A$, $b = \tan B$, and $c = \tan C$. Moreover, we could notice that $C \in (0^\circ, 90^\circ)$ since $c > 0$.

Continuing, consider the following inequality by Jensen.

$$\sum_{\text{cyc}} \frac{1}{\sqrt{1+\tan^2 A}} = 3\sum_{\text{cyc}} \frac{\cos A}{3} \leq 3\cos\left(\frac{A+B+C}{3}\right) = \frac{3}{2}$$

Therefore, the inequality holds.

I think what makes this problem particularly elegant is the fact that we could transform the given conditions to different forms and we could use that to find the constant. Meaning, we could have derived this problem from scratch without even looking at the right-hand side.

11.2.2 Solving FE and Geometry Problems

Guess, draw well, then prove

11.2.3 Proving Techniques

Induction Base case, use base to go from a to a+1

§11.3 Practice Problems

Below are some practice problems with a subjective rating on difficulties. The solutions are included at the end of Math Olympiad part of LibreNotebook.

11.3.1. **1996 Ireland Problem 8 (● ● ○)**
Given a prime p and $a, n \in \mathbb{N}$, show that if $2^p + 3^p = a^n$, then $n = 1$.

Note 12

Solutions

Below are the solutions to the practice problems at the end of each note.

§12.1 Solutions for Note 1

Part II

Linear Algebra

Contents

Introducing the Notes **167**

13 Vectors and Vector Spaces **169**
13.1 Basic Terms and Notations 169
13.2 Fundamental Properties 173
13.3 Subspaces 180
13.3.1 Linear Combinations and Span 182
13.3.2 Linear Independence 185
13.3.3 Basis and Dimension 190
13.4 Practice Problems 197

14 Matrices **199**
14.1 Basic Terms and Notations 199
14.2 Fundamental Properties 202
14.3 Matrix Multiplication 206
14.3.1 The Method and Properties 208
14.4 Additional Fundamental Topics 216
14.4.1 Transpose 216
14.4.2 Partitions 221
14.4.3 Special Forms of Matrix 223
14.5 Determinants 227
14.5.1 Determinant and Products 232
14.6 Practice Problems 238

15 System of Linear Equations **241**
15.1 Basics and Intuitions 241
15.2 Gaussian Elimination 246
15.3 The Multiplicative Inverse 251
15.4 LU Decomposition 262
15.4.1 Application in Linear Systems 265
15.5 Practice Problems 268

16 Matrices and Vector Spaces **271**
16.1 Row Space and Rank of a Matrix 271
16.2 Linear Transformations 282
16.2.1 Brief Review of Functions 282
16.2.2 Basic Terms and Notations 284
16.2.3 Matrices and Linear Transformation 289
16.2.4 Kernels and Images 294
16.3 Eigenvalues and Eigenvectors 304
16.3.1 Properties of Eigenvalues 310
16.3.2 Diagonalization 315
16.4 Practice Problems 319

17 Solutions **321**
17.1 Solutions for Note 1 321
17.2 Solutions for Note 2 323
17.3 Solutions for Note 3 326
17.4 Solutions for Note 4 329
17.5 Final Thoughts 332

Introducing the Notes

Linear algebra is a foundational branch of mathematics that primarily investigates in linear equations, vectors, and their relations. The application of linear algebra is so immense that numerous fields require foundational understanding of linear algebra. For instance, it would not be possible to gain profound understandings in machine learning and data science without the basics of linear algebra. This collection of notes discuss the essence of linear algebra in perspective of teaching.

One of the most effective way to learn, at least for me, is to teach. During the summer of 2026, I was enrolled in the course XM511 from Stanford Pre-Collegiate University-Level Online Math, instructed by Dr. Margarita Kanarsky [Kan26]. During the lectures, I thought it would be a great idea to write notes on linear algebra just as I wrote the notes for machine learning. Writing the teaching-style notes for machine learning allowed me to differentiate what I truly know.

That being said, my notes do not represent the course. I have included my understandings the supporting textbook [BC07] from numerous external sources and restructured the notes itself to adhere to the Permission to Use Materials. Therefore, any errors in these notes are my responsibility.

Note 13

Vectors and Vector Spaces

If you have backgrounds on physics or computer science, you are probably familiar with what vectors are. In physics classes, we learn that vectors are quantities with magnitude and direction. In computer science, vectors are introduced as lists or arrays of data. Both definitions capture different aspects of what vectors are. Throughout the notes, we will implement both understandings. In this note, we will discuss fundamental terms for vectors and vector spaces before implementing matrices in vector spaces. Also, most of the heavy abstractions are done in modern algebra, so it will be an introduction.

§13.1 Basic Terms and Notations

First, let's get started by building from common notations and definitions.

Definition 13.1.1

$\mathbb{R}$ and $\mathbb{C}$ denote the set of real and complex numbers respectively. Epsilon symbols $\in$ and $\notin$ are used to denote whether an element is in the set or not.

Examples of using such notations would be $\pi \in \mathbb{C}$ and $i \notin \mathbb{R}$.

Definition 13.1.2

A **vector space** $\mathbb{V}$ is a set of elements and scalars with addition properties and scalar multiplication that adheres to the following ten properties, or **axioms for a vector space**. Moreover, the elements of $\mathbb{V}$ are known as **vectors**.

Vector Addition

A1. **Closure:** $\forall\, u, v \in \mathbb{V}$, $u \oplus v \in \mathbb{V}$.

A2. **Commutativity:** $\forall\, u, v \in \mathbb{V}$, $u \oplus v = v \oplus u$.

A3. **Associativity:** $\forall\, u, v, w \in \mathbb{V}$, $u \oplus (v \oplus w) = (u \oplus v) \oplus w$.

A4. **Additive Identity Element:** $\exists\, 0 \in \mathbb{V}$ such that $\forall\, u \in \mathbb{V}, u \oplus 0 = u$, also known as the **zero vector**.

A5. **Additive Inverses:** $\forall\, u \in \mathbb{V}, \exists\, -u \in \mathbb{V}$ such that $u \oplus (-u) = 0$.

Scalar Multiplication

S1. **Closure:** $\forall\, u \in \mathbb{V}$ and scalars α, $\alpha \odot u \in \mathbb{V}$.

S2. **Associativity:** $\forall\, u \in \mathbb{V}$ and scalars α, β, $\alpha \odot (\beta \odot u) = (\alpha \cdot \beta) \odot u$.

S3. **Identity Element:** $\forall\, u \in \mathbb{V}$, $1 \odot u = u$.

S4. **Distributivity I:** $\forall\, u \in \mathbb{V}$ and scalars α, β, $(\alpha + \beta) \odot u = \alpha \odot u \oplus \beta \odot u$.

S5. **Distributivity II:** $\forall\, u, v \in \mathbb{V}$ and scalars α, $\alpha \odot (u \oplus v) = \alpha \odot u \oplus \alpha \odot v$.

Side note here is that for this section of the note, we will be using the binary operation symbols $\oplus$ and $\odot$ to denote vector addition and scalar multiplication. However, from the next section, we will be using more standard notations like $v + u$ and λv for vector addition and scalar multiplication. The reason I decided to use this is that this is the way I first learned and it helped me to emphasize axioms for vector spaces.

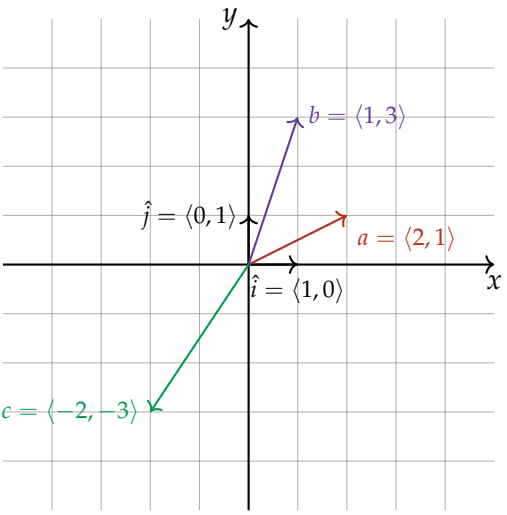
Definition 13.1.3

An n-**tuple** is an ordered array, or list, of n real numbers. For an n-tuple, its dimension is said to be n-dimensional and the set of such real tuples is n-**space**, denoted as $\mathbb{R}^n$.

For instance, the following 4-tuple is in 4-space.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \in \mathbb{R}^4$$

With tuples in mind, let's take a look at how we can represent 2-tuples, 3-tuples, and vectors in $\mathbb{R}^2$ and $\mathbb{R}^3$. The vector space $\mathbb{R}^2$ can simply be conceived as 2-dimensional space, or plane. Also, in linear algebra, we generally don't move the tail of a vector from the origin as we do in introductory physics classes.



From the graph above, there are two vectors $\hat{\imath}$ and $\hat{\jmath}$. They are called "i hat" and "j hat" respectively.

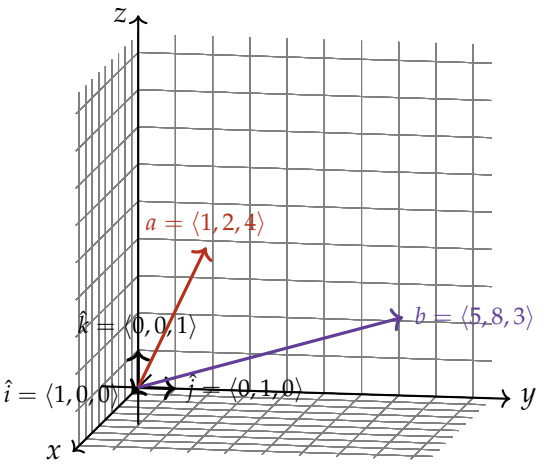
Definition 13.1.4

Unit Vectors are vectors with magnitude 1 that are used to specify direction.

Here, $\hat{i}$ and $\hat{j}$ denote the unit vectors, and they are the standard symbol. If you wondered if unit vectors are used to construct vectors, you are right! In our examples above, we can write each vector as the following.

$$\begin{aligned} a &= \langle 2, 1 \rangle = 2\hat{i} + \hat{j} \\ b &= \langle 1, 3 \rangle = 1\hat{i} + 3\hat{j} \\ c &= \langle -2, -3 \rangle = -2\hat{i} - 3\hat{j} \end{aligned}$$

Now, let's take a look at vector space $\mathbb{R}^3$. There are two ways of graphing 3-dimensional space: right-handed system and left-handed system. However, we will be using right-hand system throughout the notes.



Here, $\hat{i}$, $\hat{j}$, and $\hat{k}$ are the unit vectors. Going back to the definition of vector spaces, we can write the following equations for $u = \langle x_1, y_1, z_1 \rangle, v = \langle x_2, y_2, z_2 \rangle \in \mathbb{R}^3$ and $\alpha \in \mathbb{R}$.

$$\begin{aligned} u \oplus v &= \langle x_1, y_1, z_1 \rangle + \langle x_2, y_2, z_2 \rangle = \langle x_1 + x_2, y_1 + y_2, z_1 + z_2 \rangle \\ \alpha \odot u &= \alpha \odot \langle x_1, y_1, z_1 \rangle = \langle \alpha x_1, \alpha y_1, \alpha z_1 \rangle \end{aligned}$$

Continuing, we can also distinguish vector spaces with different scalars with the following terms.

Definition 13.1.5

If the set of scalars is in $\mathbb{R}$, then the vector space is known as a **vector space over** $\mathbb{R}$, or a **real vector space**. Similarly, if the scalars are in $\mathbb{C}$, then the vector space is denoted as a **vector space over** $\mathbb{C}$, or a **complex vector space**.

One thing to note is that there is a concept called **scalar fields**; however, we will not go too deep into such fields. Before we move on to different concepts in vector spaces, I wanted to discuss about tuples and dimensions.

Now that we saw some geometrical interpretation of vector spaces, let's discuss more information with some problems.

§13.2 Fundamental Properties

Let's first start by solving some problems.

Exercise 13.2.1

Identify if the set $\mathbb{S} = \{(x, y) \in \mathbb{R}^2 \mid x + y = 5\}$ is a vector space.

Solution.

We could see if the set satisfies all axioms for a vector space. However, just by looking at the condition $x + y = 5$, we know that there cannot exist a zero vector since $0 + 0 \neq 5$. Because the set violates A4, the set is not a vector space.

Exercise 13.2.2

Identify if the set $\mathbb{S} = \{\langle x, y\rangle \in \mathbb{R}^2 \mid x + y = 0\}$ is a vector space.

Solution.

Unlike the previous problem, we know that there exists a zero vector. For this one, let's go through each axiom. Let $u = \langle x_1, -x_1\rangle$ and

$v = \langle x_2, -x_2 \rangle$. Notice that $u \oplus v = \langle x_1 + x_2, -x_1 - x_2 \rangle \in \mathbb{R}^2$ and satisfies the condition $x_1 + x_2 + (-x_1 - x_2) = 0$. Therefore, A1 is satisfied. Moreover, consider the following equations for A2 and A3 where $w = \langle x_3, -x_3 \rangle$.

$$\begin{aligned}
u \oplus v &= \langle x_1, -x_1 \rangle + \langle x_2, -x_2 \rangle = \langle x_1 + x_2, -x_1 - x_2 \rangle \\
&= \langle x_2 + x_1, -x_2 - x_1 \rangle = \langle x_2, -x_2 \rangle + \langle x_1, -x_1 \rangle \\
&= v \oplus u \\
u \oplus (v \oplus w) &= \langle x_1, -x_1 \rangle + (\langle x_2, -x_2 \rangle + \langle x_3, -x_3 \rangle) \\
&= \langle x_1, -x_1 \rangle + \langle x_2 + x_3, -x_2 - x_3 \rangle \\
&= \langle x_1 + x_2 + x_3, -x_1 - x_2 - x_3 \rangle \\
&= \langle x_1 + x_2, -x_1 - x_2 \rangle + \langle x_3, -x_3 \rangle \\
&= (u \oplus v) \oplus w
\end{aligned}$$

Therefore, A2 and A3 are satisfied. Furthermore, notice that $\langle 0, 0 \rangle \in \mathbb{R}^2$ and $0 + 0 = 0$, satisfying A4. Lastly for vector addition A5 holds as $\langle -x, x \rangle$ is the additive inverse of $\langle x, -x \rangle$ that satisfies the provided conditions. Thus, all axioms for vector addition are met.

Continuing with scalar multiplication, notice that S1 and S3 satisfy as $\langle x, y \rangle \in \mathbb{R}^2$ and $1 \odot \langle x, y \rangle = \langle x, y \rangle$. Moreover, consider the following equations for $\alpha, \beta \in \mathbb{R}$.

$$\begin{aligned}
\alpha \odot (\beta \odot u) &= \alpha \odot \langle \beta x_1, -\beta x_1 \rangle = \langle \alpha\beta x_1, -\alpha\beta x_1 \rangle = \alpha\beta \odot \langle x_1, -x_1 \rangle \\
&= (\alpha\beta) \odot u \\
(\alpha + \beta) \odot u &= \langle (\alpha + \beta)x_1, -(\alpha + \beta)x_1 \rangle = \langle \alpha x_1 + \beta x_1, -\alpha x_1 - \beta x_1 \rangle \\
&= \langle \alpha x_1, -\alpha x_1 \rangle + \langle \beta x_1, -\beta x_1 \rangle \\
&= \alpha \langle x_1, -x_1 \rangle + \beta \langle x_1, -x_1 \rangle \\
&= \alpha \odot u \oplus \beta \odot u \\
\alpha \odot (u \oplus v) &= \alpha \langle x_1 + x_2, -x_1 - x_2 \rangle = \langle \alpha x_1 + \alpha x_2, -\alpha x_1 - \alpha x_2 \rangle \\
&= \langle \alpha x_1, -\alpha x_1 \rangle + \langle \alpha x_2, -\alpha x_2 \rangle
\end{aligned}$$

$$= \alpha\langle x_1, -x_1\rangle + \alpha\langle x_2, -x_2\rangle$$
$$= \alpha \odot u \oplus \alpha \odot v$$

Thus, S2, S4, and S5 are all satisfied. Because all ten axioms for a vector space are met, the set $\mathbb{S} = \{\langle x, y\rangle \in \mathbb{R}^2 \mid x + y = 0\}$ is a vector space.

Exercise 13.2.3

Consider vectors in the form $\langle x, y, z\rangle$ where the vector addition is defined as $\langle x, y, z\rangle \oplus \langle x', y', z'\rangle = \langle x + x', y + y', z + z'\rangle$ and scalar multiplication as $\alpha\langle x, y, z\rangle = \langle x^\alpha, y^\alpha, z^\alpha\rangle$. Identify if the vectors form a vector space.

Solution.

First, notice that the vector addition adheres to the standard vector addition. Therefore, we only have to look at scalar multiplication if any axioms are violated. From a glance, S1, S2, and S3 appears to be satisfied. Therefore, let's take a look at S4.

$$(\alpha + \beta) \odot u = \left(x^{\alpha+\beta}, y^{\alpha+\beta}, z^{\alpha+\beta}\right)$$

Notice that in general, $x^{\alpha+\beta} \neq x^\alpha + x^\beta$. Therefore, the set of vectors does not form a vector space.

Just as we have vector spaces $\mathbb{R}^n$, we could also define spaces depending on the objects.

Definition 13.2.4

The symbol $\mathbb{P}^n$ denotes the vector space composed of polynomials of degree at most n and adheres to the ten properties. $\mathbb{M}_{m\times n}$ represents the vector space with matrices of order $m \times n$ that adheres to the ten properties.

Therefore, we can say that $\mathbb{R}^n$ is more or less the same as $\mathbb{M}_{1\times n}$. With

the understandings above, we can derive some important theorems. Although most of them are intuitive, it's always nice to formally prove them so let's do that now.

Theorem 13.2.5

For all vector spaces, zero vector, or the additive identity element, is unique.

Proof.

For the sake of contradiction, let $0_1, 0_2 \in V$ be distinct additive identity element for a vector space V. By definition, the following equations hold.

$$0_1 \oplus 0_2 = 0_1$$
$$0_1 \oplus 0_2 = 0_2$$
$$\therefore 0_1 \oplus 0_2 = 0_1 = 0_2$$

Because $0_1 = 0_2$ is contradictory from the fact that 0_1 and 0_2 are distinct, the initial assumption that there exists distinct additive identity element in a vector space is false.

Continuing from the uniqueness of the zero vector in a vector space, we can assert the following theorem.

Theorem 13.2.6

For all $u \in V$ for any vector space V, $v \in V$ such that $u \oplus v = 0$ is unique.

Proof.

For the sake of contradiction, assume $v, w \in V$ are distinct additive inverses of u. By definition and properties of vector spaces, the following equations hold.

$$v = v \oplus 0 = v \oplus (u \oplus w) = (v \oplus u) \oplus w = 0 \oplus w = w$$

Because $v = w$ contradicts the definition that v and w are unique, the initial assumption that there exists more than one additive inverse of a vector in vector spaces is false.

Just as $x = 0$ if $x + x = x$, we could do similar things in vector spaces.

Theorem 13.2.7

For all $v \in V$ and any vector space V, $v \oplus v = v$ if and only if $v = 0$.

Proof.

First, the first half of the statement can be proven. Consider the following equations.

$$\begin{aligned}(v \oplus v) \oplus (-v) &= v \oplus (-v) = 0 \\ &= v \oplus (v \oplus (-v)) = v \oplus 0 = v\end{aligned}$$

Therefore, if $v \oplus v = v$, then $v = 0$. Moreover, if $v = 0$, then $v \oplus v = 0 \oplus 0 = 0$. Therefore, the premise holds.

Now let's take a look at theorems for scalar multiplications.

Theorem 13.2.8

The following properties hold for all vector space V over $\mathbb{F}$, $u, \mathbf{0} \in V$, and any scalars α.

1. $0 \odot u = \mathbf{0}$
2. $\alpha \odot \mathbf{0} = \mathbf{0}$.
3. $(-1) \odot u = -u$
4. $u = -(-u)$
5. $\alpha \odot u = \mathbf{0}$ if and only if $\alpha = 0$ or $u = \mathbf{0}$.

Let's start with the first proof.

Proof.

Consider the following equation.

$$0 \odot u = (0 + 0) \odot u = 0 \odot u \oplus 0 \odot u$$

By Theorem 13.2.7, $0 \odot u = \mathbf{0}$.

Below is the proof for the second property.

Proof.

Consider the following equation.

$$\alpha \odot \mathbf{0} = \alpha \odot (\mathbf{0} \oplus \mathbf{0}) = \alpha \odot \mathbf{0} \oplus \alpha \odot \mathbf{0}$$

By Theorem 13.2.7, $\alpha \odot \mathbf{0} = \mathbf{0}$

Continuing, here's the proof for the third one.

Proof.

By the properties of vector spaces, $\mathbf{0} = 0 \odot u = (1 - 1) \odot u = 1 \odot u \oplus (-1) \odot u = u \oplus (-1) \odot u$. Because $(-1) \odot u$ is the additive inverse of u, $(-1) \odot u = -u$.

Below is the proof for the fourth one.

Proof.

Notice that $-(-u)$ is the additive inverse of $-u$. However, by definition, u is the additive inverse of $-u$. By Theorem 13.2.6, $u = -(-u)$.

Finally, let's prove the last property.

Proof.

It suffices to show the latter half of the statement as the first half is shown by the first two properties. Consider the following equations

for $\alpha \neq 0$.

$$\begin{aligned}\frac{1}{\alpha} \odot (\alpha \odot u) &= \frac{1}{\alpha} \odot \mathbf{0} = \mathbf{0} \\ &= \left(\frac{1}{\alpha} \cdot \alpha\right) \odot u = 1 \odot u = u\end{aligned}$$

Therefore, if $\alpha \neq 0$, then $u = \mathbf{0}$. Similarly, if $u \neq \mathbf{0}$, then $\alpha = 0$ as the product of nonzero numbers is a nonzero number.

We have covered the fundamental concepts and properties in vector spaces. Before we move on with the next topic in vector spaces, I wanted to discuss briefly on dot product as it will frequently appear when you deal with vectors.

Definition 13.2.9

Dot product is an algebraic operation defined as the following for vectors $a = [a_1, \ldots, a_n]$ and $b = [b_1, \ldots, b_n]$

$$a \cdot b = \sum_{k=1}^{n} a_k b_k$$

Here notice that the number of elements of the vectors in dot product must be equal for the product to be defined. Below are some examples of dot products.

$$\begin{bmatrix}1\\2\\3\end{bmatrix} \cdot \begin{bmatrix}-1\\-2\\-3\end{bmatrix} = 1(-1) + 2(-2) + 3(-3) = -14$$

$$\begin{bmatrix}0 & 1 & 2\end{bmatrix} \cdot \begin{bmatrix}0 & 1 & 0\end{bmatrix} = 0 \cdot 0 + 1 \cdot 1 + 2 \cdot 0 = 1$$

Note that this is very different from the matrix multiplication that we will discuss in the next note and we must write the notation $\cdot$ to ensure that we are not doing matrix multiplication. With that in mind, let's continue our discussion with subspaces.

§13.3 Subspaces

In the previous section, we discussed that all ten axioms must be satisfied for a set to be a vector space. However, testing and proving that a set satisfies all ten axioms, to be honest, is quite tedious. The good news is that we only need to prove three conditions to show that a set is a vector space thanks to **subspaces**!

Definition 13.3.1

A **subspace** of a vector space V is a subset of V that uses the same definition of vector addition and scalar multiplication from V.

Now for a set to be a subspace, it must satisfy the following conditions.

Theorem 13.3.2

A non-empty subset S of a vector space V is a subspace of V if and only if the following conditions are satisfied.

1. **Closure for Addition:** $\forall u, v \in S, u + v \in S$
2. **Closure for Scalar Multiplication:** $\forall u \in S$ and scalars α, $\alpha u \in S$

Proof.

Notice that it suffices to show that S is a vector space, or adhere to the then axioms for a vector space. Moreover, A1 and S1 are assumed to be true by the initial condition.

By the second condition, $\mathbf{0} = 0u \in S$ and $-u = (-1)u \in S$. Therefore, A4 and A5 are satisfied. For A2 and A3, $u, v, w \in V$ is true for $u, v, w \in S$ by definition. Moreover, because V is a vector space, A2 and A3 hold. Similarly, because V is a vector space that satisfies S1, S2, S3, S4, and S5, its subset S also satisfies the conditions. Therefore, if the two closure conditions are met and S is a subset of V, then S is a subspace of V.

Now the good thing is that subspaces are vector spaces, so it suffices

to show that a set is a subspace of a known vector space with the two conditions shown above to show that it is a vector space. Let's take a look at an example.

Exercise 13.3.3

Determine whether $S = \{(x, y, z) \in \mathbb{R}^3 \mid y = 0\}$ is a vector space.

Solution.

Notice that S is a non-empty subset of $\mathbb{R}^3$, which is a known vector space. Therefore, it suffices to show that $u = (x_1, 0, z_1), v = (x_2, 0, z_2) \in S$ and scalars α satisfy the closure conditions for addition and scalar multiplication.

Notice that $u + v = (x_1 + x_2, 0, z_1 + z_2) \in S$. Moreover, $\alpha u = (\alpha x_1, 0, \alpha z_1) \in S$. Therefore, S is a vector space, more specifically a subset of $\mathbb{R}^3$.

As you can see from the solution above, subspaces really made our lives easier. However, to make it even better, we can boil it down to a single condition.

Corollary 13.3.4

For scalars α and β and $u, v \in S$ where S is a non-empty subset of a vector space V that has the same operations for vector addition and scalar multiplication, S is a subspace if and only if the following condition is satisfied.

$$\alpha u + \beta v \in S$$

Proof.

First and foremost, the first half of the statement can be proven. If $\alpha u + \beta v \in S$, then $u + v \in S$ where $\alpha = \beta = 1$ since α and β are all scalars. Similarly, $\alpha u + \beta v \in S$ implies that $\alpha u \in S$ where $\beta = 0$. By Theorem 13.3.2, S is a subspace of V.

> Continuing with the second half of the statement, if S is a subset of V, then $\alpha u, \beta v \in S$. Therefore, $\alpha u + \beta v \in S$, proving the corollary.

More examples on this can be found in the practice problems below. Now let's get into the topics of **linear combination** and the **span** of a set.

13.3.1 Linear Combinations and Span

As always, let's start with definitions.

Definition 13.3.5

For scalars α_i and v_i in a vector space V, the vector u defined as the following is a **linear combination of the vectors** v_i.

$$u = \sum_{k=1}^{n} \alpha_k v_k$$

For instance, $[1, 2, 3]$ is a linear combination of $[2, -2, 1]$, $[1, 0, 2]$, and $[1, -1, 1]$ as it can be represented as the following.

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 \end{bmatrix} + 3 \begin{bmatrix} 1 & 0 & 2 \end{bmatrix} - 4 \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}$$

Now we have a special term for the collection of such vectors.

Definition 13.3.6

The **span of the set of vectors** $\{v_1, \dots v_n\}$ is the set of all possible linear combinations of the vectors.

$$\text{span}\{v_1, \dots, v_n\} = \left\{ \sum_{k=1}^{n} \alpha_k v_k \;\middle|\; \alpha_k \in \mathbb{R} \right\}$$

With the definition, we could notice an important theorem.

Theorem 13.3.7

For a vector space V and its subset $S = \{v_1, \ldots, v_n\}$, $\text{span}(S)$ is a subspace of V.

Proof.

First and foremost, notice that $\mathbf{0} \in \text{span}(S)$ since all the scalars in linear combination can equal zero. Moreover, let $u = \sum_{k=1}^{n} \alpha_k v_k$ and $v = \sum_{k=1}^{n} \alpha_k v_k$ for any scalars α_k and β_k. By definition, $\alpha u + \beta v \in \text{span}(S)$ is true. Therefore, by Corollary 13.3.4, $\text{span}(S)$ is a subspace of V.

Side note here is that there are spans of infinite sets and they adhere to the theorem above. Continuing with the definition of the span of a set and its relation with subspace, we can show the two following theorems.

Theorem 13.3.8

Consider the following statements for a vector space V and its subspace W and non-empty subset S such that $S \subset W$.

1. $\text{span}(S) \subset W$
2. The intersection of all subspaces containing S is $\text{span}(S)$.

Let's start by proving the first statement.

Proof.

Let $u = \sum_{k=1}^{n} \alpha_k v_k \in \text{span}(S)$. By definition, $v_k \in W$ for integer $k \in [1, n]$. Moreover, $u \in W$ by Corollary 13.3.4 and $\text{span}(S) \subset W$.

Below is the proof for the second one.

Proof.

First and foremost, notice that by the previous statement, it was shown that $\text{span}(S)$ is included in all subspaces that contains S. Let A be the intersection of all subspaces that contain S. Therefore, $\text{span}(S) \subset A$. Moreover, by Theorem 13.3.7 $\text{span}(S)$ is a subspace

of V that contains S. Thus, $A \subset \text{span}(S)$. Because $\text{span}(S) \subset A$ and $A \subset \text{span}(S)$, $\text{span}(S) = A$.

Next, let's conclude our discussion on linear combination and span of a set of vectors by proving the following statements.

Theorem 13.3.9

For a vector space V and its subsets $S, T \subset V$, consider the following statements.

1. If $T \subset S$, then $\text{span}(T) \subset \text{span}(S)$.
2. $\text{span}(\text{span}(S)) = \text{span}(S)$.
3. If $T \subset \text{span}(S)$, then $\text{span}(T) \subset \text{span}(S)$.
4. For nonzero vectors $v_1, \ldots, v_n \in V$, scalar $\lambda \in \mathbb{R}$, and $1 \leq j, k \leq n$ such that $\lambda \neq 0$ and $j \neq k$, the following equations hold.

$$\text{span}\{v_1, \ldots, v_k, \ldots, v_n\} = \text{span}\{v_1, \ldots, v_k + \lambda v_j, \ldots, v_n\}$$
$$\text{span}\{v_1, \ldots, v_k, \ldots, v_n\} = \text{span}\{v_1, \ldots, \lambda v_k, \ldots, v_n\}$$

There's a lot to prove, but let's do them one by one starting with the first one.

Proof.

Notice that since all $t_k \in T$ satisfy $t_k \in S$, all $\sum_{k=1}^{n} c_k t_k \in \text{span}(T)$ for some appropriate n and constant c_k also satisfy $\sum_{k=1}^{n} c_k t_k \in \text{span}(S)$. Therefore, $\text{span}(T) \subset \text{span}(S)$.

Below is the proof for the second statement, which is a corollary of the first statement.

Proof.

First and foremost, notice that $S \subset \text{span}(S)$. By the first statement, $\text{span}(S) \subset \text{span}(\text{span}(S))$ holds. Moreover, notice that because $\text{span}(S)$ contains all linear combinations of S, making more linear combinations of the linear combinations of S will not add more ele-

ments. Therefore, $\text{span}(\text{span}(S)) \subset \text{span}(S)$ and $\text{span}(\text{span}(S)) = \text{span}(S)$.

An interesting fact about this property is that this applies for any subset S, finite or infinite. This is because even if $\text{span}(S)$ is infinite, it only uses finite number of vectors to create each linear combination. Continuing, let's prove the third statement.

Proof.

Notice that by the first statement, $\text{span}(T) \subset \text{span}(\text{span}(S))$. Continuing with the second statement shows that $\text{span}(T) \subset \text{span}(S)$.

Finally, here is the proof for the last statement.

Proof.

First, it is self-evident that $\text{span}\{v_1, \ldots, v_k + \lambda v_j, \ldots, v_n\} \subset \text{span}\{v_1, \ldots, v_k, \ldots, v_n\}$ since all linear combination of the set of vectors on the left can be represented as the linear combination of the set of vectors on the right. Moreover notice that v_k can be represented as $v_k = (v_k + \lambda v_j) - \lambda v_j$ even if $k = j$. Therefore, $v_k \in \text{span}\{v_1, \ldots, v_k + \lambda v_j, \ldots, v_n\}$ and $\text{span}\{v_1, \ldots, v_k, \ldots, v_n\} \subset \text{span}\{v_1, \ldots, v_k + \lambda v_j, \ldots, v_n\}$ by statement 3. Therefore, the equations in the statement hold.

With the ideas of linear combinations and the span of a set of vectors in mind, let's discuss about linear independence.

13.3.2 Linear Independence

One of the most important reasons why we use Linear Algebra is due to efficiency. As you could have guessed, most vector spaces have infinitely many vectors. Of course, there must be an efficient way of representing the vector space instead of manually listing. Determining such representative vectors involves numerous factors, and that's what we will discuss with linear independence, basis, and dimension.

Definition 13.3.10

Consider the linear combination of $V = \{v_1, \ldots, v_n\}$ in a vector space and scalars $c_1, \ldots, c_n$ that satisfy the following equation.

$$\sum_{k=1}^{n} c_k v_k = \mathbf{0}$$

The set of vectors V is **linearly dependent** if there exists scalars $c_1, \ldots, c_k$ that are not all equal to zero. The set of vectors is **linearly independent** if all scalars are zero in the equation.

Let's take a look at a quick example.

Exercise 13.3.11

Determine whether $V = \{\langle 1, 1\rangle, \langle 2, 3\rangle \in \mathbb{R}^2\}$ is linearly independent.

Solution.

By definition, for the set to be linearly independent, all scalars c_1 and c_2 must be zero for the following equation to hold.

$$c_1\langle 1, 1\rangle + c_2\langle 2, 3\rangle = \langle 0, 0\rangle$$

Writing in system of linear equations, the equation above can be written as the following.

$$c_1 + 2c_2 = 0$$
$$c_1 + 3c_2 = 0$$

Solving the equation, $c_1 = c_2 = 0$ and no other set of scalars can lead to the equation. Therefore, the set of vectors is linearly independent.

One interesting observation that we can make is that for $n \in \mathbb{Z}$, if the set of vectors in $\mathbb{R}^n$ contains more than n elements, then the set is linearly dependent. We will discuss more on this later when we

discuss matrix and its application in linear systems, but for such cases, we have more variables than linear equations, which implies that there exists infinitely many solutions. Therefore, such a set of vectors must be linearly dependent. Continuing, below is a theorem that we can deduce from the definition.

Theorem 13.3.12

A set with a finite number of nonzero vectors is linearly dependent if and only if a vector in the ordering of the set can be represented as a linear combination of vectors preceding it.

Proof.

First and foremost, the first half of the theorem can be shown. Consider the set of vectors $\{v_1, \ldots, v_{j-1}, v_j, \ldots, v_n\}$. Assume for scalars c_k, the following equation is true.

$$v_j = \sum_{k=1}^{j-1} c_k v_k$$

Therefore, $\mathbf{0} = \sum_{k=1}^{j-1} c_k v_k - v_j$. Letting $c_k = 0$ for integer $k \in (j, n]$, the following equation holds.

$$\sum_{k=1}^{j-1} c_k v_k - v_j + \sum_{k=j+1}^{n} c_k v_k = \mathbf{0}$$

Because at least one scalar is nonzero, the set of vectors $\{v_1, \ldots, v_n\}$ is linearly dependent.

Continuing, let the set of vectors $\{v_1, \ldots, v_n\}$ be linearly dependent. For the corresponding scalars c_k, let c_j be the last scalar that is nonzero. Thus, the following equation holds.

$$\sum_{k=1}^{j} c_k v_k + \sum_{k=j+1}^{n} 0 v_k = \mathbf{0}$$

Rearranging the equation, $v_k = -\sum_{k=1}^{j-1} \frac{c_k}{v_j} v_k$, which is a linear combination. Because the converse holds, the theorem is true.

From this theorem, we can deduce a corollary.

Corollary 13.3.13

For a vector space $V = \text{span}(S)$ and linearly dependent set $S = \{v_1, \ldots, v_n\} \subset V$, then there exists a set $T \subset S$ such that $|T| = n - 1$ and $V = \text{span}(T)$.

Proof.

By Theorem 13.3.12, there exists element $v_j \in S$ such that v_j is the linear combination of the previous elements as listed in S.

Let $T = \{v_1, \ldots, v_{j-1}, v_{j+1}, \ldots, v_n\}$. Consider the following equations for scalars d_k and c_k.

$$\begin{aligned} V &= \sum_{k=1}^{n} d_k v_k = \sum_{k=1}^{j-1} d_k v_k + d_j v_j + \sum_{k=j+1}^{n} d_k v_k \\ &= \sum_{k=1}^{j-1} d_k v_k + d_j \left(\sum_{k=1}^{j-1} c_k v_k \right) + \sum_{k=j+1}^{n} d_k v_k \\ &= \sum_{k=1}^{j-1} (d_j c_k + d_k) v_k + \sum_{k=j+1}^{n} d_k v_k = \text{span}(T) \end{aligned}$$

Therefore, the corollary holds.

Continuing from the corollary, the following theorems could be asserted.

Theorem 13.3.14

Consider the following statements for set S.

1. If S contains a zero vector, then it is linearly dependent.
2. If $|S| = 1$, then it is linearly dependent if and only if its element is the zero vector.
3. If $|S| = 2$, then it is linearly dependent if and only if a vector is a scalar multiple of the other.
4. If S is linearly dependent, then every set that contains S is linearly dependent.
5. If S is linearly independent, then all its subsets are linearly independent.

There's quite a bit to prove! Let's start with the first one.

Proof.

Let $S = \{\mathbf{0}, v_1, \ldots, v_n\}$. Notice that $\alpha(\mathbf{0}) + \sum_{k=1}^{n} 0v_k = \mathbf{0}$ for any scalar α. Therefore, S is linearly dependent.

Below is the proof for the second one.

Proof.

Let $S = \{v\}$. Notice that if $v \neq \mathbf{0}$, it is not possible for a nonzero scalar α to make $\alpha v = \mathbf{0}$. Therefore, if $|S| = 1$, then $v = \mathbf{0}$. For the converse, if $v = \mathbf{0}$, then any scalar α will let $\alpha v = \mathbf{0}$. Therefore the statement holds.

Here is the proof for the third statement.

Proof.

Let $S = \{v_1, v_2\}$. First, if $v_2 = \alpha v_1$ for any scalar α, then $\mathbf{0} = \alpha v_1 - \alpha v_1 = \alpha v_1 - v_2$. Therefore, S is linearly dependent. For the converse, $c_1 v_1 + c_2 v_2 = 0$ for scalars c_1 and c_2. Therefore, assuming that the scalars are nonzero without loss of generality, $v_2 = -\frac{c_1}{c_2} v_1$. Thus, the statement is true.

Below is the proof for the fourth one.

> *Proof.*
>
> Let $S = \{v_1, \ldots, v_n\}$ and $T = \{u_1, \ldots, u_n, v_1, \ldots, v_n\}$. By definition, there exists c_k that are not all zero such that $\sum_{k=1}^{n} c_k v_k = \mathbf{0}$. Because $\sum_{k=1}^{n} 0u_k + \sum_{k=1}^{n} c_k v_k = \mathbf{0}$, T is also linearly dependent.

Finally, here is the proof for the last statement.

> *Proof.*
>
> Let $T \subset S$ where S is linearly independent. Assume for the sake of contradiction that there exists a subset T that is linearly dependent. However, if T is linearly dependent, then S is linearly dependent by the previous property, which is a contradiction. Therefore, by contradiction, if S is linearly independent then all its subsets T are also linearly independent.

With this, we have concluded our discussion on linear dependence and independence. Before concluding this note, let's discuss basis and dimension of a set of vectors.

13.3.3 Basis and Dimension

As always, let's start with definition.

Definition 13.3.15

For a vector space V and its subset S, if $\text{span}(S) = V$, then S is a spanning set for V.

When we say aloud, we say that S spans V. For instance, the set $\{\langle 1, 0\rangle, \langle 0, 1\rangle\}$ spans $\mathbb{R}^2$ since all vectors $\langle a, b\rangle \in \mathbb{R}^2$ can be represented as the sum $a\langle 1, 0\rangle + b\langle 0, 1\rangle$.

Definition 13.3.16

A linearly independent subset that spans V is a **basis** for V.

Continuing from the previous example, we can see that $\{\langle 1,0\rangle, \langle 0,1\rangle\}$ not only spans $\mathbb{R}^2$, but is also a basis for $\mathbb{R}^2$ since it is linearly independent. As you can see, the same will apply if we generalize it to $\mathbb{R}^n$.

Definition 13.3.17

The **standard basis for** $\mathbb{R}^n$ is the set $S = \{e_1, \dots e_n\}$ where e_i is an element of $\mathbb{R}^n$ with 1 in its i^{th} entry and 0 elsewhere.

We could do the same for polynomials. First, notice that the set $\{1, x, x^2\}$ spans $\mathbb{P}^2$. Moreover, the set is linearly independent. Therefore, $\{1, x, x^2\}$ is a basis for $\mathbb{P}^2$.

Definition 13.3.18

The **standard basis for** $\mathbb{P}^n$ is the set $S = \{1, x, x^2, \dots, x^n\}$.

From the two definitions above, we can notice that there are finite elements for the two standard bases for $\mathbb{R}^n$ and $\mathbb{P}^n$. We call such vector spaces **finite-dimensional**.

Definition 13.3.19

A vector space with a basis of finite elements is called **finite-dimensional**. Whereas, **infinite dimensional** vector spaces do not contain a basis with finite elements.

An example of infinite dimensional vector space would be $\mathbb{P}$ since the basis is required to be an infinite set. However, we will not be discussing infinite dimensional vector spaces. From the definitions above, we could establish a theorem.

Theorem 13.3.20

For a basis S of a vector space V such that $|S| = n$, any set in V with more than n elements is linearly dependent.

Proof.

Let $S = \{v_1, \ldots, v_n\}$ and $T = \{u_1, \ldots, u_m\}$ such that $S, T \in V$ and $m > n$. Notice that because $\text{span}(S) = V$ and $T \in V$, there exists a set of constants a_{ij} that satisfies the following equation (note that we will be using this indices in the next note when we discuss matrices.).

$$\begin{aligned} u_1 &= \sum_{j=1}^{n} a_{1j} v_j \\ u_2 &= \sum_{j=1}^{n} a_{2j} v_j \\ &\vdots \\ u_m &= \sum_{j=1}^{n} a_{mj} v_j \end{aligned}$$

Therefore, it suffices to show that there exists constants $c_1, \ldots, c_m$ such that not all are zero and that satisfies the following equation.

$$\sum_{i=1}^{m} \sum_{j=1}^{n} c_i a_{ij} v_j = \mathbf{0}$$

Continuing, it suffices to show that the variables c_i and constants a_{ij} satisfy the following equations where not all c_i are zero.

$$\begin{aligned} \sum_{i=1}^{m} c_i a_{i1} &= 0 \\ \sum_{i=1}^{m} c_i a_{i2} &= 0 \\ &\vdots \\ \sum_{i=1}^{m} c_i a_{in} &= 0 \end{aligned}$$

Notice that there are m variables and n equations. It is known that

for a linear system of equations, there exists an infinite number of solutions if there are more variables than the equations (don't worry we will prove this in future notes.). Because $m > n$, there exists $c_1, \ldots c_m$ that are not all zero and that satisfy the equations for linear dependence, the theorem holds.

Continuing, we can derive three corollaries from the theorem.

Corollary 13.3.21

If a basis for the vector space V contains n elements, then any linearly independent subsets of V contain at most n elements.

Proof.

By definition, a basis of a vector space V spans V. Because any linearly dependent subsets of V contain more than n elements by Theorem 13.3.20, any linearly independent subsets of V must not contain more than n elements.

Below is the second corollary.

Corollary 13.3.22

If V is a finite vector space, every basis of V will contain the same number of elements.

Proof.

Let S and T be bases of V with p and q elements respectively. By definition, the bases of a vector space are linearly independent. By the previous corollary, it is evident that $q \leq p$ and $p \leq q$ are true. Therefore, $p = q$ and the corollary holds.

Now before continuing with our third corollary, let's first define what the dimension of a vector space is.

Definition 13.3.23

For a finite dimensional vector space V, the number of elements in any basis of V is called the **dimension** of V, denoted as $\dim(V)$.

For instance, recall the standard basis of $\mathbb{R}^n$. From the number of elements in the basis, we can write that $\dim(\mathbb{R}^n) = n$. Similarly, we can also notice that $\dim(\mathbb{P}^n) = n + 1$. Now let's continue with our third corollary.

Corollary 13.3.24

Every set S in an n-dimensional vector space such that $|S| = n + 1$ is linearly dependent.

Proof.

By definition, a basis of an n-dimensional vector space V contains n elements. By Theorem 13.3.20, since $n + 1 > n$, S must be linearly dependent.

One fun fact to note is that for the vector space $\{0\}$, we say $\dim(\{0\}) = 0$ by convention. Moreover, consider the following theorem.

Theorem 13.3.25

For a basis $S = \{v_1, \ldots, v_n\}$ of a vector space V, any vector $u \in V$ can be written as a unique linear combination of S.

Proof.

For the sake of contradiction, assume that u can be written as $u = \sum_{k=1}^{n} c_k v_k = \sum_{k=1}^{n} d_k v_k$ where $c_k \neq d_k$ for some choice of k. Therefore,

the following equations can be obtained.

$$\sum_{k=1}^{n} c_k v_k = \sum_{k=1}^{n} d_k v_k$$
$$\sum_{k=1}^{n} (c_k - d_k) v_k = 0$$

Note that by definition, S is linearly independent and $c_k - d_k = 0$ for all integer $k \in [1, n]$. Because it contradicts with the assumption that $c_k \neq d_k$ for some choice of k, the theorem holds.

Now with the theorem in mind, because we know that the linear combination of a basis to represent a vector is unique, we can establish the following definition.

Definition 13.3.26

For a basis $S = \{v_1, \ldots, v_n\}$ of a vector space V and any vector $u = \sum_{k=1}^{n} c_k v_k \in V$, the **coordinates of u with respect to** S, denoted as $(u)_S$ is the list of coefficients $c_1, \ldots, c_n$.

We often represent the coordinates with n-tuple. To see that, let's take a look at the following example.

Exercise 13.3.27

Find the coordinate of $f(x) = x^2 + 2x + 3$ with respect to the basis $S = \{x^2, 2x^2 + 1, x\}$ in the vector space $\mathbb{P}^2$.

Solution.

Consider the following equation.

$$f(x) = 3\left(2x^2 + 1\right) - 5\left(x^2\right) + 2(x)$$

Therefore, the coordinate of $f(x)$ can be represented as the following

3-tuple.

$$\begin{bmatrix} -5 \\ 3 \\ 2 \end{bmatrix}_S$$

Lastly, let's conclude our discussion on basis and dimension with a theorem.

Theorem 13.3.28

Consider the following statements for a vector space V and its subset $S \subset V$.

1. There exists a basis for V that is a subset of S if $\text{span}(S) = V$.
2. There exists a basis for V that includes all elements in S if S is linearly independent.

Let's start by proving the first statement.

Proof.

Notice that by definition, a basis of a vector space V is a linearly independent set that spans V. Therefore, it suffices to show that there exists a linearly independent set that spans V that is also a subset of S.

By Theorem 13.3.12, a subset of S is linearly dependent if any only if a vector can be represented as the linear combination of the preceding vectors. Let T be a subset of S without the vectors that can be represented as the linear combination of the preceding vectors. Notice that T spans V as the vector removed can be represented as the linear combination of the other vectors. Moreover, because T is linearly independent, there exists a basis for V that is a subset of S.

Below is the proof for the second statement.

Proof.

Let T be a basis for V and $A \coloneqq S \cup T$. By definition, $\text{span}(A) = V$.

Construct $B \in A$ by removing all vectors that can be represented as a linear combination of the preceding vectors. Notice that no elements from S will be removed as S is linearly independent. By construction, B spans V and is also linearly independent. Because $S \subset B$, the statement holds.

We discussed fundamental topics of vector spaces. Of course there are a lot more to discuss in vector spaces; however, to do so, we need some understanding of matrices. So let's pause our discussions on vector spaces for now, and move on to matrices and their applications in linear system of equations before looking at their implications in vector spaces.

§13.4 Practice Problems

1. Consider the set $S = \{(x, y, z) \in \mathbb{R}^3 \mid x = 0\}$. Identify whether the set is a vector space under the standard vector addition and scalar multiplication.

2. Determine whether $S = \{f \in \mathbb{R} \mid f'(0) = 0\}$ is a vector space for differentiable functions f.

3. For all $n \in \mathbb{Z}$ and $a_i \in \mathbb{R}$, determine whether

$$S = \left\{ (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; \sum_{k=1}^{n} a_k x_k = 0 \right\}$$

   is a vector space.

4. Show that the intersection of subspaces of a vector space V is also a subspace of V.

5. Show that for distinct real numbers $a_1, \ldots, a_n$, the set $\{e^{a_1 x}, \ldots, e^{a_n x}\}$ is linearly independent.

6. Let $A, B \in \mathbb{R}^3$ be lines defined as $x = y = z$ and $3x = 4y = 5z$ respectively. If $C = \{a + b \mid a \in A, b \in B\}$, show that $C = A \oplus B$ where $\oplus$ represents the direct sum [Erd20].

7. If $\{a, b, c\}$ is linearly independent in a vector space V, show that $\{a + b, b + c, c + a\}$ is also linearly independent [Erd20].

8. Show that the set of all polynomials in $\mathbb{P}^3$ that includes the origin is a subspace of $\mathbb{P}^3$.

Note 14

Matrices

Matrices are one of the fundamental components that build linear algebra. This note discusses the fundamental concepts and properties of matrices.

§14.1 Basic Terms and Notations

As always, let's get started with the definition.

Definition 14.1.1

A **matrix** is a 2-dimensional array composed of entries called **elements**. A matrix is said to have **order** "m by n" and written as $m \times n$ if there are m rows and n columns.

Technically, the elements of a matrix can be anything including functions. However, the majority of the matrices discussed in linear algebra will

have numeric elements. Let's take a look at some examples.

$$A = \underbrace{\left[\begin{array}{cccc} 1 & 2 & 3 & 4 \\ -1 & -2 & -3 & -4 \\ 0 & -1 & 0 & 1 \end{array}\right]}_{\text{4 columns}} \Bigg\} \text{3 rows}, \qquad B = \underbrace{\left[\begin{array}{c} \pi \\ e \\ 0 \\ 1 \end{array}\right]}_{\text{1 column}} \Bigg\} \text{4 rows}$$

Here, matrix A has order 3×4 and matrix B has order 4×1. The form of a matrix that has order $m \times n$ can be generalized as the following.

$$A = \underbrace{\left[\begin{array}{ccccc} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,n} \end{array}\right]}_{m \text{ columns}} \Bigg\} n \text{ rows}$$

It is also worth noting that matrix A is often represented as $[a_{ij}]$ and the elements as a_{ij} like a_{12} for $a_{1,2}$.

Definition 14.1.2

An element a_{ij} is said to have row index of i and column index of j.

As you probably have guessed, we assign special names to some matrices depending on m and n.

Definition 14.1.3

Square matrices are matrices with an equal number of rows and columns. The elements with the same row and column index are called **diagonal elements** of the matrix. The collection of such elements is called **main diagonal**, or **principal diagonal**.

What if either m or n is equal to 1? Yes, we do have special terms for

them.

Definition 14.1.4

Row matrices are matrices that are composed of only one row. A **column matrix** is an array composed of one column.

If you have a machine learning background, you likely have heard of the term "high dimensional arrays or vectors." Here, the term "dimension" works the same way.

Definition 14.1.5

The **dimension** of a row or column matrix is the number of **components**, or the elements. An **n-tuple** is an n-dimensional row or column vector.

Let's take a look at an example from the previous matrix B reproduced below.

$$B = \underbrace{\begin{bmatrix} \pi \\ e \\ 0 \\ 1 \end{bmatrix}}_{\text{1 column}} \left.\vphantom{\begin{bmatrix} \pi \\ e \\ 0 \\ 1 \end{bmatrix}}\right\} \text{4 rows}$$

Here, matrix B is a 4-dimensional matrix with four components. This column matrix can also be called as 4-tuple.

We could also define matrices by their elements.

Definition 14.1.6

A **zero matrix** is a matrix with all its elements as 0.

With the fundamental terms in mind, let's discuss the properties of matrices.

§14.2 Fundamental Properties

First, when are two matrices "equal" to each other? Consider the following definition for a formal explanation.

Theorem 14.2.1

Two matrices A and B with orders $m \times n$ and $p \times q$ respectively are said to be equal to each other if they retain the same order and corresponding elements, i.e. they satisfy the following properties.

1. $m = p$
2. $n = q$
3. For elements a_{ij} in A and b_{ij} in B, the following equation satisfies.

$$a_{ij} = b_{ij} \, \forall \, i \in [1, m] \text{ and } j \in [1, n]$$

Although this is very trivial and intuitive, it is always nice to formally define concepts!

Theorem 14.2.2

For matrices A and B with same order, the element in the (i, j)-entry of their sum $A + B$ is equivalent to the sum of elements in the (i, j)-entry of A and B.

Note that the orders of matrices being added must be identical. We do not consider additions with matrices of different orders. Below is an example of addition.

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & \pi & 4 \end{bmatrix} + \begin{bmatrix} -2 & 1 & 3 \\ 3 & \pi & -4 \end{bmatrix} = \begin{bmatrix} -1 & 4 & 8 \\ 5 & 2\pi & 0 \end{bmatrix}$$

Similarly, we could see that subtractions are defined in similar way. Subtracting the matrices above, we get the following results.

$$\begin{bmatrix} 1 & 3 & 5 \\ 2 & \pi & 4 \end{bmatrix} - \begin{bmatrix} -2 & 1 & 3 \\ 3 & \pi & -4 \end{bmatrix} = \begin{bmatrix} 3 & 2 & 2 \\ -1 & 0 & 8 \end{bmatrix}$$

Now, as we look at the additions and subtractions of matrices, you probably wondered about the application of additive properties. Intuitively, the fundamental properties of addition can be applied to matrices addition.

Theorem 14.2.3

For matrices A and B with order $m \times n$, $A + B = B + A$.

Proof.

Let $A = [a_{ij}]_{m\times n}$ and $B = [b_{ij}]_{m\times n}$. By definition, the sum $A + B = [a_{ij} + b_{ij}]$. Because addition is commutative, the following equation holds.

$$A + B = [a_{ij} + b_{ij}] = [b_{ij} + a_{ij}] = B + A$$

Thus, the commutative property of matrix addition holds.

In addition to the commutative property, the associative property is also applicable.

Theorem 14.2.4

For matrices A, B, and C with order $m \times n$, $A + (B + C) = (A + B) + C$.

Proof.

Let $A = [a_{ij}]_{m\times n}$, $B = [b_{ij}]_{m\times n}$, and $C = [c_{ij}]_{m\times n}$. Consider the following equation.

$$\begin{aligned} A + (B + C) &= [a_{ij}] + [b_{ij} + c_{ij}] = [a_{ij} + (b_{ij} + c_{ij})] = [a_{ij} + b_{ij} + c_{ij}] \\ &= [(a_{ij} + b_{ij}) + c_{ij}] = [a_{ij} + b_{ij}] + [c_{ij}] \\ &= (A + B) + C \end{aligned}$$

Thus, the associative property of addition applies to the matrix addition.

One obvious property in addition is that when you add 0 to a number,

you get the number itself. The same applies in matrix addition.

Theorem 14.2.5

The sum of matrices $A_{m\times n}$ and $0_{m\times n}$ yields the matrix A, i.e. the zero matrix is the identity elements of matrix addition.

Proof.

Let $A = [a_{ij}]_{m\times n}$. By definition, the sum $A + 0 = [a_{ij} + 0] = [a_{ij}] = A$. Therefore, the zero matrix is the identity element of matrix addition.

Now what if we want to add the same matrix multiple times? Fortunately, we already resolved the issue. For instance, we write $x + x + x$ as $3x$. We can multiply matrices with scalars.

Theorem 14.2.6

For a complex number λ and $A = [a_{ij}]$, $\lambda A = [\lambda a_{ij}]$.

Let's take a look at an example.

Exercise 14.2.7

Compute $2A - 3B$ for A and B defined as the following.

$$A = \begin{bmatrix} 2 & 3 & -1 \\ 1 & 7 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} -2 & 1 & 0 \\ 5 & 3 & 8 \end{bmatrix}$$

Solution.

Before subtracting, we could perform scalar multiplication first.

$$\begin{aligned} 2A - 3B &= 2\begin{bmatrix} 2 & 3 & -1 \\ 1 & 7 & 3 \end{bmatrix} - 3\begin{bmatrix} -2 & 1 & 0 \\ 5 & 3 & 8 \end{bmatrix} \\ &= \begin{bmatrix} 4 & 6 & -2 \\ 2 & 14 & 6 \end{bmatrix} - \begin{bmatrix} -6 & 3 & 0 \\ 15 & 9 & 24 \end{bmatrix} \end{aligned}$$

From here we could subtract two matrices. Therefore, $2A - 3B$ can be represented as following.

$$2A - 3B = \begin{bmatrix} 10 & 3 & -2 \\ -13 & 5 & -18 \end{bmatrix}$$

With the scalar multiplication in mind, we can expand the application of foundational concepts here with matrices.

Theorem 14.2.8

For $A = [a_{ij}]_{m\times n}$, $B = [b_{ij}]_{m\times n}$, and $\lambda_1, \lambda_2 \in \mathbb{C}$, the following properties hold.

1. $\lambda_1(A \pm B) = \lambda_1 A \pm \lambda_1 B$
2. $(\lambda_1 \pm \lambda_2)A = \lambda_1 A \pm \lambda_2 A$
3. $\lambda_1(\lambda_2 A) = (\lambda_1\lambda_2)A$

Let's prove them! Below is the proof for the first property.

Proof.

Consider the following equation.

$$\lambda_1(A \pm B) = \lambda_1([a_{ij}] \pm [b_{ij}]) = \lambda_1([a_{ij} \pm b_{ij}])$$

Note that individual elements are multiplied in scalar multiplication. Moreover, because $\lambda_1(a_{ij} \pm b_{ij}) = \lambda_1 a_{ij} \pm \lambda_1 b_{ij}$, the following equation holds.

$$\lambda_1([a_{ij} \pm b_{ij}]) = [\lambda_1 a_{ij} \pm \lambda_1 b_{ij}] = [\lambda_1 a_{ij}] \pm [\lambda_1 b_{ij}] = \lambda_1 A \pm \lambda_1 B$$

Therefore, $\lambda_1(A \pm B) = \lambda_1 A \pm \lambda_1 B$.

Now let's prove the second property.

Proof.

By definition of scalar multiplication, $(\lambda_1 \pm \lambda_2)A$ can be represented

as the following.

$$(\lambda_1 \pm \lambda_2)A = (\lambda_1 \pm \lambda_2)[a_{ij}] = [(\lambda_1 \pm \lambda_2)a_{ij}]$$

Using the distributive property, the following equation holds.

$$[(\lambda_1 \pm \lambda_2)a_{ij}] = [\lambda_1 a_{ij} \pm \lambda_2 a_{ij}] = [\lambda_1 a_{ij}] \pm [\lambda_2 a_{ij}] = \lambda_1 A \pm \lambda_2 A$$

Thus, the equation $(\lambda_1 \pm \lambda_2)A = \lambda_1 A \pm \lambda_2 A$ holds.

Finally, here is the proof to the third property.

Proof.

Consider the following equation.

$$\begin{aligned}\lambda_1(\lambda_2[a_{ij}]) &= \lambda_1[\lambda_2 a_{ij}] = [\lambda_1\lambda_2 a_{ij}] = [(\lambda_1\lambda_2)a_{ij}] \\ &= (\lambda_1\lambda_2)[a_{ij}] = (\lambda_1\lambda_2)A\end{aligned}$$

Therefore, the property holds.

We have discussed fundamental properties so far. In the next section of the note, let's discuss what matrix multiplication with intuitions and computations.

§14.3 Matrix Multiplication

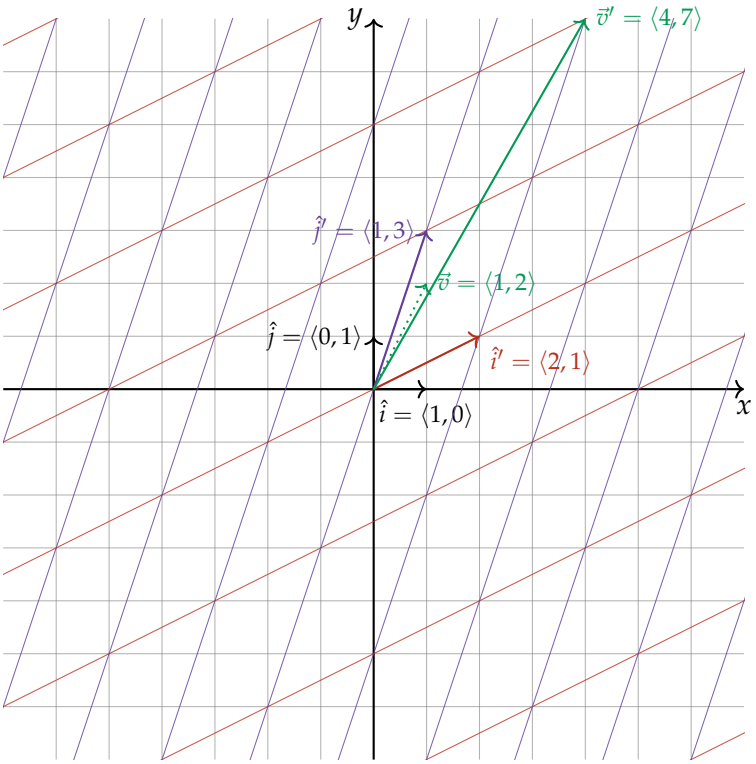
Before we get into formulas and computations, let's build our intuition on what a matrix and its multiplication is supposed to represent. We will discuss **linear transformation** in depth in latter notes, but we can think of it as a function that transforms the vector space. Matrix instructs the function how to transform the space. Let's take a look at a quick example.

Let A be a matrix that has order 2×2. The first column tells how to map

the $\hat{i}$ and the second column represents where our $\hat{j}$ is mapped.

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

Matrix A is telling us to map the standard basis $\hat{i} = \langle 1, 0 \rangle$ to $\hat{i}' = \langle 2, 1 \rangle$ and from $\hat{j} = \langle 0, 1 \rangle$ to $\hat{j}' = \langle 1, 3 \rangle$. Consider the diagram below.



After the transformation, you can imagine mapping everything in the standard vector space with respect to the new colored space. That, is what matrix multiplication does. See how a vector $\vec{v} = \langle 1, 2 \rangle$ is now on $\langle 4, 7 \rangle$. Using matrix multiplication, we can represent the mapping as the following.

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

Because we cannot draw such planes every time, let's take a look at how the vector $\vec{v} = \langle 1, 2 \rangle$ is mapped to $\langle 4, 7 \rangle$. First, from the previous note, we have seen that the first row represents the x coordinate and the second row represents the y coordinate. Now, to find where the point $(1, 2)$ is located in the "scaled" version, we could simply look at the basis vectors in the "scaled" plane. Because the vector that we are looking for is the sum of the two "scaled" basis vector, we can represent our multiplication as the following.

$$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 1 \begin{bmatrix} 2 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 2 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

Nice! We get the same result. Now that we have intuitions on what a matrix and its multiplication represents, let's take a look at algebraic properties and resulting formulas from the intuition.

14.3.1 The Method and Properties

As you can see above, matrix multiplication is not defined element-wise unlike addition and scalar multiplication. First, take a look at the following formula for individual elements.

Theorem 14.3.1

For matrices $[a_{ij}]_{m \times n}$ and $[b_{ij}]_{n \times p}$, the elements c_{ij} of the product matrix $[c_{ij}]_{m \times p}$ are defined as the following expression.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

Let's take a look at an example to see how this works.

Exercise 14.3.2

Compute the following matrix multiplication.

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 5 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 4 & 0 \end{bmatrix}$$

Solution.

Consider the following computation.

$$\begin{bmatrix} 2 & 3 & 4 \\ 1 & 5 & -1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 4 & 0 \end{bmatrix} = \begin{bmatrix} 2\cdot 1+3\cdot 2+4\cdot 4 & 2\cdot 2+3\cdot 3+4\cdot 0 \\ 1\cdot 1+5\cdot 2-1\cdot 4 & 1\cdot 2+5\cdot 3-1\cdot 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2+6+16 & 4+9+0 \\ 1+10-4 & 2+15 \end{bmatrix} = \begin{bmatrix} 24 & 13 \\ 7 & 17 \end{bmatrix}$$

It is important to note that matrix multiplication applies under a specific condition.

Theorem 14.3.3

For matrices $A_{m\times n}$ and $B_{p\times q}$, the matrices can be multiplied if and only if $n = p$. The resulting matrix will have order $m \times q$.

I believe the condition above is somewhat self-evident. When the condition above is not met, we say the product is **undefined**. Continuing from this condition, we could get an interesting property.

Theorem 14.3.4

Matrix multiplication is not commutative, i.e. for matrices A and B, it is not necessarily true that $AB = BA$.

Proof.

I will leave a more formal proof for the readers. However, I believe

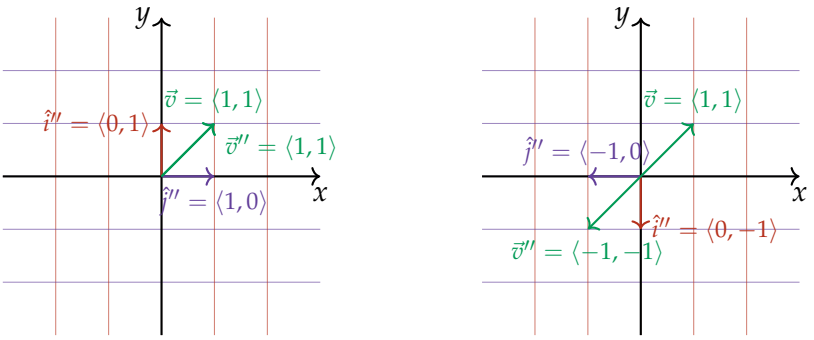
> this is self-evident as for matrices $A_{m\times n}$ and $B_{n\times p}$ where $m \neq p$, AB is defined while BA is not.

A fun way of understanding commutativity is to think about transformation questions that we see in elementary or middle school competition math. You might be familiar with questions asking orders of transformation of an object. For instance, let A be defined as the 90° rotation counterclockwise and let transformation B be reflection across the y axis. Is performing AB equal to BA? Obviously, the answer is false when you visually think about it. What if we apply the same idea here with matrices?

Notice the matrices A and B can be represented as following using matrices.

$$A = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Taking a look at the visualization, the left diagram represents AB and the right diagram visualizes BA with $\vec{v} = \langle 1, 1\rangle$ as the reference.



Let's take a look at what's happening here. $\vec{v}''$ in our left diagram can be represented as the following.

$$\vec{v}'' = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right)$$

Notice that we have to write it in the order $B(A\vec{v})$ as we would be performing A first. We can think of it as reading it from left to right

just as we would read from left to right when computing composite functions.

$$\vec{v}'' = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

This is exactly what we have for the left diagram. We could do the same for the right side.

$$\vec{v}'' = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \left(\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \left(\begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

This is also supported! Alternatively, we could just compute AB and BA to see what our "final transformation is".

$$BA = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$AB = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

Clearly, we could see that the mapping is different.

Another interesting property for matrices A, B, and C is that even if $AC = BC$, it does not necessarily mean that $A = B$. Think about the case when C is a zero matrix. Although matrix multiplication is not commutative, the associative and distributive property still applies.

Theorem 14.3.5

For matrices A, B, and C, the following equations are true given that the conditions for the addition and multiplication are met.

1. $A(BC) = (AB)C$
2. $A(B + C) = AB + AC$
3. $(B + C)A = BA + CA$
4. $\lambda(AB) = A(\lambda B)$

Let's start by proving the associative property.

Proof.

Let $A = [a_{ij}]_{m\times n}$, $B = [b_{ij}]_{n\times q}$, and $C = [c_{ij}]_{q\times r}$. First, notice that the elements of the product BC, or $D = [d_{ij}]_{n\times r}$ can be represented as $d_{ij} = \sum_{k=1}^{q} b_{ik}c_{kj}$. Multiplying it again with $[a_{ij}]$, the elements of the product $E = [e_{ij}]_{m\times r}$ can be represented as the following.

$$e_{ij} = \sum_{s=1}^{n} a_{is}d_{sj} = \sum_{s=1}^{n}\left(a_{is}\sum_{k=1}^{q} b_{sk}c_{kj}\right) = \sum_{s=1}^{n}\sum_{k=1}^{q} a_{is}b_{sk}c_{kj}$$

Similarly, we could do the same for the right-hand side of the equation. Note that the elements of the product AB, or $D' = [d'_{ij}]_{m\times q}$ can be represented as $d'_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$. Continuing, the elements in the final product $E' = [e'_{ij}]_{m\times r}$ can be written as the following.

$$\begin{aligned} e'_{ij} &= \sum_{s=1}^{q} d'_{is}c_{sj} = \sum_{s=1}^{q}\left(\left(\sum_{k=1}^{n} a_{ik}b_{ks}\right)c_{sj}\right) = \sum_{s=1}^{q}\sum_{k=1}^{n} a_{ik}b_{ks}c_{sj} \\ &= \sum_{k=1}^{q}\sum_{s=1}^{n} a_{is}b_{sk}c_{kj} = \sum_{s=1}^{n}\sum_{k=1}^{q} a_{is}b_{sk}c_{kj} = e_{ij} \end{aligned}$$

Because $e_{ij} = e'_{ij}$ for all possible combinations, $A(BC) = (AB)C$.

In my head...

You might wonder if we are allowed to say that the following equation is true.

$$\sum_{k=1}^{q}\sum_{s=1}^{n} a_{is}b_{sk}c_{kj} = \sum_{s=1}^{n}\sum_{k=1}^{q} a_{is}b_{sk}c_{kj}$$

When proving this property, I initially thought that my proof contained an error and spent my time reading the proof over and over again. This actually applies and visualizing it helped me. This part is not rigorous, so I didn't include it in the proof above, but you can

think of rotating the entire summation by 90°.

$$\sum_{k=1}^{q}\sum_{s=1}^{n} a_{is}b_{sk}c_{kj} = \begin{matrix} a_{i1}b_{11}c_{1j} \\ \vdots \\ a_{in}b_{n1}c_{1j} \end{matrix} + \cdots + \begin{matrix} a_{i1}b_{1q}c_{qj} \\ \vdots \\ a_{in}b_{nq}c_{qj} \end{matrix}$$

Similarly, the second sum can be represented as the following.

$$\sum_{s=1}^{n}\sum_{k=1}^{q} a_{is}b_{sk}c_{kj} = \begin{matrix} a_{i1}b_{11}c_{1j} \\ \vdots \\ a_{i1}b_{1q}c_{qj} \end{matrix} + \cdots + \begin{matrix} a_{in}b_{n1}c_{1j} \\ \vdots \\ a_{in}b_{nq}c_{qj} \end{matrix}$$

Now imagine you write the entire terms out, and rotate it 90°. They are identical!

Next, let's prove the first distributive law.

Proof.

Let $A = [a_{ij}]_{m\times n}$, $B = [b_{ij}]_{n\times q}$, and $C = [c_{ij}]_{n\times q}$. Consider the following equation.

$$\begin{aligned}
[a_{ij}]\left([b_{ij}] + [c_{ij}]\right) = [a_{ij}][b_{ij} + c_{ij}] &= \left[\sum_{k=1}^{n} a_{ik}(b_{kj} + c_{kj})\right] \\
&= \left[\sum_{k=1}^{n} a_{ik}b_{kj} + \sum_{k=1}^{n} a_{ik}c_{kj}\right] \\
&= \left[\sum_{k=1}^{n} a_{ik}b_{kj}\right] + \left[\sum_{k=1}^{n} a_{ik}c_{kj}\right] \\
&= [a_{ij}][b_{ij}] + [a_{ij}][c_{ij}]
\end{aligned}$$

Therefore, the distributive property $A(B + C) = AB + AC$ applies.

We could prove the right distributive law in similar fashion.

Proof.

Let $A = [a_{ij}]_{n\times q}$, $B = [b_{ij}]_{m\times n}$, and $C = [c_{ij}]_{m\times n}$. Consider the

following equation.

$$\begin{aligned}\left([b_{ij}]+[c_{ij}]\right)[a_{ij}] &= [b_{ij}+c_{ij}][a_{ij}] = \left[\sum_{k=1}^{n}(b_{ik}+c_{ik})a_{kj}\right] \\ &= \left[\sum_{k=1}^{n} b_{ik}a_{kj} + \sum_{k=1}^{n} c_{ik}a_{kj}\right] \\ &= \left[\sum_{k=1}^{n} b_{ik}a_{kj}\right] + \left[\sum_{k=1}^{n} c_{ik}a_{kj}\right] \\ &= [b_{ij}][a_{ij}] + [c_{ij}][a_{ij}]\end{aligned}$$

Thus, the right distributive property $(B+C)A = BA + CA$ holds.

Finally, let's prove the fourth property.

Proof.

Let $A = [a_{ij}]_{m\times n}$ and $B = [b_{ij}]_{n\times p}$. Consider the following equation.

$$\lambda(AB) = \lambda\left[\sum_{k=1}^{n} a_{ik}b_{kj}\right] = \left[\sum_{k=1}^{n} \lambda a_{ik}b_{kj}\right] = \left[\sum_{k=1}^{n} a_{ik}(\lambda b_{kj})\right] = A(\lambda B)$$

Thus, the fourth property holds.

In addition to the visual implications of a matrix and its multiplication above, matrix multiplication can be extended to systems of equations. For instance, consider the following system of equations.

$$\begin{aligned}2x + 3y &= 5 \\ -x + 7y &= 3\end{aligned}$$

Using **coefficient matrix**, we could represent the system above as the following matrix multiplication.

$$\begin{bmatrix}2 & 3\\ -1 & 7\end{bmatrix}\begin{bmatrix}x\\ y\end{bmatrix} = \begin{bmatrix}5\\ 3\end{bmatrix}$$

Let's take a look at more interesting facts on matrix multiplication.

Definition 14.3.6

A square matrix with 1 in all entries of its main diagonal and 0 in the other entries is an **identity matrix**, denoted as I_n for an identity matrix with order $n \times n$. Moreover, all identity matrices satisfy the following equation for square matrix A that can be multiplied.

$$AI = IA = A$$

In other word, identity matrices have the following form.

$$\begin{bmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \end{bmatrix}$$

This is called an identity matrix because when you multiply any square matrix by the identity matrix with the corresponding order, you get the square matrix itself. Consider the following example.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

From this, we can obtain an important fact about the uniqueness of the identity matrix.

Theorem 14.3.7

The identity matrix I for square matrices A is unique.

Proof.

For the sake of contradiction, let matrix $A_{m\times m}$ have two unique identity matrices $I_{m\times m}$ and $J_{m\times m}$. By definition, $AI = IA = A$ and $AJ = JA = A$. Therefore, $AI = IA = AJ = JA$ and $AI =$

> AJ. Because A is any square matrix with order $m \times m$, let $A = I$. Substituting, $II = IJ$, and $I = J$ is obtained by definition. By contradiction from the assumption that $I \neq J$, the identity matrix for a square matrix A is unique.

Another interesting insight on matrix multiplication is dissecting the matrices being multiplied. Let's take a look at a simple example.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32} \\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ b_{31} & 0 \end{bmatrix} + \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} 0 & b_{12} \\ 0 & b_{22} \\ 0 & b_{32} \end{bmatrix}$$

In the matrix multiplication above, we could see that to find the element in the entry (i, j), it is sufficient for us to see row i from the left matrix and column j from the right. In the next section of the note, we will discuss different topics about matrix.

§14.4 Additional Fundamental Topics

This section discusses frequently appearing concepts in matrices that they have their own names!

14.4.1 Transpose

Let's begin with transpose of a matrix.

Definition 14.4.1

The **transpose** of a matrix A is defined by a matrix that has switched rows and columns of A, i.e. each element is defined as following.

$$(A^\intercal)_{ij} = A_{ji}$$

Let's take a look at an example. You could think of it as holding the left top and bottom right of a matrix and rotating to see its back. For matrix A defined as the following,

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

its transpose, or A transpose is defined as the following.

$$A^\intercal = \begin{bmatrix} 1 & 5 \\ 2 & 6 \\ 3 & 7 \\ 4 & 8 \end{bmatrix}$$

As you can see, if A has the order $m \times n$, its transpose will have the order $n \times m$. Another observation that we can make is that a transpose of a row matrix is a column matrix and vice versa. Here is a quick example. Assume matrix A defined as the following.

$$A = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

The transpose of A can be represented as the following.

$$A^\intercal = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Notice that the transpose of transpose of A is simply A.

$$(A^\intercal)^\intercal = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = A$$

Before formally stating our observations, here are few definitions to note.

Definition 14.4.2

A positive integral powers of A, or multiplying the matrix n times can be represented as A^n. A^0 is defined to be the identity matrix I_n with corresponding order n. A square matrix is **nilpotent** if there exists $n \in \mathbb{Z}$ such that $A^n = 0$.

Other definitions from transpose is symmetry and skew-symmetry.

Definition 14.4.3

For matrix A and its transpose $A^\intercal$, matrix A is **symmetric** if $A^\intercal = A$ and **skew-symmetric** if $A^\intercal = -A$.

An example of a symmetric matrix is the identity matrix. For more general example, matrix A and B below are symmetric and skew-symmetric respectively.

$$A = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}, \quad B = \begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix}$$

Notice that all symmetric and skew-symmetric matrices are square matrices. Moreover, the diagonal elements must be zero for a square matrix to be skew-symmetric.

Now let's discuss the properties of transpose of a matrix.

Theorem 14.4.4

For matrix A, B, scalar λ and positive integer n, the following equations are true assuming that the addition and multiplication conditions are satisfied.

1. $(A^\intercal)^\intercal = A$
2. $(A + B)^\intercal = A^\intercal + B^\intercal$
3. $(\lambda A)^\intercal = \lambda A^\intercal$
4. $(AB)^\intercal = B^\intercal A^\intercal$
5. $(A^n)^\intercal = (A^\intercal)^n$

There are lots of properties to prove, but let's do them one by one. Below is the proof for the first property.

Proof.

Let $A = [a_{ij}]_{m\times n}$. By definition, its transpose $A^\intercal := [a_{ji}]_{n\times m}$. Continuing, $\left([a_{ji}]_{n\times m}\right)^\intercal = [a_{ij}]_{m\times n} = A$. Therefore, $(A^\intercal)^\intercal = A$.

Below is the proof for the second property.

Proof.

Let $A = [a_{ij}]_{m\times n}$ and $B = [b_{ij}]_{m\times n}$. Consider the following equation.

$$\begin{aligned}(A + B)^\intercal &= \left([a_{ij}]_{m\times n} + [b_{ij}]_{m\times n}\right)^\intercal = \left([a_{ij} + b_{ij}]_{m\times n}\right)^\intercal \\ &= [a_{ji} + b_{ji}]_{n\times m} = [a_{ji}]_{n\times m} + [b_{ji}]_{n\times m} \\ &= A^\intercal + B^\intercal\end{aligned}$$

Thus, the property holds.

Let's continue our proof for $(\lambda A)^\intercal = \lambda A^\intercal$.

Proof.

Let $A = [a_{ij}]_{m\times n}$. Because $(\lambda A)^\intercal = [\lambda a_{ij}]^\intercal_{m\times n} = [\lambda a_{ji}]_{n\times m} = \lambda[a_{ji}]_{n\times m} = \lambda A^\intercal$, the property holds.

We have two more left! Here is the proof for the fourth property.

Proof.

Let $A = [a_{ij}]_{m\times n}$ and $B = [b_{ij}]_{n\times p}$. The following equation is satisfied by definitions of matrix multiplication and transpose.

$$(AB)^\intercal = \left([a_{ij}]_{m\times n}[b_{ij}]_{n\times p}\right)^\intercal = \left(\left[\sum_{k=1}^{n} a_{ik}b_{kj}\right]_{m\times p}\right)^\intercal$$
$$= \left[\sum_{k=1}^{n} a_{jk}b_{ki}\right]_{p\times m}$$

Continuing from the right hand side, the following equation is obtained.

$$B^\intercal A^\intercal = \left[b_{ij}^\intercal\right]_{p\times n}\left[a_{ij}^\intercal\right]_{n\times m} = \left[\sum_{k=1}^{n} b_{ik}^\intercal a_{kj}^\intercal\right]_{p\times m}$$
$$= \left[\sum_{k=1}^{n} b_{ki}a_{jk}\right]_{p\times m} = \left[\sum_{k=1}^{n} a_{jk}b_{ki}\right]_{p\times m}$$

Because both the left-hand side and the right-hand side of the equation lead to the same matrix, the property holds.

Finally, below is the proof for the corollary of the fourth property.

Proof.

First, notice that by the fourth property, the following equation holds.

$$\left(A^2\right)^\intercal = A^\intercal A^\intercal = (A^\intercal)^2$$

Therefore, it suffices to show that $(A^n)^\intercal = (A^\intercal)^n$ holds for $n = k+1$ if it is true for $n = k$. Assume $(A^k)^\intercal = (A^\intercal)^k$. By the fourth property, the following equation is obtained.

$$\left(A^{k+1}\right)^\intercal = (A^kA)^\intercal = A^\intercal(A^k)^\intercal = A^\intercal(A^\intercal)^k = (A^\intercal)^{k+1}$$

By induction, the property holds for $k \geq 2$. For $k = 1$, it is self

> evident that the property holds true. Therefore, the property holds for all positive integers n.

For the next part of the section, we will discuss partitions.

14.4.2 Partitions

As always, let's start with key definitions.

Definition 14.4.5

As the name suggests a **submatrix** is a matrix of A that is obtained by omitting certain rows or columns.

Consider matrix A defined as the following.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Below are a few example of submatrices of A.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ \cancel{4} & \cancel{5} & \cancel{6} \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 7 & 8 & 9 \end{bmatrix}$$

$$A = \begin{bmatrix} \cancel{1} & \cancel{2} & \cancel{3} \\ 4 & \cancel{5} & 6 \\ \cancel{7} & \cancel{8} & \cancel{9} \end{bmatrix} \rightarrow \begin{bmatrix} 4 & 6 \end{bmatrix}$$

Definition 14.4.6

We say the matrix is **partitioned** if horizontal and vertical lines are used to divide the matrix into submatrices.

Here are a few examples of different partitions of A.

$$\left[\begin{array}{ccc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}\right], \quad \left[\begin{array}{c|cc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right], \quad \left[\begin{array}{c|c|c} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ 7 & 8 & 9 \end{array}\right]$$

From different partitions, we can obtain different **blocks**.

Definition 14.4.7

A **block** of a partition of A is a submatrix of A divided by the horizontal and vertical partition lines.

The highlighted portion of the second partition above is an example of a block. Now, why should we even care about partitions and blocks? Partitions of a matrix are important because it could be a very useful tool in computationally heavy matrix multiplication. Let's take a look at a generalized example AB.

$$AB = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix}$$

At least for me, this multiplication looks very painful. Fortunately, using partitions, we can turn this monster into a baby monster.

$$\begin{aligned} AB &= \left[\begin{array}{cc|cc} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{array}\right] \left[\begin{array}{ccc} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ \hline b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{array}\right] \\ &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} + \begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \\ a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23} \\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23} \\ a_{31}b_{11}+a_{32}b_{21} & a_{31}b_{12}+a_{32}b_{22} & a_{31}b_{13}+a_{32}b_{23} \\ a_{41}b_{11}+a_{42}b_{21} & a_{41}b_{12}+a_{42}b_{22} & a_{41}b_{13}+a_{42}b_{23} \end{bmatrix}$$

$$+ \begin{bmatrix} a_{13}b_{31}+a_{14}b_{41} & a_{13}b_{32}+a_{14}b_{42} & a_{13}b_{33}+a_{14}b_{43} \\ a_{23}b_{31}+a_{24}b_{41} & a_{23}b_{32}+a_{24}b_{42} & a_{23}b_{33}+a_{24}b_{43} \\ a_{33}b_{31}+a_{34}b_{41} & a_{33}b_{32}+a_{34}b_{42} & a_{33}b_{33}+a_{34}b_{43} \\ a_{43}b_{31}+a_{44}b_{41} & a_{43}b_{32}+a_{44}b_{42} & a_{43}b_{33}+a_{44}b_{43} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31}+a_{14}b_{41} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32}+a_{14}b_{42} & a_{11}b_{13}+a_{12}b_{23}+a_{13}b_{33}+a_{14}b_{43} \\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31}+a_{24}b_{41} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32}+a_{24}b_{42} & a_{21}b_{13}+a_{22}b_{23}+a_{23}b_{33}+a_{24}b_{43} \\ a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31}+a_{34}b_{41} & a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32}+a_{34}b_{42} & a_{31}b_{13}+a_{32}b_{23}+a_{33}b_{33}+a_{34}b_{43} \\ a_{41}b_{11}+a_{42}b_{21}+a_{43}b_{31}+a_{44}b_{41} & a_{41}b_{12}+a_{42}b_{22}+a_{43}b_{32}+a_{44}b_{42} & a_{41}b_{13}+a_{42}b_{23}+a_{43}b_{33}+a_{44}b_{43} \end{bmatrix}$$

Well, this looks good to me! If the matrices being multiplied have higher dimension, partitioning such matrices could help in computation and reducing mistakes. Now that we have seen partitions of matrices, we could derive special forms from the ideas.

14.4.3 Special Forms of Matrix

As always, let's build from definitions.

Definition 14.4.8

A **symmetrically partitioned matrix** refers to a partitioned matrix such that its arrangements of the blocks are diagonally symmetric. The **diagonal blocks** are the blocks that pass through the diagonal symmetry.

We could also define matrices with diagonal blocks that are not necessarily symmetrically partitioned.

Definition 14.4.9

A **block diagonal matrix** is a matrix with square diagonal blocks and zero on the other entries.

Below is the general form for square matrices A_i.

$$A = \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & A_n \end{bmatrix}$$

Continuing, we could define rows and columns based on its elements.

Definition 14.4.10

Zero rows and **zero columns** are rows and columns composed of zeros respectively.

For instance for matrix defined as the following,

$$\begin{bmatrix} 1 & 2 & 0 & 3 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the zero row is the second row and the zero column is the third column.

This definition is quite straightforward, and now that we have basic terms in mind, let's discuss two forms from the definitions above.

Definition 14.4.11

Matrix in **row echelon form (REF)** satisfies the conditions below.

- All nonzero leading entry is 1. (Side note here is that some textbooks and lectures do not require the leading entry to be 1, but I actually would like to keep it this way as this is how I initially learned it.)
- All nonzero leading entries in a nonzero row in a later row are to the right of nonzero leading entries in previous rows.
- If there are both nonzero and zero rows, all zero rows must appear after the nonzero rows above.

The leading entry of matrices in REF is known as a **pivot**. Similarly, matrices in **reduced row echelon form (RREF)** are matrices that satisfy the following conditions.

- All nonzero leading entries in a nonzero row is 1.
- If there are both nonzero and zero rows, all zero rows must appear after the nonzero rows above.
- All leading 1s in the succeeding rows appear in a column after the columns with previous leading 1s.
- The leading 1 must be the only nonzero element present in each column if there exists such 1.

Below are examples of a matrices in REF and RREF respectively.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Matrices that follow these forms will be very useful when solving linear systems of equations, but we will save that for the latter notes. Now let's discuss about triangular matrices.

Triangular matrices are a special type of square matrices that is determined by elements below the main diagonal. If all its elements below the main diagonal are zeros, we call it an **upper triangular** matrix. Similarly,

if all elements above the main diagonal are zeros, we call them **lower triangular** matrices. For more formal statement, consider the following definition.

Definition 14.4.12

A square matrix $A = [a_{ij}]$ is **upper triangular** if $a_{ij} = 0 \,\forall\, i > j$. Similarly, the matrix is **lower triangular** if $a_{ij} = 0 \,\forall\, i < j$.

Let's take a look at an example. The left and right matrices below are upper triangular and lower triangular matrices respectively.

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 0 & 3 & 2 & 0 \end{bmatrix}$$

Now that we have definitions of triangular matrices in mind, we can discuss a theorem from such matrices.

Theorem 14.4.13

The product of two lower triangular matrices is a lower triangular matrix, and the product of two upper triangular matrices is an upper triangular matrix.

Proof.

Let $A = [a_{ij}]_{n\times n}$ and $B = [b_{ij}]_{n\times n}$ be lower triangular matrices. Consider the following equation for their product matrix $C = [c_{ij}]_{n\times n}$.

$$C = \left[\sum_{k=1}^{n} a_{ik}b_{kj}\right] = \left[\sum_{k=1}^{j-1} a_{ik}b_{kj} + \sum_{k=j}^{n} a_{ik}b_{kj}\right]$$

From the expression, it suffices to show that $\sum_{k=1}^{j-1} a_{ik}b_{kj} + \sum_{k=j}^{n} a_{ik}b_{kj} = 0$ for $i < j$. Notice that by definition, $b_{kj} = 0$ for all $k = 1$ to $k = j - 1$. Similarly, $a_{ik} = 0$ for all integer $k \in [j, n]$. Therefore,

$c_{ij} = 0$ for all $i < j$, proving the first half of the theorem.

Similarly, let $A' = [a'_{ij}]_{n\times n}$ and $B' = [b'_{ij}]_{n\times n}$ be upper triangular matrices. The product matrix $C' = [c'_{ij}]_{n\times n}$ can be represented as following.

$$C' = \left[\sum_{k=1}^{n} a'_{ik}b'_{kj}\right] = \left[\sum_{k=1}^{j-1} a'_{ik}b'_{kj} + \sum_{k=j}^{n} a'_{ik}b'_{kj}\right]$$

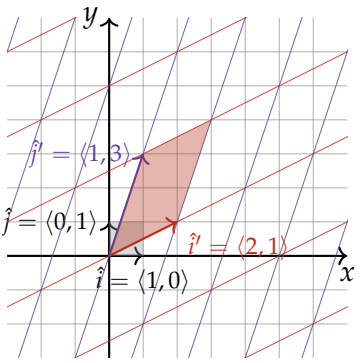
By definition, it is evident that $a'_{ik} = 0$ for integer $k \in [1, j-1]$ and $b'_{kj} = 0$ for integer $k \in [j, n]$. Therefore, for $i > j$, $c'_{ij} = 0$ and the theorem is true.

This is it for different topics on fundamentals of matrices! I know this note is quite long, but let's conclude our notes with the discussion on determinants.

§14.5 Determinants

Finally determinants! One of the most foundational concepts in linear algebra is the determinant. Being one of the foundational concepts, I also think it is a hard concept if the intuition is excluded in the discussion due to its formula heavy nature.

As you could have noticed, a linear transformation does not only change the position of the vectors. Consider the following diagram.



Notice that from the diagram above, the gray area is scaled to the new red area. Notice that the area of the new parallelogram is $3 \cdot 4 - 1 \cdot 2 \cdot \frac{1}{2} \cdot 2 - 1 \cdot 3 \cdot \frac{1}{2} \cdot 2 - 1 \cdot 2 = 12 - 2 - 3 - 2 = 5$. Because the area which was originally 1 is scaled to 5, we can say that

$$\begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} = 5$$

holds. The determinant is only defined for square matrices, and for each square matrix, the determinants tell how much the area has changed after the linear transformation. Determinants can surely be negative, and they represent the change in orientation. With this intuition in mind, let's discuss more computational perspective of determinants.

Before we discuss the formula for determinants, let's define a notation.

Definition 14.5.1

The **minor** of matrix A, denoted as M_{ij}, is a submatrix of A without the i^{th} row and j^{th} column of A. The **cofactor** of a_{ij} is defined as $(-1)^{i+j} \det(M_{ij})$.

Consider the following examples for A defined as the following.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

M_{11} and M_{24} are defined as the following.

$$M_{11} = \begin{bmatrix} a_{22} & a_{23} & a_{24} \\ a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} \end{bmatrix}, \quad M_{24} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{bmatrix}$$

Now let's define what determinants are with a recursive formula.

Definition 14.5.2

The **determinant** of a square matrix A, denoted as $\det(A)$ or $|A|$, is a value defined with the following rules.

1. $\det(A) = a_{11}$ if $A = [a_{11}]$.
2. The following equation can be used to determine $\det(A)$ for A with order $n \times n$ for $n > 1$.

$$\det(A) = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \det(M_{1j})$$

Let's take a look at an example with a 2×2 matrix $A = [a_{ij}]$.

$$\begin{aligned}
\det(A) &= \sum_{j=1}^{2} (-1)^{1+j} a_{1j} \det(M_{1j}) \\
&= (-1)^{1+1} a_{11} \det(M_{11}) + (-1)^{1+2} a_{12} \det(M_{12}) \\
&= (-1)^2 a_{11} a_{22} + (-1)^3 a_{12} a_{21} \\
&= a_{11} a_{22} - a_{12} a_{21}
\end{aligned}$$

Returning to our first example, we can verify that the formula holds.

$$\begin{vmatrix} 2 & 1 \\ 1 & 3 \end{vmatrix} = 2 \cdot 3 - 1 \cdot 1 = 5$$

We could also derive the formula for higher orders with many computations but I will leave them for you! I also think it is just better to use the recursive formula and the formula for 2×2 matrices instead of memorizing complex formulas, but I don't know. I think it's up to you. Below is another example.

$$\begin{aligned}
\det\left(\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) &= \sum_{j=1}^{3} (-1)^{1+j} a_{1j} \det(M_{1j}) \\
&= a_{11} \det(M_{11}) = 1 \cdot 1 - 0 \cdot 0 = 1
\end{aligned}$$

With these in mind, we can establish a very important theorem in computing determinants.

Theorem 14.5.3

For any square matrix A, the cofactor expansion along the i^{th} row and the cofactor expansion along the j^{th} column equal. In other words, the following equation holds.

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(M_{ij}) = \sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det(M_{ij})$$

Let's save the proof for the theorem for later as we need to show different properties of determinants before proving the theorem above. For now, let's discuss a special property of triangular matrices.

Theorem 14.5.4

For a triangular (upper or lower) matrix $A_{n\times n} = [a_{ij}]$, the determinant is defined as the following.

$$\det(A) = \prod_{i=1}^{n} a_{ii}$$

Proof.

First, the case for upper triangular matrix can be proven. Notice that $\det(A) = \sum_{i=1}^{n} (-1)^{i+1} a_{i1} \det(M_{i1}) = a_{11} \det(M_{11})$ by Theorem 14.5.3. Continuing, notice that M_{11} is another upper triangular matrix by definition.

If such calculations are continued, it is evident that the last term would be a_{nn}. Similarly, the second to the last term would be $a_{nn}a_{n-1,n-1}$. By inductive hypothesis, it is evident that

$$\det(A) = a_{11} \det(M_{11}) = a_{11} \prod_{i=2}^{n} a_{ii} = \prod_{i=1}^{n} a_{ii}$$

and the theorem holds for upper triangular matrix.

The case for lower triangular matrix can be proven in similar manner. Note that $\det(A) = \sum_{j=1}^{n}(-1)^{1+j}a_{1j}\det(M_{1j}) = a_{11}\det(M_{11})$. With the same inductive hypothesis above, it can be concluded that

$$\det(A) = a_{11}\det(M_{11}) = a_{11}\prod_{i=2}^{n}a_{ii} = \prod_{i=1}^{n}a_{ii}$$

and the theorem holds.

As you could have guessed, we get a direct corollary from the theorem.

Corollary 14.5.5

The determinant of any diagonal matrix is the product of the diagonal entries.

Proof.

First and foremost, notice that a diagonal matrix is both an upper and lower triangular matrix. Thus by Theorem 14.5.4, the corollary holds.

Continuing from triangular matrices, we can also apply our understanding of a transpose of a matrix. Consider the following theorem.

Theorem 14.5.6

$\det(A) = \det(A^\intercal)$ holds for all square matrix A.

Proof.

First and foremost, notice that $A = A^\intercal$ if A is a 1×1 matrix. Thus, the theorem holds for any $A_{1\times1}$. Continuing, note that for $A_{n\times n}$ with $n > 1$, M_{ij} equals M'_{ij} for A^T. Moreover by definition, $M'_{ij} = M_{ji}$

and the following equation holds $A^\intercal = [a'_{ij}]$.

$$\begin{aligned}\det(A) &= \sum_{i=1}^{n}(-1)^{i+1}a_{i1}\det(M_{i1}) \\ &= \sum_{j=1}^{n}(-1)^{1+j}a'_{1j}\det(M'_{1j}) \\ &= \det\left(A^\intercal\right)\end{aligned}$$

Thus, the theorem holds.

Now that we discussed the relationships between matrices and determinants, let's conclude our discussion with products and determinants.

14.5.1 Determinant and Products

Before we conclude our discussion on determinant and this note, I wished to introduce one of the most important results of determinants. In this part of the section, we will try to prove that the product of the determinants is the same as the determinant of the product. To prove this result, we would need a lot of lemmas, so let's get started with the first one.

Lemma 14.5.7

For a matrix $A_{n\times n}$ and an elementary matrix $E_{n\times n}$ for interchanging two rows, $\det(E) = -1$ and $\det(EA) = \det(E)\det(A)$.

Proof.

First and foremost, the case when E exchanges two consecutive rows can be proven. Let $B = FA$ where F is the $n \times n$ elementary matrix that interchanges the k^{th} and $k+1^{\text{th}}$ rows of A. Therefore, for $A = [a_{ij}]$ and $B = [b_{ij}]$, $b_{k+1,j} = a_{kj}$ holds and for the minor of B as M'_{ij}, $M'_{i+1,j} = M_{ij}$. Moreover, notice that $F = -1$ for $n = 2$ and by inductive hypothesis with exchanging the use of $\det(A) = (-1)^2 a_{11}\det(M_{11})$ and $\det(A) = (-1)^{2n}a_{nn}\det(M_{nn})$, it is evident

that $\det(F) = -1$. Thus, the following equations hold.

$$\begin{aligned}\det(B) &= \sum_{j=1}^{n}(-1)^{k+1+j} b_{k+1,j} \det(M'_{k+1,j}) \\ &= \sum_{j=1}^{n}(-1)^{k+1+j} a_{kj} \det(M_{kj}) \\ &= -\sum_{j=1}^{n}(-1)^{k+j} a_{kj} \det(M_{kj}) \\ &= -\det(A)\end{aligned}$$

Thus, $\det(FA) = \det(F) \cdot \det(A)$. Continuing, notice that any exchange between two consecutive rows requires one F. Moreover, any exchange between two rows with a row in between requires three Fs. Since any exchange can be performed with the combination of such two operations, it is evident that every E can be represented as odd number of Fs. Therefore, the following equations hold for odd m.

$$\begin{aligned}\det(EA) &= \det(F_1 F_2 \cdots F_m A) = \det(F_1)\det(F_2)\cdots\det(F_m)\det(A) \\ &= (-1)^m \det(A) = -\det(A) \\ &= \det(E)\det(A)\end{aligned}$$

Thus, the lemma holds.

Continuing from this lemma, we can establish an interesting corollary.

Corollary 14.5.8

The equation $\det(A) = 0$ holds for all square matrices A with either two identical rows or columns.

Proof.

Let k_1 and k_2 be the identical rows of the matrix A. For an elementary matrix E that interchanges the rows k_1 and k_2, the following

equations hold by Lemma 14.5.7.

$$\det(A) = \det(EA) = \det(E)\det(A) = -\det(A)$$

Thus, $\det(A) = 0$. Similarly, if A has two identical columns k_1 and k_2, the same argument applies by Theorem 14.5.6 as $\det(A) = \det(A^\intercal)$.

For our second lemma, we have another elementary row operation.

Lemma 14.5.9

For a matrix $A_{n\times n}$ and an elementary matrix $E_{n\times n}$ that scales k^{th} row of A by some nonzero scalar λ, $\det(E) = \lambda$ and $\det(EA) = \det(E)\det(A)$.

Proof.

First and foremost, because $E_{n\times n} = k$, it is evident that utilizing $\det(A) = (-1)^2 a_{11}\det(M_{11})$ and $\det(A) = (-1)^{2n} a_{nn}\det(M_{nn})$, appropriately will lead to $\det(E) = \lambda$ with inductive hypothesis. Moreover, consider the following equations.

$$\begin{aligned}\det(EA) &= \sum_{i=1}^{n}(-1)^{1+k}\lambda a_{ik}\det(M_{ik}) \\ &= \lambda\sum_{i=1}^{n}(-1)^{1+k}a_{ik}\det(M_{ik}) \\ &= \lambda\det(A) = \det(E)\det(A)\end{aligned}$$

Thus, the lemma holds.

Just like our first lemma, we can expand this second lemma with an interesting corollary.

Corollary 14.5.10

For any matrix $A_{n\times n}$ and scalars λ, $\det(\lambda A) = \lambda^n\det(A)$.

Proof.

Let E_k be the elementary row operation that scales the k^{th} row of A by λ. Then, the following equations hold by Lemma 14.5.9.

$$\begin{aligned}\det(\lambda A) &= \det(E_n E_{n-1} \cdots E_1 A) \\ &= \det(E_n)\det(E_{n-1}) \cdots \det(E_1)\det(A) \\ &= \lambda^n \det(A)\end{aligned}$$

Thus, the corollary holds.

Below is the third lemma on determinants and elementary matrices.

Lemma 14.5.11

For a matrix $A_{n\times n}$ and an elementary matrix E of the same order that adds λ times row k_1 to row k_2 of A, then $\det(E) = 1$ and $\det(EA) = \det(E)\det(A)$.

Proof.

First and foremost, notice that E is a triangular matrix and the elements in its diagonal are all 1. Thus by Theorem 14.5.4, $\det(E) = 1$.

Consider a new matrix B that is obtained by replacing k_2 with k_1 from A. Then, $b_{k_2 j} = a_{k_1 j}$ and the minor M'_{ij} of B satisfy $M'_{k_2 j} = M_{k_2 j}$. Moreover by Corollary 14.5.8, $\det(B) = 0$. Therefore, the following equations hold.

$$\begin{aligned}&\det(EA) \\ &= \sum_{j=1}^{n} (-1)^{k_2+j}(a_{k_2 j} + \lambda a_{k_1 j})\det(M_{k_2 j}) \\ &= \sum_{j=1}^{n} (-1)^{k_2+j}(a_{k_2 j})\det(M_{k_2 j}) + \lambda \sum_{j=1}^{n} (-1)^{k_2+j}(a_{k_1 j})\det(M_{k_2 j}) \\ &= \det(A) + \lambda \det(B) = \det(A)\end{aligned}$$

Thus $\det(EA) = \det(A) = \det(E)\det(A)$.

With the three lemmas above, we can establish an important theorem that links the determinants of a matrix and an elementary matrix.

Theorem 14.5.12

For a matrix $A_{n\times n}$ and any elementary matrix E of the same order, $\det(EA) = \det(E)\det(A)$.

Proof.

By definition, there exists three different elementary matrices that interchanges rows, scales a row, or add a scaled row to the other. By Lemmas 14.5.7, 14.5.9, and 14.5.11, $\det(EA) = \det(E)\det(A)$ for all elementary matrices E.

Now, I know it's lots of statements, but I promise we will get to the main result. Let's take a look at one last important result to get to our desired theorem! This theorem and its result include the topic of invertibility, which has not been introduced yet. Please feel free to read Section 3.3 and return or vice versa as it shouldn't be something too hard.

Theorem 14.5.13

A square matrix A is invertible if and only if $\det(A) \neq 0$.

Proof.

First, notice that A is invertible if and only if there exists a sequence of elementary matrices $E_1, \ldots, E_k$ such that $E_k \cdots E_1 A = U$ holds for some upper triangular matrix A. In other words,

$$\det(E_k) \cdots \det(E_1)\det(A) = \det(U)$$

where $\det(E_i) \neq 0$ and $\det(U) \neq 0$. Moreover, there exists such elementary matrices if and only if $\det(A) \neq 0$. Thus, the theorem holds.

Now finally! Let's prove the theorem!

Theorem 14.5.14

For square matrices A and B with the same order, $\det(AB) = \det(A)\det(B)$.

Proof.

First, the theorem can be shown for the case where A is not invertible. Notice that if A is not invertible, then AB must also be not invertible as if AB is invertible, then there exists a matrix C such that $(AB)C = I$ and $A(BC) = I$, which is contradictory. Thus by Theorem 14.5.13, $\det(AB) = 0 = 0 \cdot \det(B) = \det(A)\det(B)$ and the theorem holds for not invertible A.

Continuing, the case for invertible A can be proven. By definition, if A is invertible, then it can be represented as a product of sequence of elementary matrices $E_1, \ldots, E_k$. Therefore, $\det(AB) = \det(E_1 \cdots E_k B) = \det(E_1 \cdots E_k)\det(B) = \det(A)\det(B)$ by Theorem 14.5.12. Thus, the theorem holds.

With this important theorem in mind, let's wrap up our notes with two corollaries.

Corollary 14.5.15

For an invertible matrix A, $\det(A^{-1}) = \frac{1}{\det(A)}$.

Proof.

Notice that $1 = \det(AA^{-1}) = \det(A)\det(A^{-1})$ by Theorem 14.5.14. Therefore, $\det(A^{-1}) = \frac{1}{\det(A)}$.

Continuing, below is the second corollary.

Corollary 14.5.16

If two matrices A and B are similar, then $\det(A) = \det(B)$.

Proof.

By definition, if A and B are similar, then there exists an invertible matrix P that satisfy $A = P^{-1}BP$. Therefore by Theorem 14.5.14 and Corollary 14.5.15, $\det(A) = \det(P^{-1})\det(B)\det(P) = \frac{1}{\det(P)} \cdot \det(B)\det(P) = \det(B)$ and the corollary holds.

This is it for determinants! I think it is an interesting way to connect the relationship between linear transformations, which we will discuss in future notes and matrix representation of their effects. In the next notes, we will discuss different applications of matrices.

§14.6 Practice Problems

1. Consider matrices A and B defined as the following.

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 2 \\ 0 & 3 \end{bmatrix}$$

   Compute the following matrix operations.

   (a) $A - 2B$
   (b) AB
   (c) $(AB - BA)^\intercal$

2. Find a_2B if AB is defined as the following.

$$AB = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

3. Find matrix A such that the following equation holds.

$$2A - 3A^{\intercal} = \begin{bmatrix} -1 & -5 \\ 0 & -4 \end{bmatrix}$$

4. Show that all square matrices can be represented as the sum of symmetric and skew-symmetric matrices.

5. Show that if $AB = BA$ for nilpotent matrices A and B, then the product AB is nilpotent.

6. Show that for square matrices A and B, $AB + (AB)^{\intercal}$ is symmetric.

7. For square matrices A and B of order $n \times n$, show that the following equation holds [Erd20].

$$\det\left(\begin{bmatrix} A & B \\ B & A \end{bmatrix}\right) = \det(A + B)\det(A - B)$$

8. Show that if a square matrix A is skew-symmetric, then A^3 is also skew-symmetric.

Note 15

System of Linear Equations

One of the most important applications of matrices would be in linear systems of equations. With the ideas on matrices, we can open our third eye when viewing systems of linear equations. This note is dedicated to matrices in linear equations, Gaussian Elimination, and LU decompositions.

§15.1 Basics and Intuitions

As always, let's get started with fundamental definitions, concepts, and intuitions that will be used throughout.

First, I am pretty sure most of you are aware of what a system of equations is. However, because it is always nice to formally state our understanding, here is the definition of a linear system of equations.

Definition 15.1.1

A **system of linear equations** is composed of linear equations, or equations whose highest degree is 1. A system of m equations and n variables adhere to the following forms where a_{ij} and b_i are known quantities.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

Below is an example of a system of linear equations.

$$\begin{aligned} x + 2y + 3z &= 4 \\ -x + 3y + 2z &= 5 \end{aligned}$$

Notice that all equations above are linear, meaning, their degrees are 1. Now, what does it mean to find a solution to such systems?

Definition 15.1.2

A **solution to a system of linear equations** is a set of scalar values for unknown variables such that all linear equations are satisfied when substituted.

For instance, $(2, 1)$ is the solution to the following system of linear equations.

$$\begin{aligned} x - 2y &= 0 \\ x + y &= 3 \end{aligned}$$

To check it, you can simply substitute to see if the equations are satisfied.

Now, if you recall from the previous note where we discussed matrix

multiplication, system of linear equations can be written as a matrix multiplication.

Definition 15.1.3

A system of linear equations

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

can be written as $Ax = b$ where A, x, and b are matrices defined as the following.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}, \quad b \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix}$$

When you actually perform matrix multiplication, you can see that $Ax = b$ is identical to the system of linear equations. Now that we have a basic understanding of the relationship between matrices and linear systems of equations, let's take a look at a theorem and an important corollary.

Theorem 15.1.4

For two distinct solutions x_1 and x_2 of $Ax = b$, $\alpha x_1 + \beta x_2$ is also a solution to the linear system given that $\alpha, \beta \in \mathbb{R}$ and $\alpha + \beta = 1$.

Proof.

From Theorem 14.3.5, it was shown that $\lambda(AB) = A(\lambda B)$ for scalar

λ and matrices A and B that can be multiplied. Substituting $\alpha x_1 + \beta x_2$, the following system is obtained.

$$\begin{aligned} A(\alpha x_1 + \beta x_2) &= A(\alpha x_1) + A(\beta x_2) = \alpha(Ax_1) + \beta(Ax_2) \\ &= \alpha b + \beta b = b \end{aligned}$$

Therefore, $(\alpha x_1 + \beta x_2)$ is another solution to the system.

From this theorem, we can establish an important corollary about the number of solutions to a system of linear equations.

Corollary 15.1.5

A system of linear equations can retain 0, 1, or infinitely many solutions.

Proof.

First, it is self-evident that a system can retain 0 or 1 solution. Therefore, the fact that there exist infinite solutions if there are more than 1 solution must be proven.

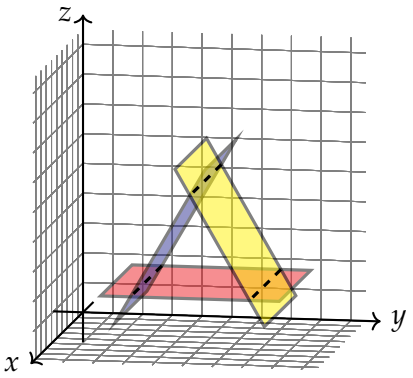
Let $f(\alpha) = \alpha x_1 + (1 - \alpha)x_2$ be a function that returns possible solutions where $x_1 \neq x_2$ are fixed values. It suffices to show that the function f is injective as there exists infinitely many values of α. Consider the following equations.

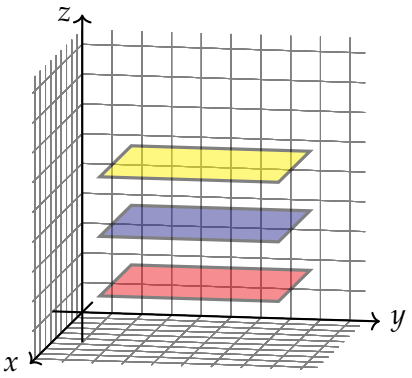
$$\begin{aligned} f(\alpha) &= \alpha x_1 + (1 - \alpha)x_2 = \alpha(x_1 - x_2) + x_2 \\ f(\alpha') &= \alpha' x_1 + (1 - \alpha')x_2 = \alpha'(x_1 - x_2) + x_2 \end{aligned}$$

If $f(\alpha) = f(\alpha')$, then $\alpha(x_1 - x_2) + x_2 = \alpha'(x_1 - x_2) + x_2$ and $\alpha = \alpha'$, proving the injectivity. Therefore, a system of linear equations can only have 0, 1, or infinitely many solutions.

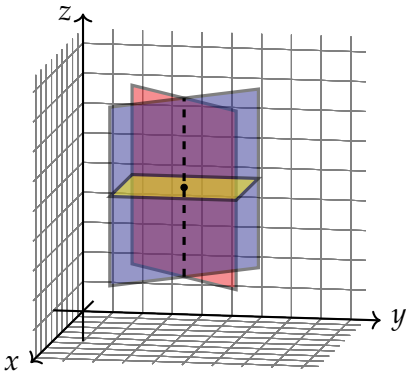
In middle school, you probably saw a visual understanding of the corollary above. Let's do the same, except in 3-dimensional space. Below are two examples of when there exists no solution. Note that not all three

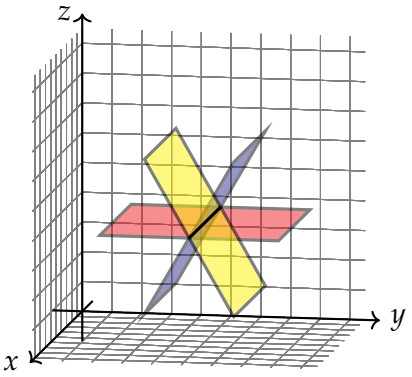
planes intersect at a single point.





Continuing, the left diagram represents the case where there is a single solution while the right diagram shows the system with infinite solutions.





Wow, not going to lie, it took some time to draw the diagrams above. Anyhow, with that in mind, let's discuss other fundamental terms and concepts before we move on to the next section of the note.

Definition 15.1.6

A linear system is **consistent** if it retains at least one solution, whereas a system is **inconsistent** if there is no solution.

We could also define a system based on the column matrix b.

Definition 15.1.7

A linear system is **homogeneous** if the column matrix b in the linear system $Ax = b$ is a zero matrix. If the matrix is not a zero matrix, the system is **nonhomogeneous**, or **inhomogeneous**.

Now from the definitions, we can see that all homogeneous systems are consistent. As you have guessed, we have a special name for a special solution in such cases.

Definition 15.1.8

Trivial solution is a solution where all the entries for x in $Ax = b$ are zero.

With the fundamental terms in mind, let's take a look at how matrices can be used to solve complex linear system with Gaussian Elimination.

§15.2 Gaussian Elimination

When you solve a linear system, you probably multiply constants and add or subtract equations to cancel variables. That is exactly what we will be doing in Gaussian Elimination, but with matrices. First, define how we solve a linear system in a more rigorous way.

Definition 15.2.1

For a system of linear equations, performing the following operations will result in a linear system with an identical solution.

1. Change the order of linear equations in which they are listed
2. Multiply a nonzero constant, or nonzero scalars, to an equation
3. Add equations from the system to form a new equation

Now before we connect this idea with matrices, let's define a matrix that we will use with the property of a linear system above.

Definition 15.2.2

An **augmented matrix** for a linear system $Ax = b$ is a partitioned matrix in the form $[A \mid b]$.

Consider the following system for an example.

$$\begin{aligned} x + 2y + 3z &= 4 \\ 2x + 3y + 4z &= 5 \\ 3x + 4y + 5z &= 6 \end{aligned}$$

In the linear system above, the augmented matrix will be the following matrix.

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{array}\right]$$

Now that we have an augmented matrix in mind, let's take a look at how the three properties that we discussed apply to matrices.

Definition 15.2.3

For an augmented matrix for a linear system, performing the following operations will result in a linear system with an identical solution as the initial system.

R_1. Rearrangement of the rows in the augmented matrix

R_2. Multiply a nonzero scalar to a row in the augmented matrix

R_3. Add a row multiplied by a scalar with another to replace a row with new row

Such operations are referred to as **elementary row operations**.

If you compare the operations one by one, you can notice that the two sets of properties refer to the same property. Now with the set of properties of augmented matrices in mind, here are the steps for the Gaussian Elimination algorithm.

Theorem 15.2.4

Execute the following steps in order for Gaussian Elimination.

1. Construct the augmented matrix.
2. Utilize the elementary row operations to transform the augmented matrix into REF.
3. Convert the augmented matrix in REF into equations of a transformed linear system, referred to as the **derived set**.
4. Back substitute to solve the system.

Let's take a look at a few examples.

Exercise 15.2.5

Find solution(s) to the following system of linear equations if any.

$$\begin{aligned} x + 2y + z &= 1 \\ 3x - y + z &= 6 \\ x + y - z &= -2 \end{aligned}$$

Solution.

First, the augmented matrix can be found from the system. Consider the matrix below.

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 3 & -1 & 1 & 6 \\ 1 & 1 & -1 & -2 \end{array}\right]$$

Using the elementary row operations, the augmented matrix can be transformed as the following.

$$\left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 3 & -1 & 1 & 6 \\ 1 & 1 & -1 & -2 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & -4 & 4 & 12 \\ 1 & 1 & -1 & -2 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & -4 & 4 & 12 \\ 0 & -1 & -2 & -3 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & -1 & 1 & 3 \\ 0 & 1 & 2 & 3 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & -1 & 1 & 3 \\ 0 & 0 & 3 & 6 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 1 & 1 \\ 0 & 1 & -1 & -3 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

Because the augmented matrix is now in REF, the transformed linear system can be rewritten.

$$\begin{aligned} x + 2y + z &= 1 \\ y - z &= -3 \\ z &= 2 \end{aligned}$$

Back substituting, it is evident that $z = 2$ and $y = -1$. Moreover, $x = 1 + 2 - 2 = 1$. Therefore, the triple $(1, -1, 2)$ is the solution to the linear system.

Try substituting to the original system! You can notice that the solution is valid. Below is the second example of using Gaussian Elimination.

Exercise 15.2.6

Find solution(s) to the following system of linear equations if any.

$$\begin{aligned} x + 2y + 3z &= 4 \\ y + 3z &= 5 \\ x + y &= 1 \end{aligned}$$

Solution.

First and foremost, the augmented matrix can be found.

$$\left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 5 \\ 1 & 1 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 5 \\ 0 & -1 & -3 & -3 \end{array}\right] \rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 3 & 4 \\ 0 & 1 & 3 & 5 \\ 0 & 0 & 0 & 2 \end{array}\right]$$

Writing the transformed augmented matrix into a system of linear

equations, the following system is obtained.

$$\begin{aligned} x + 2y + 3z &= 4 \\ y + 3z &= 5 \\ 0 &= 2 \end{aligned}$$

Notice that $0 = 2$ can never be true. Therefore, no solution exists.

We could double check our answer without using matrices. Notice that $3z - x = 4$ and $x = 3z - 4$. Similarly, $y = 5 - 3z$. Substituting to the first equation, $3z - 4 + 10 - 6z + 3z = 4$, or $6 = 4$, which is not true. Therefore, our results align! Below is the third example of using the algorithm.

Exercise 15.2.7

Find solution(s) to the following system of linear equations if any.

$$\begin{aligned} x - y - 2z &= 3 \\ 2x + y - 7z &= 3 \\ x + y - 4z &= 1 \end{aligned}$$

Solution.

Consider the following augmented matrix.

$$\left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 2 & 1 & -7 & 3 \\ 1 & 1 & -4 & 1 \end{array}\right]$$

Utilizing Gaussian Elimination, the augmented matrix can be trans-

formed as following.

$$\left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 2 & 1 & -7 & 3 \\ 1 & 1 & -4 & 1 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 2 & 1 & -7 & 3 \\ 0 & 2 & -2 & -2 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 0 & 1 & -1 & -1 \\ 2 & 1 & -7 & 3 \end{array}\right]$$
$$\to \left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 0 & 1 & -1 & -1 \\ 0 & 3 & -3 & -3 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & -1 & -2 & 3 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

Rewriting the system, the following equations are obtained.

$$\begin{aligned} x - y - 2z &= 3 \\ y - z &= -1 \end{aligned}$$

Notice that $y = z - 1$ and $x = y + 2z + 3 = z - 1 + 2z + 3 = 3z + 2$. Because there exists infinitely many z, the system has infinitely many solutions.

When writing such solutions, it could be written as the following.

$$x = \begin{bmatrix} 3z + 2 \\ z - 1 \\ z \end{bmatrix}$$

For the next section, let's discuss the inverse of a matrix and its application in linear systems.

§15.3 The Multiplicative Inverse

As we discuss matrix multiplication, you may have wondered about matrix division. I mean it would be so nice if we could just divide the coefficient matrix to find our variables. Although there is no "division" operation per se, we could achieve similar operation with the multiplicative inverse of a matrix, or simply the inverse matrix. In other words, some solutions to linear systems can be written as $x = A^{-1}b$.

Definition 15.3.1

The **inverse of a matrix** is a matrix that produces an identity matrix when multiplied with the original matrix. The inverse of A is denoted as A^{-1}.

We could see that matrices A and B are the inverse of each other if the following equation is satisfied.

$$AB = BA = I$$

Below is an example where $B = A^{-1}$.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}, \quad B = \begin{bmatrix} -5 & 2 \\ 3 & -1 \end{bmatrix}$$

One thing to note from the definition is that A and B must be square matrices. Say we have $A_{4\times 2}$ and $B_{2\times 4}$. Even if both AB and BA return an identity matrix, the order will be different. Therefore, for a matrix to have the inverse, it must be a square matrix.

Definition 15.3.2

A matrix is **invertible**, or **nonsingular**, if it retains an inverse matrix. A matrix is **singular** if it does not have an inverse matrix.

Now from the definition above, we could establish a theorem.

Theorem 15.3.3

If all diagonal elements of a matrix are nonzero constants, then the matrix is invertible.

Proof.

Consider the matrix $A_{m\times m}$ defined below.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mm} \end{bmatrix}$$

It suffices to show that there exists the inverse matrix for all values of the elements. Consider the inverse below.

$$A^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{a_{22}} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{a_{33}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{a_{mm}} \end{bmatrix}$$

Because a_{ii} for integer $i \in [1, m]$ are nonzero numbers, all elements in the matrix A^{-1} are defined, and the inverse matrix exists.

Now, it is important to note that not all square matrices have the inverse. For example, consider a matrix with zero row. The same row in the product matrices will become a zero row. By definition, it is not possible to make an identity matrix with a zero row.

Now that we discussed the existence, we could also investigate if the inverse matrix is unique just as a nonzero integer has a unique reciprocal.

Theorem 15.3.4

If there exists an inverse matrix, the inverse is unique.

Proof.

For the sake of contradiction, let B and C be different inverse of A.

Consider the following equation.

$$B = BI = B(AC) = (BA)C = IC = C$$

Because the fact that $B = C$ contradicts the initial assumption that $B \neq C$, all inverse matrices are unique.

Before we discuss how we actually find an inverse, let's discuss four important properties of inverse matrices.

Theorem 15.3.5

For invertible square matrices A and B and a nonzero scalar λ, the following equations hold.

1. $(A^{-1})^{-1} = A$
2. $(A^\intercal)^{-1} = (A^{-1})^\intercal$
3. $(AB)^{-1} = B^{-1}A^{-1}$
4. $(\lambda A)^{-1} = (\frac{1}{\lambda})A^{-1}$

There's a lot to prove and let's do them one by one starting with the first one!

Proof.

Let matrix $B = A^{-1}$. Substituting, it suffices to show that $B^{-1} = A$. By definition of inverse matrix, the equation $AB = BA = 1$ holds. In other words, A is the inverse of B and the property holds.

Below is the proof for the second property.

Proof.

Notice that $(AB)^\intercal = B^\intercal A^\intercal$ from Theorem 14.4.4. Let matrix $B = A^{-1}$ and consider the following equation.

$$I = (AB)^\intercal = B^\intercal A^\intercal = \left(A^{-1}\right)^\intercal A^\intercal$$

In other words, $(A^{-1})^\intercal$ is the inverse of $A^\intercal$, or $(A^\intercal)^{-1}$, proving the

property.

Continuing, here is the proof for the third property.

Proof.

Consider the following equation that uses the associative property of matrix multiplication.

$$\begin{aligned}(B^{-1}A^{-1})(AB) &= B^{-1}(A^{-1}A)B = B^{-1}(I)B = B^{-1}(IB)\\ &= B^{-1}B = I\end{aligned}$$

Because $B^{-1}A^{-1}$ is the inverse of AB, the property is true.

Finally, here is the proof for the last property.

Proof.

Let $B = \lambda A$. Consider the following equation.

$$\frac{1}{\lambda}A^{-1}B = \frac{1}{\lambda}A^{-1}\lambda A = A^{-1}A = I$$

In other words, $\frac{1}{\lambda}A^{-1}$ is the inverse of λA, proving the fourth property.

Now that we took a look at different properties of inverse matrices, let's discuss how to actually find one with tools to consider.

Definition 15.3.6

For a matrix A that is not necessarily a square matrix, an **elementary matrix**, denoted as E, is a square matrix that performs an elementary row operation by multiplying with A.

Let's take a look at an example. Consider matrix A defined below, and we want to perform elementary row operations of changing the order of

rows 1 and 3 and add two times the fourth row to the second respectively.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 0 & 2 & 0 \\ 2 & 4 & 6 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

The elementary row operations above can be complicated with words, but we can represent that with two elementary matrices respectively. Letting E_1 and E_2 be elementary matrices for each operation, we can represent the elementary row operations as the following.

$$E_1 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad E_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Let's multiply to see if we are right.

$$E_1 A = \begin{bmatrix} 2 & 4 & 6 & 8 \\ 1 & 0 & 2 & 0 \\ 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \quad E_2 A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 2 \\ 2 & 4 & 6 & 8 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Now we can notice three observations and tips for constructing elementary matrices. For each elementary row operation, we could do the following.

Definition 15.3.7

Methods to construct the elementary matrix for each elementary row operation:

- To interchange row a and b, write the corresponding identity matrix and interchange row a and b from the identity matrix.
- To multiply a nonzero scalar k to a^{th} row, start from the corresponding identity matrix and change the 1 from a^{th} row of the identity matrix to k.
- To add row a multiplied by the nonzero scalar k to row b, start from the corresponding identity matrix and find the b^{th} row in the matrix. Then, change the zero from the a^{th} column of the row to k.

From this, we can notice the following theorem.

Theorem 15.3.8

All elementary matrices are invertible.

Proof.

The proof is rather simple. Because all elementary row operations can be inverted with the inverse operation, all elementary matrices are invertible.

Building onto the theorem, we could prove the uniqueness.

Theorem 15.3.9

For each elementary row operation, the corresponding elementary matrix is unique.

Proof.

For the sake of contradiction, assume there exist elementary matrices E and $F \neq E$. By definition, $EA = FA$ and $A = EE^{-1}A = E^{-1}FA$. By Theorem 14.3.7, $E^{-1}F = I$ and $E = F$ is obtained by Theorem

15.3.4. By contradiction, it is shown that elementary matrices are unique.

There are more interesting lemmas and theorems, but let's discuss them in latter notes. However, let's discuss a few theorems for each elementary row operation before moving on to finding an inverse matrix.

Theorem 15.3.10

For each elementary matrix for corresponding elementary row operations, the following properties hold.

1. For an elementary matrix that changes the order of two rows, the inverse of such a matrix is itself.
2. For an elementary matrix that scales a row with nonzero constant k, the inverse can be obtained by replacing k in the elementary matrix with $\frac{1}{k}$.
3. For an elementary matrix that adds the scaled row into the other, its inverse can be obtained by replacing the nonzero scalar k with $-k$.

As always, let's start by proving the first property.

Proof.

The inverse of an elementary matrix can be considered as undoing the elementary row operation. To undo the interchanging of two rows, the two rows can be interchanged again. In other words, because the elementary row operation $EE = I$ by Theorem 14.3.7, $E = E^{-1}$.

Continuing, here is the proof for the second property.

Proof.

Let elementary matrices E and F be matrices that scale a row by a nonzero scalar k and $\frac{1}{k}$ respectively. By definition, $F(EI) = I$ and $FE = I$. Therefore, $F = E^{-1}$ and the property is shown.

Lastly, here is the proof for the final property.

Proof.

Similar to the proof for the second property, let elementary matrices E and F represent the elementary row operations that add a row scaled by nonzero constant k to another and the operation that adds a row scaled by $-k$ to another respectively. By definition, $F(EI) = I$ and $F = E^{-1}$ as demonstrated above.

Now before we actually find an inverse, we need to know which square matrices have the inverse matrix.

Definition 15.3.11

A square matrix A is invertible if and only if it can be represented as an identity matrix after sequences of elementary row operations, i.e. there exist elementary matrices E_i such that the following equation is true.

$$E_k E_{k-1} \cdots E_1 A = I$$

This is a very important result that we will prove later in a note, so don't worry. Finally, here are the steps to find an inverse if there exists one.

Definition 15.3.12

Steps to Find the Inverse Matrix of $\mathbf{A_{m \times m}}$:

1. Write an augmented matrix $[A \mid I]$ for I_m.
2. Utilize elementary row operations on the augmented matrix for the block A to be in REF.
3. If the main diagonal of the left partition contains a zero, A is singular. If not, the next step can be continued.
4. Use elementary row operations again to transform the block A in REF to an identity matrix. The right partition block is the inverse of A.

Let's take a look at a few examples.

Exercise 15.3.13

Find the inverse of $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$.

Solution.

First, the augmented matrix can be constructed.

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 4 & 5 & 6 & 0 & 1 & 0 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array}\right]$$

Consider the following elementary row operations.

$$\left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 4 & 5 & 6 & 0 & 1 & 0 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 8 & 10 & 12 & 0 & 2 & 0 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 1 & 2 & 3 & 0 & 2 & -1 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 2 & 3 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 7 & 8 & 9 & 0 & 0 & 1 \end{array}\right]$$

It is evident that the main diagonal will contain a zero after the transformation. Therefore, the matrix is not invertible.

Exercise 15.3.14

Find the inverse of the following matrix A.

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Solution.

First and foremost, the augmented matrix can be constructed.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \\ 2 & 1 & 1 & 0 & 1 & 0 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \\ 0 & 0 & -1 & -2 & 1 & -1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 2 & -1 & 1 \end{array}\right]$$

Continuing, the left partition can be transformed into an identity matrix with elementary row operations.

$$\left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 0 & 0 & 1 \\ 0 & 0 & 1 & 2 & -1 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -4 & 2 & -1 \\ 0 & 0 & 1 & 2 & -1 & 1 \end{array}\right]$$

Therefore, the inverse of A is the following.

$$A^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix}$$

We could actually multiply to see if we get I_3.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -4 & 2 & -1 \\ 2 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Indeed it is the inverse! In the next section of the note, we will discuss LU decomposition before completing our note on linear systems.

§15.4 LU Decomposition

The last popular method to solve a linear system that we will discuss is LU decomposition. As always, let's start by discussing fundamental definitions.

Definition 15.4.1

LU decomposition is the process of factorizing an invertible matrix into the product of lower and upper triangular matrices where the diagonal elements of the lower triangular matrix are 1.

Before we look at an important theorem and how it can be used to solve a linear system, here is an example of LU decomposition for matrix A defined as the following.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 6 & 9 \\ 3 & 10 & 18 \end{bmatrix}$$

From here, notice that matrix A can be transformed to upper triangular matrix with the following elementary row operations of R_3.

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 6 & 9 \\ 3 & 10 & 18 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 3 & 10 & 18 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 4 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

Notice that the operations above can be represented as the matrix multiplication with R_3 that are lower triangular matrices.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

Therefore, the following equation is obtained.

$$A = \left(\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & 3 \end{bmatrix}$$

A is now represented as the product of lower and upper triangular matrices! Now that we know how LU decomposition works, let's take a look at an important lemma on triangular matrices to prove a theorem on LU decomposition.

Lemma 15.4.2

The inverse of upper and lower triangular matrices are upper and lower triangular matrices respectively.

Proof.

First and foremost, consider an invertible upper triangular matrix $A_{m\times m} = [a_{ij}]$ and its inverse $B = [b_{ij}]$. By definition, its product $C = [c_{ij}]$ can be represented as $C = [\sum_{k=1}^{m} a_{ij}b_{ji}]$. Because $c_{ij} = 0$ for $i \neq j$ and $c_{ij} = 1$ for $i = j$, the values of b_{ji} can be found.

By definition, $a_{ij} = 0$ for $i > j$. Therefore, $c_{ij} = 0$ for $i > j$. Similarly, if $b_{ji} = 0$ for $j > i$, then $c_{ij} = 0$ for $i \neq j$. Continuing, if $b_{ij} = \frac{1}{a_{ij}}$ for $i = j$, B can be constructed, which is an upper triangular matrix. Because the inverse matrix is unique, the inverse of an upper triangular matrix is an upper triangular matrix if there exists one.

Similarly, the fact that the inverse of an lower triangular matrix is a lower triangular matrix can be proven. Let $A'_{m\times m} = [a'_{ij}]$ be a lower triangular matrix and $B' = [b'_{ij}]$ be its inverse. By definition, their

product $C' = [c'_{ij}] = [\sum_{k=1}^{m} a'_{ij} b'_{ji}]$ is the identity matrix I_m. Thus, it suffices to show that $c'_{ij} = 0$ for $i \neq j$ and $c'_{ij} = 1$ for $i = j$.

Note that by definition, $a'_{ij} = 0$ for $i < j$. Therefore, $c'_{ij} = 0$ for $i < j$. Similarly, if $b'_{ji} = 0$ for $i > j$, and $b'_{ij} = \frac{1}{a'_{ij}}$ for $i = j$, then the valid inverse matrix B is constructed. Because B is a lower triangular matrix and the inverse matrix is unique, the inverse of a lower triangular matrix is a lower triangular matrix.

With the lemma in mind, let's prove the following theorem.

Theorem 15.4.3

An invertible matrix has a LU decomposition if and only if multiplying a sequence of lower triangular elementary matrices for elementary row operations can transform the matrix into an upper triangular matrix.

Proof.

First, the statement that a nonsingular matrix has a LU decomposition if multiplying a sequence of lower triangular elementary matrices for elementary row operations can transform the matrix into an upper triangular matrix can be proven. Let A be an invertible matrix that can be represented as the following.

$$E_{n,n-1} \cdots E_{31} E_{21} A = U$$

for lower matrices E and an upper matrix U. Notice that A can be written as the following.

$$A = \left(E_{21}^{-1} E_{31}^{-1} \cdots E_{n,n-1}^{-1} \right) U$$

From Lemma 15.4.2, the inverses of the elementary matrices are lower triangular matrices. Moreover, by Theorem 14.4.13, the product of the lower triangular matrices is also a lower triangular matrix. Therefore, for a lower triangular matrix L, A can be written as the

following.

$$A = LU$$

Therefore, the first half of the statement holds.

For the second half of the statement, consider the following equation.

$$I = AA^{-1} = LUA^{-1} = L\left(UA^{-1}\right)$$

Therefore, L is nonsingular. From the proof for Lemma 15.4.2, it is evident that the diagonal elements of an invertible lower triangular matrix are nonzero as the diagonal elements in the inverse cannot be defined if a zero is present in the diagonal elements of the lower triangular matrix.

Let D be the diagonal matrix formed with the diagonal elements of L and define L' such that the equation $L'D = L$ is satisfied. Moreover, let $U' = DU$. Thus, the following equation is obtained.

$$A = LU = L'DU = L'U'$$

Recall that L is invertible. Because L^{-1} exists and $(L'D)^{-1} = D^{-1}L'^{-1}$ exists, L' is also invertible. In other words, $L'^{-1}A = U'$. Therefore, it suffices to show that L'^{-1} is a product of lower triangular elementary matrices. By definition of D, the diagonal elements of L' are 1. Thus, it can be interpreted as a sequence of elementary row operations. In other words, it can be represented as a product of lower triangular elementary matrices, proving the latter half of the theorem.

Now that we discussed what LU decomposition is, let's see how we can use it to solve a linear system.

15.4.1 Application in Linear Systems

When we apply LU decomposition in solving linear systems, two relatively easy systems must be solved. Consider the system $Ax = b$ where

$A = LU$. First, y must be found from the following system with forward substitution.

$$Ly = b$$

Then, we can solve for x with back substitution with the following system.

$$Ux = y$$

The good news is that both systems are relatively easy to solve. The reason is simple. Consider the following equation.

$$Ax = LUx = b$$

Let $y = Ux$, then $Ly = b$. If we find $Ly = b$, then we can find $Ux = y$ as we know y. This way, we can find x while exploiting the triangular form. Let's take a look at an example.

Exercise 15.4.4

Solve the following system of linear equations.

$$\begin{aligned} x - 2y + z &= 2 \\ 2x + y - z &= 1 \\ 3x - y + 2z &= 15 \end{aligned}$$

Solution.

First and foremost, the linear system can be represented as the matrix multiplication below.

$$\begin{bmatrix} 1 & -2 & 1 \\ 2 & 1 & -1 \\ 3 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 15 \end{bmatrix}$$

Continuing, the coefficient matrix A can be transformed as the

following.

$$\begin{bmatrix} 1 & -2 & 1 \\ 2 & 1 & -1 \\ 3 & -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 3 & -1 & 2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 5 & -1 \end{bmatrix}$$
$$\rightarrow \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 2 \end{bmatrix}$$

Using elementary matrices, the transformation above can be written as the following.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} A = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 2 \end{bmatrix}$$

Completing the LU decomposition, A can be represented as the product of the following lower and upper triangular matrices.

$$A = \left(\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix} \right) \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 2 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 2 \end{bmatrix}$$

Continuing, the following system could be solved.

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 1 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 15 \end{bmatrix}$$

The matrix multiplication above can be written as the following

system of linear equations.

$$\begin{aligned} x' &= 2 \\ 2x' + y' &= 1 \\ 3x' + y' + z' &= 15 \end{aligned}$$

Substituting, $y' = 1 - 4 = -3$ and $z' = 15 - 6 + 3 = 12$. Thus, the final system can be written as the following.

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 5 & -3 \\ 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ 12 \end{bmatrix}$$

Writing in linear system, x, y, and z can be found.

$$\begin{aligned} x - 2y + z &= 2 \\ 5y - 3z &= -3 \\ 2z &= 12 \end{aligned}$$

Therefore, $z = 6$, $y = 3$, and $x = 2$.

With LU decomposition, we completed our discussion on solving systems of linear equations! Below are some practice problems for the topics that we covered.

§15.5 Practice Problems

1. Show that for an invertible square matrix A, the system $Ax = b$ is always consistent.

2. Show that the following system cannot have a unique solution for all $k \in \mathbb{R}$.

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ k \end{bmatrix}$$

3. Use Gaussian Elimination to find the solution for the following system.

$$\begin{aligned} x + y + z &= 10 \\ 3x + 4y - z &= 2 \\ -x - 3y + z &= 4 \end{aligned}$$

4. Solve the same system above using LU decomposition.

5. Find the inverse of the following matrix.

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix}$$

Then, find the values of x, y, z that satisfy the following system.

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 2 \\ 4 \end{bmatrix}$$

Note 16

Matrices and Vector Spaces

Now that we discussed matrices and their application in linear systems, let's return to the vector spaces to see what matrices represent in vector spaces. This note will introduce more concepts on vector spaces and matrices like row spaces and rank of a matrix and linear transformation.

§16.1 Row Space and Rank of a Matrix

As always, let's get started with the definition.

Definition 16.1.1

The span of rows of a matrix $A_{m\times n}$ in $\mathbb{R}^n$ is the **row space** of the matrix A. The dimension of the row space is known as the **row rank**.

We can notice that by definition, the row space is a subspace of $\mathbb{R}^n$. Let's take a look at an example. Consider the following matrix A.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

The row space $\text{row}(A)$ is represented by $\text{span}\{(1,2),(3,4),(5,6)\}$ and its rank $\text{r}(A) = \dim(\text{row}(A)) = \dim(\mathbb{R}^2) = 2$. Connecting this with our knowledge in matrices, we can establish the following theorem.

Theorem 16.1.2

A basis for the row space of a matrix A in REF is the set of nonzero rows, and the number of such rows is the row rank.

Proof.

By definition, the nonzero rows span the row space since the zero rows do not contribute the vector addition. Therefore, it suffices to show that the set of nonzero vectors is linearly independent.

By definition of matrices in REF, there exist no two rows with the same initial nonzero entry, implying its uniqueness. Therefore, by induction, all coefficients in the linear combination of the set of nonzero vectors must be zero. Thus, the set of nonzero rows is a basis for the row space of A. Moreover, by definition, the rank of A is the number of nonzero rows in A.

Continuing, we can also establish the following theorem.

Theorem 16.1.3

If A and B are matrices that can be obtained by applying elementary row operations to each other, then $\text{row}(A) = \text{row}(B)$.

Proof.

Let $A = \{v_1, \ldots, v_n\}$ and B be a matrix that is obtained by a single row operation from A. For some $j, k \in [1, n]$ such that $j \neq k$, it is evident that $\text{span}\{v_1, \ldots, v_j, \ldots, v_k, \ldots, v_n\} = \text{span}\{v_1, \ldots, v_k, \ldots, v_j, \ldots, v_n\}$. Moreover, for some scalar λ, $\text{span}\{v_1, \ldots, \lambda v_k, \ldots, v_n\} = \{v_1, \ldots, v_n\}$ and $\text{span}\{v_1, \ldots, v_k + \lambda v_j, \ldots, v_n\}$ by statement four in Theorem 13.3.9. Continuing with induction, the same applies to B that is obtained by a sequence of elementary operations from A.

Thus, the theorem holds.

From the two theorems above, we can notice that for any matrix A, we can find its basis and rank by transforming it into REF with elementary row operations and observing the nonzero rows. Below is a quick example.

Exercise 16.1.4

Find the basis, row space, and rank of the following matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Solution.

First and foremost, the matrix can be transformed into REF with elementary row operations.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 7 & 8 & 9 \end{bmatrix} \to \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 7 & 8 & 9 \end{bmatrix}$$

$$\to \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & -6 & -12 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

Therefore, the basis for the row space, row space, and rank of the matrix are $\{(1,2,3),(0,1,2)\}$, span$\{(1,2,3),(0,1,2)\}$, and 2 respectively.

We could also consider the following theorem.

Theorem 16.1.5

For a vector space V and its basis $S = \{v_1, \ldots, v_n\}$ where $u, \{u_1, \ldots, u_m\} \in V$, $u \in \text{span}\{u_1, \ldots, u_m\}$ if and only if a linear combination of a coordinate representation of $\{u_1, \ldots, u_m\}$ with respect to S is the coordinate representation of u with respect to S.

Proof.

First and foremost, notice that $u \in \text{span}\{u_1, \ldots, u_m\}$ if and only if for some constants c_k, $u = \sum_{k=1}^{m} c_k u_k$. Let the coordinate representation of u_i with respect to S be $[a_{i1} \cdots a_{in}]$. Substituting, it could be inferred that $u \in \text{span}\{u_1, \ldots, u_m\}$ holds if and only if u can be represented as the following.

$$u = \sum_{i=1}^{m} \sum_{j=1}^{n} c_i(a_{ij} v_j) = \sum_{i=1}^{n} \sum_{j=1}^{m} (c_j a_{ji}) v_i$$

In other words, $u \in \text{span}\{u_1, \ldots, u_m\}$ if and only if the coordinate representation of u with respect to S is the following.

$$[u]_S = \begin{bmatrix} \sum_{j=1}^{m} (c_j a_{j1}) \\ \sum_{j=1}^{m} (c_j a_{j2}) \\ \vdots \\ \sum_{j=1}^{m} (c_j a_{jn}) \end{bmatrix} = \sum_{k=1}^{m} c_k \begin{bmatrix} a_{k1} \\ \vdots \\ a_{kn} \end{bmatrix}$$

Therefore, the theorem holds.

Continuing from this theorem, we can establish another theorem.

Theorem 16.1.6

For an n-dimensional vector space V and its basis B, consider a subset $U = \{u_1, \ldots, u_m\} \subset V$. Construct a matrix $A = [a_{ij}]$ such that the coordinate representation of u_i with respect to B is the i^{th} row of A. For some set of vectors $S \subset V$, if a basis for the row space of A is the coordinate representation of vectors in S with respect to B, then S is a basis for $\text{span}(U)$.

Proof.

First and foremost, assume that the coordinate representation of vectors in S with respect to B is a basis for $\text{row}(A)$. By definition, the span of such representations is the row space of A. Notice that by Theorem 16.1.5, an arbitrary vector $v \in \text{span}(S)$ if and only if a linear combination of a coordinate representation of S with respect to B is the coordinate representation of v with respect to B. In other words, $v \in \text{span}(S)$ if and only if the coordinate representation of v is in $\text{row}(A)$. By Theorem 16.1.5 again, the statement implies that $v \in \text{span}(U)$. Thus, $v \in \text{span}(S)$ if and only if $v \in \text{span}(U)$ and $\text{span}(S) = \text{span}(U)$.

To show that S is a basis for $\text{span}(U)$, or $\text{span}(S)$, it suffices to show that S is linearly independent. Notice that because the set of coordinate representations of S is a basis for $\text{row}(A)$, the set of coordinate representations of S is linearly independent. Moreover, by the proof for the Theorem 16.1.5, S is also linearly independent. Thus, S is a basis for $\text{span}(U)$.

There are lots of theorems and results! But here is another one that connects vectors and matrices.

Theorem 16.1.7

For $S \subset \mathbb{R}^n$ such that $|S| = k$, construct matrix $A_{k\times n}$ such that the rows are each vectors in S. S is linearly independent if and only if $\mathrm{r}(A) = k$.

Proof.

First and foremost, notice that by definition, S is linearly independent if and only if it is a basis for its span. Moreover, by construction, $\mathrm{span}(S) = \mathrm{row}(A)$ and S is linearly independent if and only if it is a basis for the row space of A. Continuing, S is a basis for the row space of A if and only if $\mathrm{r}(A) = |S|$ by definition. Therefore, S is linearly independent if and only if $\mathrm{r}(A) = |S|$.

To be honest, I procrastinated a bit when I was learning to prove the theorems above. For some reason, the proofs were hard to understand and less intuitive than the proofs that I have done through olympiad style problems. I think what helped me to understand the proofs is to just continue looking at it over and over again until I fully knew the terms. Because the proofs above prioritize understanding of the basic concepts over creativity, I think procrastinating helped me fully understand the proofs.

So far, we discussed about row spaces and row ranks. How about columns? As you may have guessed, we could do the same for columns!

Definition 16.1.8

The span of the columns of a matrix $A_{m\times n}$ is the **column space** of A, denoted as $\mathrm{col}(A)$. The dimension of such space is **column rank** of A.

We could notice that because rows and columns of a matrix are not necessarily the same, the row space and column space of a matrix are generally different. One special example in which they are the same would be identity matrices.

Now when we discussed about row rank of a matrix A, we used the notation $\text{row}(A)$. That is because the row rank and column rank of a matrix are always equal.

Theorem 16.1.9

The row rank and column rank of a matrix A are always equal.

Proof.

First and foremost, notice that by definition, $\text{row}(A^\intercal) = \text{col}(A)$ and $\text{row}(A) = \text{col}(A^\intercal)$. Therefore, the following equations are true.

$$\dim(\text{row}(A^\intercal)) = \dim(\text{col}(A))$$
$$\dim(\text{row}(A)) = \dim(\text{col}(A^\intercal))$$

Notice that if $\dim(\text{col}(A)) \leq \dim(\text{row}(A))$, then the following expression holds.

$$\dim(\text{row}(A)) = \dim(\text{col}(A^\intercal)) \leq \dim(\text{row}(A^\intercal)) = \dim(\text{col}(A))$$

Therefore, if $\dim(\text{col}(A)) \leq \dim(\text{row}(A))$, then $\dim(\text{row}(A)) \leq \dim(\text{col}(A))$ and $\dim(\text{col}(A)) = \dim(\text{row}(A))$. Thus, it suffices to show that $\dim(\text{col}(A)) \leq \dim(\text{row}(A))$.

Consider a matrix $A = [a_{ij}]$ that has the order $m \times n$. Notice that each row can be represented as an n-tuple. Let $U = \{u_1, \ldots, u_k\} \subset \mathbb{R}^n$ be a basis for $\text{row}(A)$. By definition, each row A_i of A can be written as the following for some constants c_{ij}.

$$A_i = \sum_{j=1}^{k} c_{ij} u_j$$

Moreover, notice that each element can be written as the following.

$$a_{ij} = \sum_{n=1}^{k} c_{in} u_{nj}$$

Continuing, let $V = \{v_1, \ldots, v_k\}$ be the set of such coefficients where v_i is defined as the following.

$$v_i = \begin{bmatrix} c_{1i} \\ c_{2i} \\ \vdots \\ c_{mi} \end{bmatrix}$$

Therefore, each column of A can be written as the following m-tuple.

$$\begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix} = \sum_{n=1}^{k} \begin{bmatrix} c_{1n} \\ c_{2n} \\ \vdots \\ c_{mn} \end{bmatrix} u_{nj} = \sum_{n=1}^{k} u_{nj} v_n$$

Because each column is in the span of V, $\dim(\text{col}(A)) \leq k = \dim(\text{row}(A))$. Thus, the theorem holds.

Now that we have column space in mind, let's return to linear systems. Let's first start with an example before looking at the theorem.

$$\begin{aligned} x + 2y + 3z &= 4 \\ x - y + z &= -1 \\ 4x + 3y + 2z &= 1 \end{aligned}$$

Notice that we can write the system as the following.

$$x \begin{bmatrix} 1 \\ 1 \\ 4 \end{bmatrix} + y \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} + z \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 4 \\ -1 \\ 1 \end{bmatrix}$$

This splits the coefficient matrix A in $Ax = b$ to its columns. Now the left-hand side can be interpreted as the linear combination of the columns of A, or an element in $\text{col}(A)$. In other words, the solution exists if $b \in \text{col}(A)$ and the system is inconsistent if $b \notin \text{col}(A)$. If $b \in \text{col}(A)$,

then the rank of the augmented matrix $[A \mid b]$ will equal $\mathrm{r}(A)$ since $\mathrm{col}(A) = \mathrm{col}([A \mid b])$. However, if $b \notin \mathrm{col}(A)$, then $\mathrm{r}([A \mid b]) = \mathrm{r}(A) + 1$ since we need an extra vector for the basis for the column space. We can now write this idea as a theorem.

Theorem 16.1.10

A linear system $Ax = b$ has at least one solution if and only if $\mathrm{r}(A) = \mathrm{r}([A \mid b])$.

Proof.

For a linear system $Ax = b$, let $A_{m\times n} = [a_{ij}]$, A_i be the i^{th} column of A, $x = [x_i]$, and $b = [b_i]$ where x and b are m-tuples. Notice that the system can be represented as the following.

$$\sum_{k=1}^{n} x_k A_k = b$$

The system will hold if and only if b is in the span of the columns of A. Continuing, $\mathrm{r}(A) = \mathrm{r}([A \mid b])$ if and only if b is in the span of the existing columns of A. Thus, a solution to the linear system exists if and only if $\mathrm{r}(A) = \mathrm{r}([A \mid b])$.

Though satisfying, the theorem may not be significantly useful since we anyways have to write the matrices in REF. However, we can notice another interesting result.

Theorem 16.1.11

For a consistent system with n variables, if $\mathrm{r}(A) = k$, then the final solutions can be written with $n - k$ free variables.

Proof.

First and foremost, notice that because the system $Ax = b$ is consistent and $\mathrm{r}(A) = k$, $\mathrm{r}([A \mid b]) = k$. By Theorem 16.1.2, there exist k nonzero rows, or k variables with set values for $\mathrm{r}([A \mid b])$ in REF.

Therefore, there exist $n - k$ free variables.

Continuing with this theorem, we can set a corollary for homogeneous systems.

Corollary 16.1.12

For a homogeneous system $Ax = \mathbf{0}$ with n variables, there exist nontrivial solutions if and only if $\mathrm{r}(A) \neq n$.

Proof.

By Theorem 16.1.11, there exist free variables if $\mathrm{r}(A) < n$, which are not necessarily trivial solutions. However, if $\mathrm{r}(A) = n$, then there exist zero free variables and a unique solution, which is the trivial solution.

Continuing from the system, we could also apply our knowledge to invertibility of a square matrix. Now before we prove an important theorem on the relationship between rank and invertibility, let's prove a property that we couldn't prove in our previous notes. In our previous notes, we proved the uniqueness of the inverse of a square matrix. Moreover, we also knew that if $AB = BA = I$, then A and B are inverse of each other. What if we truncate the equation to $AB = I$? The answer is that they imply the same thing! Indeed for two square matrices A and B with the same order, if $AB = I$, then $BA = I$. To prove this, let's first prove two lemmas.

Lemma 16.1.13

For two square matrices A and B with order $n \times n$ such that $AB = I$, $\mathrm{r}(A) = n$.

Proof.

First and foremost, notice that for all integers $i, j \in [1, n]$, the following equation is satisfied for A_i the i^{th} row of A and B_j the j^{th} column

of B.

$$A_i B_j = I_{ij}$$

Moreover, notice that if $\sum_{i=1}^{n} c_i A_i = \mathbf{0}$ for some constants c_i, then $\sum_{i=1}^{n} c_i A_i B_j = \mathbf{0}$ for any choice of j. By definition, $A_i B_j = 1$ if and only if $i = j$. In other words, if $\sum_{i=1}^{n} c_i A_i = \mathbf{0}$, then $c_i = 0$ for all integer $i \in [1, n]$. Therefore, the rows of A are linearly independent. By definition, the rows compose the basis of $\text{row}(A)$ and $\text{r}(A) = n$.

Continuing, let's prove the second lemma.

Lemma 16.1.14

For a square matrix $A_{n \times n}$, if $\text{r}(A) = n$, then there exists a matrix B where $BA = I$.

Proof.

Notice that because $\text{r}(A) = n$, none of the elements in the main diagonal for A in REF is zero after applying series of elementary row operations. From A in REF, more elementary row operations can be applied to convert A in REF to an identity matrix. Thus, the lemma holds.

Now that we have the two lemmas in mind, let's prove the main result.

Theorem 16.1.15

For square matrices A and B with the order $n \times n$, if $AB = I$, then $BA = I$.

Proof.

First and foremost, by Lemma 16.1.13, $\text{r}(A) = n$. Moreover by Lemma 16.1.14, there exists matrix C such that $CA = I$. Continuing, $C = CI = CAB = IB = B$. Thus, if $AB = I$, then $BA = I$.

This is the proof for the definition that we skipped in the earlier notes! Now with the two lemmas and the theorem in mind, we can derive the following result.

Theorem 16.1.16

A square matrix $A_{n\times n}$ is invertible if and only if $\mathrm{r}(A) = n$.

Proof.

First and foremost, the first half of the theorem can be proven. By definition, if A is invertible, then there exists a square matrix of the same order B such that $AB = I$. By Lemma 16.1.13, $\mathrm{r}(A) = n$.

Continuing, by Lemma 16.1.13, if $\mathrm{r}(A) = n$, then there exists a matrix C such that $CA = I$. By Theorem 16.1.15 and definition of the inverse matrix, A is invertible.

This is it for row space and rank of a matrix! In the next section, we will discuss about linear transformations, an important topic in Linear Algebra.

§16.2 Linear Transformations

When we discussed about matrices, we saw that they kind of "change" the vector space. When I first learned about matrices in an academy as a middle school student, all I had in mind was "this is an interesting way of organizing items!" I knew what vectors and matrices were, but I didn't really know them. That changed until I learned about linear transformations. Of course, I wouldn't expect my middle school self to learn all these and truly understand what matrices represent in a vector space, but this moment, or section really changed my understanding. Like all the other sections, this is an interesting one, and let's get started!

16.2.1 Brief Review of Functions

Before we get into linear transformations, let's review about functions. I think this would be a good refresher on the terms.

Definition 16.2.1

A **function** is a special **relation** where each input from its domain has exactly one corresponding output.

For instance, a function $f : X \to Y$ is a special relation whose domain is X and whose corresponding outputs lie in Y. As you probably know, there are special terms for X and Y.

Definition 16.2.2

The set of all inputs of a function is called **domain** of the function. The set of all corresponding outputs is known as **image**, or range, of the function denoted as $\Im(f)$.

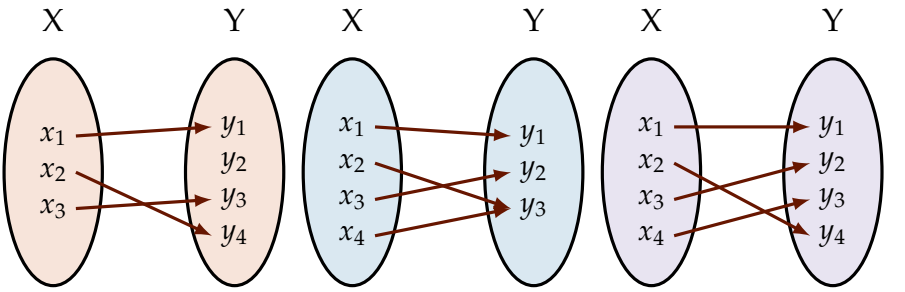
You might be familiar with the term **range**, but **image** and **range** refer to the same thing, and **image** is more commonly used in linear algebra and other courses. Now let's review about bijection, injection, and surjection.

Definition 16.2.3

A function is **injective**, or one-to-one, if each element in the domain maps to a unique element in the image. A function is **surjective**, or onto, if the image and codomain are identical. A **bijective** functions are functions that are both injective and surjective.

For more visual representation, below is a diagram that manifests the definition above.

Injective Function	**Surjective Function**	**Bijective Function**
If $f(x_1) = f(x_2)$, then $x_1 = x_2$	$\forall\, y_i \in Y,\ \exists\, x_j \in X$ $s.t.\ f(x_j) = y_i$	$\forall\, y_i \in Y,\ \exists!\, x_j \in X$ $s.t.\ f(x_j) = y_i$



Below is a fun problem on bijectivity.

Exercise 16.2.4

Show that $f : \mathbb{R} \to \mathbb{R}$ is bijective if $f(f(x)) = x$.

Solution.

First, the injectivity can be proven. Let $f(a) = f(b)$ and consider the following equation.

$$a = f(f(a)) = f(f(b)) = b$$

Therefore, if $f(a) = f(b)$, then $a = b$ and the function is injective. Continuing, let $f(x) = y$. Substituting, $f(y) = x$ is true for all $x \in \mathbb{R}$. Because the image is $\mathbb{R}$, the function is bijective.

Now that we reviewed about functions, let's discuss linear transformation.

16.2.2 Basic Terms and Notations

If we have functions over different domains, can we have a function whose domain and range are vector spaces? The answer is yes, and we have a special name for such functions.

Definition 16.2.5

A function whose domain and range are defined over vector spaces is known as a **transformation**, denoted as T.

A transformation that maps a vector space V to W is denoted as $T : V \to W$. Just like how we have $f(x)$, we do the same like $T(v)$ for $v \in V$, but it is more common to write Tv. Continuing, the definitions for image, injection, surjection, and bijection all apply in similar fashion.

There are many different transformations, but some transformation moves the vector space in "parallel" in a way that preserves the fundamental definition of vector addition and scalar multiplication.

Definition 16.2.6

A **linear transformation** is a transformation that preserves the definition of vector addition and scalar multiplication of the input vector space. In other words, for scalars α and β and input vectors v_1 and v_2, a transformation T is linear if the following equation holds.

$$T(\alpha v_1 + \beta v_2) = \alpha T(v_1) + \beta T(v_2)$$

One way to think of the equation above is preservation of the **linear** combination. Let's take a look at an example.

Exercise 16.2.7

Show that a transformation defined as $T(v) = 2v$ is linear.

Solution.

Consider the following equation for some scalars α and β.

$$T(\alpha v_1 + \beta v_2) = 2(\alpha v_1 + \beta v_2) = 2\alpha v_1 + 2\beta v_2 = \alpha T(v_1) + \beta T(v_2)$$

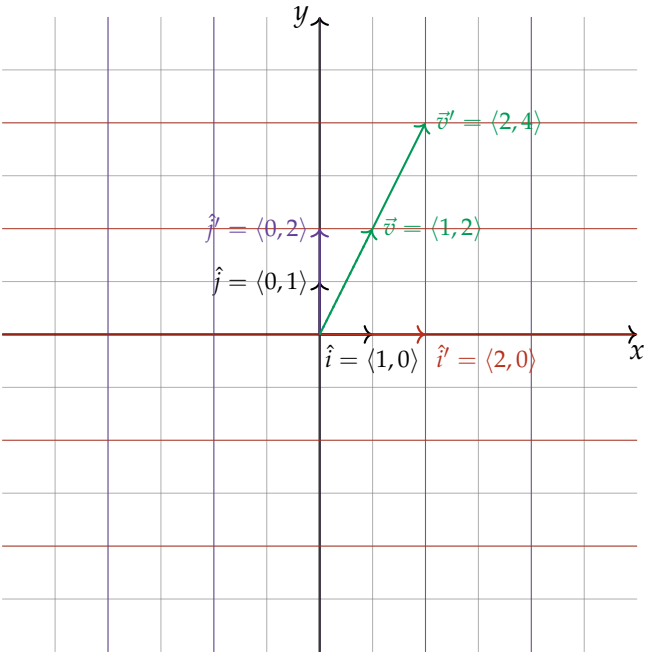
Thus, the transformation is linear.

As you can see above, if $T : \mathbb{R}^2 \to \mathbb{R}^2$, you can imagine the vectors dilating. Indeed, such linear transformations are called **dilation**

Definition 16.2.8

A **dilation** is a linear transformation that scales the vectors in the input vector space defined as $Tv = kv$ for some scalar k.

Let's take a look at the visual representation of the simple version of the example above. Consider the transformation defined as $T(\langle a, b\rangle) = \langle 2a, 2b\rangle$. We know from the previous example that the transformation is linear. Now let's try and graph it.



There are other special linear transformations that output the same vector space or the zero vector space.

Definition 16.2.9

A **zero transformation** is a linear transformation defined as $Tv = \mathbf{0}$. An **identity transformation** is a linear transformation that returns the same input vector as the output.

Let's take a look at a few more examples on linear transformations.

Exercise 16.2.10

Determine whether a transformation defined as $Tv = v^2$ is linear.

Solution.

Consider the following equation for arbitrary scalars α and β.

$$T(\alpha v_1 + \beta v_2) = (\alpha v_1 + \beta v_2)^2$$

Notice that the following equation holds.

$$= \alpha T(v_1) + \beta T(v_2) = \alpha v_1^2 + \beta v_2^2$$

Because $(\alpha v_1 + \beta v_2)^2 \neq \alpha v_1^2 + \beta v_2^2$ in general, the transformation is not linear.

Below is another example.

Exercise 16.2.11

Determine whether a transformation defined as $Ty = \frac{dy}{dx}$ for some differentiable function $y = f(x)$ is linear.

Solution.

Consider the following equation for some scalars α and β.

$$\begin{aligned} T(\alpha y_1 + \beta y_2) &= \frac{d}{dx}(\alpha y_1 + \beta y_2) = \frac{d}{dx}(\alpha y_1) + \frac{d}{dx}(\beta y_2) \\ &= \alpha \frac{d}{dx} y_1 + \beta \frac{d}{dx} y_2 = \alpha T(y_1) + \beta T(y_2) \end{aligned}$$

Thus, the transformation is linear.

One thing that we can notice from the problem above is that the transformation is not linear since changing the constants of an input will not change the output. The transformation is also not surjective as not all functions can be represented as the derivative of other functions. Continuing, we can consider the following transformation.

Exercise 16.2.12

Determine whether a transformation defined as $T(\langle a, b\rangle) = \langle -a, -b\rangle$ is linear.

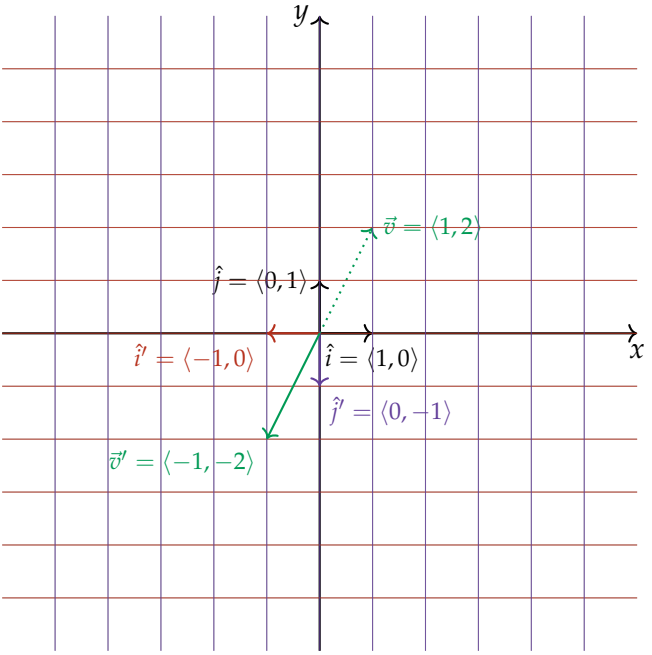
Solution.

Consider the following equation for arbitrary scalars α and β.

$$\begin{aligned} & T(\alpha\langle a, b\rangle + \beta\langle a', b'\rangle) \\ &= T(\langle \alpha a, \alpha b\rangle + \langle \beta a', \beta b'\rangle) = T(\langle \alpha a + \beta a', \alpha b + \beta b'\rangle) \\ &= \langle -\alpha a - \beta a', -\alpha b - \beta b'\rangle = \langle -\alpha a, -\alpha b\rangle + \langle -\beta a', -\beta b'\rangle \\ &= \alpha\langle -a, -b\rangle + \beta\langle -a', -b'\rangle = \alpha T(v_1) + \beta T(v_2) \end{aligned}$$

Thus, the transformation is linear.

Let's try and graph this to visually see how the vector space is transformed.



This transformation reflects the vectors across the origin! Now that we

discussed what a linear transformation is, let's see how matrices relate to it.

16.2.3 Matrices and Linear Transformation

Recall that every vector in a vector space can be represented as a linear combination of the basis. We will apply our understanding of a linear transformation to vector representations in a vector space to find the relation.

Consider a basis $S = \{v_1, \ldots, v_n\}$ for a vector space V. Notice that by definition, an arbitrary vector $v \in V$ can be represented as a linear combination of a basis as shown below (where c_k are constants).

$$v = \sum_{k=1}^{n} c_k v_k$$

Therefore, the following equation holds.

$$(v)_S = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Now returning to the definition of a linear transformation, we can consider the following equation.

$$Tv = T\left(\sum_{k=1}^{n} c_k v_k\right) = \sum_{k=1}^{n} c_k T v_k$$

Notice that the coordinate is preserved. In other words, the following equation holds for the new basis $S' = \{Tv_1, \ldots, Tv_n\}$

$$(Tv)_{S'} = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

This could let us geometrically understand that a linear transformation

"preserves" the general shape of a vector space since the new vectors have the same coordinates as the original ones, but with different basis. With this in mind, we can establish two important theorems.

Theorem 16.2.13

For a vector space V and a basis $S = \{v_1, \ldots, v_n\}$, the transformation $T : V \to \mathbb{R}^n$ defined as $Tv = (v)_S$ is linear and bijective.

Proof.

First, the linearity of the transformation could be proven. Consider the following equations for $u_1, u_2 \in V$ and some constants c_k and d_k.

$$\begin{aligned} T(\alpha u_1 + \beta u_2) &= T\left(\alpha \sum_{k=1}^{n} c_k v_k + \beta \sum_{k=1}^{n} d_k v_k\right) \\ &= T\left(\sum_{k=1}^{n} (\alpha c_k + \beta d_k) v_k\right) \\ &= \begin{bmatrix} \alpha c_1 + \beta d_1 \\ \alpha c_2 + \beta d_2 \\ \vdots \\ \alpha c_n + \beta d_n \end{bmatrix} = \begin{bmatrix} \alpha c_1 \\ \alpha c_2 \\ \vdots \\ \alpha c_n \end{bmatrix} + \begin{bmatrix} \beta d_1 \\ \beta d_2 \\ \vdots \\ \beta d_n \end{bmatrix} = \alpha \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} + \beta \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \\ &= \alpha T(u_1) + \beta T(u_2) \end{aligned}$$

Thus, T is linear. Continuing, notice that if $T(u_1) = T(u_2)$, then $(u_1)_S = (u_2)_S$ and $u_1 = u_2$. Therefore, the transformation is injective. The transformation is also surjective as the entries c_k in the coordinates are arbitrary real numbers. Thus, T is linear and bijective.

Continuing, below is an important theorem and definition.

Theorem 16.2.14

For vector spaces V and W with basis $B = \{v_1, \ldots, v_n\}$ and $C = \{w_1, \ldots, w_m\}$ respectively, there exists a matrix A such that the following equation holds for a linear transformation $T : V \to W$ and arbitrary $v \in V$.

$$(Tv)_C = A(v)_B$$

Proof.

First and foremost, consider the following equation for any $Tv_j \in W$ and constant a_{ij}.

$$Tv_j = \sum_{i=1}^{m} a_{ij} w_i$$

Let $A = [a_{ij}]$ and consider the following equation.

$$A(v_2)_B = A \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix}$$

Generalizing, $(Tv_j)_C = A(v_j)_B$ and the following equations hold for constants c_j.

$$\begin{aligned} (Tv)_C &= \left(\sum_{j=1}^{n} c_j T v_j \right)_C = \sum_{j=1}^{n} (c_j T v_j)_C = \sum_{j=1}^{n} c_j (Tv_j)_C \\ &= \sum_{j=1}^{n} c_j A(v_j)_B = A \left(\sum_{j=1}^{n} (c_j v_j)_B \right) = A \left(\sum_{j=1}^{n} c_j v_j \right)_B \\ &= A(v)_B \end{aligned}$$

Thus, there always exists such a matrix A.

The matrix A above has a special name. The matrix is known as the matrix representation of T with respect to B and C, denoted as A_B^C. In short, we also say A from B to C.

Continuing from the theorem, we can also notice that for a vector space V and its basis A and B, there exists a matrix P_A^B such that $(v)_B = P_A^B(v)_A$ holds for all $v \in V$. To prove this, consider the identity transformation $T : V \to V$. By theorem 16.2.14, there exists a matrix P such that $(v)_B = P_A^B(v)_A$. This matrix too has a special name.

Definition 16.2.15

A **transition matrix** is a matrix that converts the coordinate representation of a vector in a vector space V from a basis in V to another basis in V.

Now from the definitions, we could notice that linear transformations are highly related to the basis of a vector space. If we know how the basis is mapped, we can understand the linear transformation. Consider the following theorem.

Theorem 16.2.16

For a vector space and its bases A and B, the coordinate representation of the j vector of A with respect to B is the j^{th} column of the transition matrix P_A^B.

Proof.

Let $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$. For some scalars p_{ij}, a_i can be written as $a_j = \sum_{i=1}^{n} p_{ij} b_i$. Let a vector v in the vector space be written as $v = \sum_{k=1}^{n} a'_k + a_k = \sum_{k=1}^{n} b'_k + b_k$. Continuing, the following equation holds.

$$v = \sum_{j=1}^{n} a'_j a_j = \sum_{j=1}^{n} a'_j \left(\sum_{i=1}^{n} p_{ij} b_i \right) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} p_{ij} a'_j \right) b_i$$

In other words, $b'_i = \sum_{j=1}^{n} p_{ij} a'_j$ and $(v)_B = [p_{ij}](v)_A = P_A^B(v)_A$.

Continuing with the theorem above, we can derive the following.

Theorem 16.2.17

A transition matrix P_A^B is always invertible and $(P_A^B)^{-1} = P_B^A$.

Proof.

Consider the following equations for a vector v.

$$(v)_B = P_A^B(v)_A$$
$$(v)_A = P_B^A(v)_B$$

Therefore, $(v)_B = P_A^B(v)_A = P_A^B P_B^A(v)_B$ for all $(v)_B$. Thus, $P_A^B P_B^A = I$ and the theorem holds.

Now how about matrix representations like A_S^S? We have an interesting theorem that links such matrices with transition matrices.

Theorem 16.2.18

Let A_B^B and A_C^C be the matrix representations of linear transformations for bases B and C. If P_B^C represents the transition matrix, then $A_B^B = P_C^B A_C^C P_B^C$.

Proof.

For a vector v in the same vector space, notice that $(Tv)_B = A_B^B(v)_B$ and $(v)_C = P_B^C(v)_B$ hold by definition. Consider the following equation.

$$\begin{aligned}(Tv)_B &= P_C^B(Tv)_C = P_C^B A_C^C(v)_C = P_C^B A_C^C P_B^C(v)_B \\ &= A_B^B(v)_B\end{aligned}$$

Thus, $P_C^B A_C^C P_B^C = A_B^B$.

Let's use A for A_B^B, $\tilde{A}$ for A_C^C, and P for P_B^C where $P^{-1} = P_C^B$. Notice that we can write $A = P^{-1}\tilde{A}P$ and $PA = \tilde{A}P$. Here, we say that A and $\tilde{A}$ are **similar**.

Definition 16.2.19

Two matrices A and B are **similar** if the linear transformation that they represent with different bases is identical. In other words, for transition matrix $P = P_A^B$, $PA = BP$.

Continuing with the definition, we can establish the following theorem.

Theorem 16.2.20

For a vector space V with basis B, let A be the matrix representation of the linear transformation $T : V \to V$ from B to B. If A is similar to another matrix representation $\tilde{A}$, then there exists a basis C such that $\tilde{A}$ represents the same linear transformation from C to C.

Proof.

First and foremost, let a transition matrix from a basis C to B be P and let v be a vector in the vector space. Consider the following equations.

$$\begin{aligned}(Tv)_B &= A(v)_B = AP(v)_C \\ &= P(Tv)_C\end{aligned}$$

Therefore, $AP(v)_C = P(Tv)_C$ and $(Tv)_C = A'(v)_C = P^{-1}AP(v)_C$. In other words, for each basis C, there exists a transition matrix similar to A. Moreover, because the whole process is invertible, the theorem holds.

Now that we discussed that matrices can be used to represent linear transformations, let's discuss about topics that are never left out in linear algebra.

16.2.4 Kernels and Images

As you could have noticed, sometimes it is easier to discuss a linear transformation with intuitions and sometimes matrices are much more

straightforward. Recall that linear transformations are functions for vector spaces. Sometimes it is easier to look at the functions directly and sometimes it is more convenient to look at the matrix representation of such functions. That is what we are going to do with kernels and images. As always, let's get started with definitions.

Definition 16.2.21

The **kernel** of a linear transformation $T : V \to W$, also known as the **nullspace** of T, is the set of vectors in V that are mapped to zero vectors when transformed with T.

$$\ker(T) = \{v \in V \mid Tv = \mathbf{0}\}$$

Continuing with the definition, we can establish two fundamental properties of the kernel of a transformation.

Theorem 16.2.22

Consider the following statements for all linear transformations $T : V \to W$.

1. $\mathbf{0} \in \ker(T)$
2. $\ker(T)$ is a subspace of V.

Let's start by proving the first statement. The proof is rather straightforward with the properties of linear transformation.

Proof.

Consider the following equation.

$$T(\mathbf{0}) = T(0\mathbf{0}) = 0T(\mathbf{0}) = \mathbf{0}$$

Because $\mathbf{0}$ is mapped to $\mathbf{0}$ after any linear transformation T, $\mathbf{0} \in \ker(T)$ for all linear transformations T.

Continuing, below is the proof for the second statement.

Proof.

By definition, $\ker(T) \in V$. Therefore, it suffices to show that the kernel of T is closed under vector addition and scalar multiplication. Because $\ker(T)$ is not empty by the first statement, consider the following equations for $u, v \in \ker(T)$ and some scalars α and β.

$$T(\alpha u + \beta v) = \alpha Tu + \beta Tv = \alpha \mathbf{0} + \beta \mathbf{0} = \mathbf{0}$$

In other words $\alpha u + \beta v \in \ker(T)$ and $\ker(T)$ is a subspace of V.

Recall that a linear transformation can be represented as some matrix A. Therefore, kernel of A can be interpreted as the set of solutions to $Ax = \mathbf{0}$. Notice that some solutions retain finite and infinite number of vectors. Therefore, we can establish the following definition.

Definition 16.2.23

The **nullity** of T, denoted as $\text{null}(T)$, is the dimension of $\ker(T)$ for some linear transformation T.

Continuing, we can define the image of a transformation, which should be unsurprising.

Definition 16.2.24

The subset of the codomain W for a transformation $T : V \to W$ that contains all output vectors is known as the **image** of T. Below defines $\Im(T)$ with $v \in V$.

$$\Im(T) = \{w \in W \mid w = Tv\}$$

Although the kernel and the image of a transformation are very different, we can establish very similar statements for the image just as how we derived them for the kernel.

Theorem 16.2.25

Consider the following statements for all linear transformations $T : V \to W$.

1. $\mathbf{0} \in \Im(T)$
2. $\Im(T)$ is a subspace of W.

Let's start by proving the first statement.

Proof.

By theorem 16.2.22, there exists a vector in V such that it is mapped to $\mathbf{0}$ in W. Thus, the statement holds.

Continuing, below is the proof for the second statement.

Proof.

First and foremost, $\Im(T) \in W$ by definition. Therefore, it suffices to show that the image is closed under vector addition and scalar multiplication. Consider the following equation for $v_1, v_2 \in V$, $Tv_1 = w_1$, $Tv_2 = w_2$, and some scalars α and β.

$$\alpha w_1 + \beta w_2 = \alpha T v_1 + \beta T v_2 = T(\alpha v_1 + \beta v_2)$$

Because $T(\alpha v_1 + \beta v_2) \in \Im(T)$, $\alpha w_1 + \beta w_2 \in \Im(T)$ and the statement holds.

Now recall the kernel of a matrix A is the set of all solutions to the equation $Ax = \mathbf{0}$. Similarly, the image is the set of all possible outcomes of the product Ax for some $x \in V$. Consider the equations below.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} a_{1k} x_k \\ \sum_{k=1}^{n} a_{2k} x_k \\ \vdots \\ \sum_{k=1}^{n} a_{mk} x_k \end{bmatrix} = \sum_{k=1}^{n} x_k \begin{bmatrix} a_{1k} \\ a_{2k} \\ \vdots \\ a_{mk} \end{bmatrix}$$

In other words, the image of A is the column space of A. Therefore, we can establish the following definition.

Definition 16.2.26

The **rank** of a transformation T, denoted as $\mathrm{r}(T)$, is $\dim(\Im(T))$.

Continuing from the definition and ideas of column space and the image, we can derive the following theorem.

Theorem 16.2.27

Let B and C be n-dimensional and m-dimensional bases for V and W in a linear transformation $T : V \to W$ respectively. If the matrix representation of T is $A = A_B^C$, then $\mathrm{r}(T) = \mathrm{r}(A)$.

Proof.

First and foremost, let a basis for $\Im(T)$ be $\{w_1, \ldots, w_k\}$. Notice that if $\{(w_1)_C, \ldots, (w_k)_C\}$ is a basis for $\mathrm{col}(A)$, then $\dim(\Im(T)) = \dim(\mathrm{col}(A))$ and $\mathrm{r}(T) = \mathrm{r}(A)$. Thus, it suffices to show that $\{(w_1)_C, \ldots, (w_k)_C\}$ is a basis for $\mathrm{col}(A)$.

Let c_k be some scalars that $x \in \mathrm{col}(A)$ is represented as the linear combination of the columns A_k of A where $x = \sum_{k=1}^{n} c_k A_k$. Note that c_k can be written as an n-tuple. Thus, let $v \in V$ be a vector such that the following equation holds.

$$(v)_B = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}$$

Continuing, $x = A(v)_B = (Tv)_C$. Because $Tv \in \Im(T)$, it is evident that $x \in \mathrm{span}\{(w_1)_C, \ldots, (w_k)_C\}$. Moreover, because x is an arbitrary vector from $\mathrm{col}(A)$, it is evident that $\mathrm{col}(A) \subset \mathrm{span}\{(w_1)_C, \ldots, (w_k)_C\}$ and $\{(w_1)_C, \ldots, (w_k)_C\}$ spans $\mathrm{col}(A)$.

By definition, $\{w_1, \ldots, w_k\}$ is linearly independent. For some scalars d_k, for $\sum_{k=1}^{n} d_k (w_k)_C$ to be a zero vector, $(\sum_{k=1}^{n} d_k w_k)_C = \mathbf{0}$ and

$\sum_{k=1}^{n} d_k w_k = \mathbf{0}$, which only happens when $d_k = 0$ for all integer $k \in [0, n]$. Thus, $\{(w_1)_C, \ldots, (w_k)_C\}$ is linearly independent and is a basis for $\text{col}(A)$.

As you can see above by the theorem, although the kernel and the image of a linear transformation appear to be totally unrelated, they are somehow related. Below is another theorem that connects the bases of the kernel and the image of a linear transformation.

Theorem 16.2.28

Let $\{v_1, \ldots, v_k\}$ be the basis of the kernel of a linear transformation $T : V \to W$. If $\{v_1, \ldots, v_k, \ldots v_n\}$ is a basis for V, then a basis of $\Im(T)$ is $\{Tv_{k+1}, \ldots, Tv_n\}$.

Proof.

To prove that $\{Tv_{k+1}, \ldots, Tv_n\}$ is a basis of $\Im(T)$, it suffices to show that the set of vectors is linearly independent and spans $\Im(T)$. First, to show that the set is linearly independent, consider the following equations for some scalars c_i.

$$\sum_{i=k+1}^{n} c_i T v_i = T\left(\sum_{i=k+1}^{n} c_i v_i\right) = \mathbf{0}$$

In other words, $\sum_{i=k+1}^{n} c_i v_i \in \ker(T)$ and it can be written as the linear combination of the basis, or $\sum_{i=k+1}^{n} c_i v_i = \sum_{i=1}^{k} c_i v_i$. By definition, because $\{v_1, \ldots, v_k, \ldots v_n\}$ is linearly independent, c_i must be all zero for integer $i \in [1, n]$ for $\sum_{i=k+1}^{n} c_i v_i - \sum_{i=1}^{k} c_i v_i = \mathbf{0}$ to satisfy. Thus, $\{Tv_{k+1}, \ldots, Tv_n\}$ is linearly independent.

Continuing, the fact that the set $\{Tv_{k+1}, \ldots, Tv_n\}$ spans $\Im(T)$ can be proven. Let $w = Tv$ for any $w \in \Im(T)$. Consider the following

equations below.

$$w = Tv = T\left(\sum_{i=1}^{n} c_i v_i\right) = \sum_{i=1}^{n} c_i T v_i$$
$$= \sum_{i=1}^{k} c_i T v_i + \sum_{i=k+1}^{n} c_i T v_i = \sum_{i=k+1}^{n} c_i T v_i \in \text{span}\{Tv_{k+1}, \ldots, Tv_n\}$$

In other words, $\{Tv_{k+1}, \ldots, Tv_n\}$ spans $\Im(T)$. Thus, the theorem holds.

Continuing from this theorem, we can establish another interesting corollary as a theorem.

Theorem 16.2.29

For a finite dimensional vector space V, the following equation is true for a linear transformation $T : V \to W$.

$$\text{r}(T) + \text{null}(T) = \dim(V)$$

Proof.

Let $k = \dim(\ker(T))$ and $n = \dim(V)$. By Theorem 16.2.28, there exists a basis for $\Im(T)$ with $n - k$ vectors, or $\text{r}(T) = \dim(\Im(T)) = n - k$. Thus, $\text{r}(T) + \text{null}(T) = n - k + k = n = \dim(V)$.

Moving on from the relationship with the kernel and the image of a linear transformation, let's discuss about injection, surjection, and bijection. Consider the following theorem.

Theorem 16.2.30

A linear transformation $T : V \to W$ is injective if and only if $\ker(T) = \{\mathbf{0}\}$.

Proof.

By definition, T is injective if $v_1, v_2 \in V$ such that $v_1 \neq v_2$ implies $Tv_1 \neq Tv_2$. Therefore, if $v \in \ker(T)$, then $Tv = \mathbf{0} = T\mathbf{0}$ and $\ker(T)$

must contain no nonzero vectors for T to be injective. Thus, $\ker(T)$ must only contain the zero vector.

Continuing, below is another theorem on injection and surjection.

Theorem 16.2.31

For a linear transformation $T : V \to W$ with $\dim(V) = \dim(W) = n$, T is injective if and only if T is surjective.

Proof.

By Theorem 16.2.30, T is injective if and only if $\ker(T) = \{\mathbf{0}\}$. Moreover by definition, $\ker(T) = \{\mathbf{0}\}$ if and only if $\text{null}(T) = 0$ and $\text{r}(T) = \dim(V) - \text{null}(T) = n$ by Theorem 16.2.29. Continuing by definition, $\text{r}(T) = n$ if and only if $\dim(\Im(T)) = n = \dim(W)$. Therefore, T is injective if and only if T is surjective.

With the interesting properties, we can assign a special name for bijective linear transformation.

Definition 16.2.32

A bijective linear transformation is called an **isomorphism**. Moreover, a vector space V is said to be **isomorphic** to another vector space W if there exists an isomorphism $T : V \to W$, denoted as $V \cong W$.

Continuing from the definition, we can derive the following properties.

Theorem 16.2.33

For vector spaces U, V, and W, isomorphism is reflexive, symmetric, and transitive.

1. $U \cong U$
2. If $U \cong V$, then $V \cong U$.
3. If $U \cong V$ and $V \cong W$, then $U \cong W$.

Let's start by proving the first statement.

Proof.

Notice that it suffices to show that the identity transformation is bijective. By definition, because each vector is mapped to itself, identity transformation is bijective and $U \cong U$.

Below is the proof for the second property.

Proof.

First and foremost, notice that if there exists $T : U \to V$ that is bijective, then there exists its inverse $T^{-1} : V \to U$ that is also bijective. Continuing, consider the equation below for some vectors $v_1, v_2 \in V$ and scalars α and β.

$$\begin{aligned}
\alpha T^{-1}v_1 + \beta T^{-1}v_2 &= T^{-1}\left(T\left(\alpha T^{-1}v_1 + \beta T^{-1}v_2\right)\right) \\
&= T^{-1}\left(\alpha T\left(T^{-1}v_1\right) + \beta T\left(T^{-1}v_2\right)\right) \\
&= T^{-1}\left(\alpha v_1 + \beta v_2\right)
\end{aligned}$$

Therefore, T^{-1} is linear and T^{-1} is an isomorphism.

Continuing, below is the proof for the last statement.

Proof.

First and foremost, let $T_1 : U \to V$ and $T_2 : V \to W$ be isomorphisms. Therefore, it suffices to show that $T_2 \circ T_1$ is an isomorphism. Notice that because T_1 and T_2 are bijective, $T_2 \circ T_1$ must also be bijective. Moreover, consider the following equation for some vectors $v_1, v_2 \in V$ and scalars α and β.

$$\begin{aligned}
(T_2 \circ T_1)(\alpha v_1 + \beta v_2) &= T_2(\alpha T_1 v_1 + \beta T_1 v_2) \\
&= \alpha T_2(T_1 v_1) + \beta T_2(T_1 v_2) \\
&= \alpha(T_2 \circ T_1)(v_1) + \beta(T_2 \circ T_1)(v_2)
\end{aligned}$$

Thus, $T_2 \circ T_1$ is linear and the statement is true.

As you can see above, the three statements in Theorem 16.2.33 each shows a relation called reflexive, symmetric, and transitive. Continuing from this terms, we can assign a special name for a relation.

Definition 16.2.34

An **equivalence relation** is a relation that is reflexive, symmetric, and transitive.

Below is another interesting theorem on isomorphism.

Theorem 16.2.35

For finite dimensional vector spaces V and W, $V \cong W$ if and only if $\dim(V) = \dim(W)$.

Proof.

First, the statement if $\dim(V) = \dim(W)$, then $V \cong W$ can be proven. Notice that the coordinate representations of infinitely many vectors in V and W are unique. Therefore, $V \cong \mathbb{R}^n$ and $W \cong \mathbb{R}^n$. By Theorem 16.2.33, $\mathbb{R}^n \cong W$ and $V \cong W$.

Continuing, if $V \cong W$, then there exists a linear transformation $T : V \to W$ that is bijective. By Theorem 16.2.30, $\ker(T) = \{\mathbf{0}\}$ and $\text{null}(T) = 0$. Moreover by definition, because T is surjective, $\text{r}(T) = \dim(\Im(T)) = \dim(W)$. Finally by Theorem 16.2.29, $\dim(W) + 0 = \dim(V)$ and the theorem holds.

From the theorem above, we can notice that if we consider a bijective linear transformation that maps to coordinate representation, then all n-dimensional vector spaces are isomorphic to $\mathbb{R}^n$. In other words, every n-dimensional vector space over a given field for some finite n can be considered as the same vector space, but just with different representations.

This is it for linear transformations, and let's discuss about eigenvalues and eigenvectors!

§16.3 Eigenvalues and Eigenvectors

As always, let's start with definitions.

Definition 16.3.1

For a linear transformation $T : V \to V$ with a nonzero vector $v \in V$ and some scalar λ,if $Tv = \lambda v$, then v is an **eigenvector** of T and λ is an **eigenvalue** of T.

From here, it is important to note that while v must be a nonzero vector, λ could equal zero. Let's take a look at an example.

Exercise 16.3.2

Let $T : \mathbb{R}^2 \to \mathbb{R}^2$ be a linear transformation defined as $T(a, b) = (a, -b)$. Find an eigenvector and eigenvalue of T.

Solution.

Consider the following equation.

$$T(0, 1) = (0, -1) = -1 \cdot (0, 1)$$

Therefore, $(0, 1)$ is an eigenvector of T and -1 is the corresponding eigenvalue of T.

Below is another example for a linear transformation $T : \mathbb{R}^2 \to \mathbb{R}^2$.

Exercise 16.3.3

Find, if any, eigenvectors and eigenvalues of $T : \mathbb{R}^2 \to \mathbb{R}^2$ defined as $T(a, b) = (b, -a)$.

Solution.

For some scalar λ, consider the following equation.

$$T(a, b) = (b, -a) = \lambda(a, b) = (\lambda a, \lambda b)$$

In other words $b = \lambda a$ and $-a = \lambda b$. Substituting, $-a = \lambda^2 a$ and $(\lambda^2 + 1)a = 0$. Because $\lambda \in \mathbb{R}$, $a = 0$ and $b = 0$. However, because eigenvectors are nonzero vectors by definition, T has not real eigenvalues and eigenvectors.

Now eigenvectors and eigenvalues are important because they make our lives easier for investigating matrix representation of a linear transformation. Consider the following theorem.

Theorem 16.3.4

For a linear transformation $T : V \to V$ for a finite dimensional vector space V, there exists a basis whose matrix representation is a diagonal matrix if and only if there exists a basis composed of only eigenvectors of T.

Proof.

First, let $B = \{v_1, \ldots, v_n\}$ be a basis for V and let the matrix representation of T be a diagonal matrix A whose diagonal entries are $\lambda_1, \ldots, \lambda_n$. Thus

$$(Tv_j)_B = \begin{bmatrix} 0 \\ \vdots \\ \lambda_j \\ \vdots \\ 0 \end{bmatrix}$$

hold by construction and $Tv_j = \lambda_j v_j$. In other words, v_j and λ_j are an eigenvector and an eigenvalue of T and there exists a basis composed of only eigenvectors of T.

Continuing, let $B = \{v_1, \ldots, v_n\}$ be a basis for V whose elements are all eigenvectors of T. Therefore, $Tv_j = \lambda v_j$ and the following

equation holds.

$$(Tv_j)_B = \begin{bmatrix} 0 \\ \vdots \\ \lambda_j \\ \vdots \\ 0 \end{bmatrix}$$

Therefore, there exists a basis whose matrix representation is a diagonal matrix.

With this theorem and connecting vector spaces with matrix again, we can redefine eigenvectors and eigenvalues with different interpretations.

Definition 16.3.5

For a square matrix $A_{n\times n}$ and a nonzero n-tuple $x \in \mathbb{R}^n$, if there exists a scalar λ such that $Ax = \lambda x$, then x is an **eigenvector** and λ is an **eigenvalue** of A.

When you think about this definition, we can see that both definitions imply the same thing as the A is the matrix representation of a linear transformation and x is a vector in the corresponding vector space. Let's take a look at a quick example.

Exercise 16.3.6

Find an eigenvector and eigenvalue of matrix A defined as the following.

$$A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Solution.

Consider the following equation.

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Thus, -1 is an eigenvalue and $[01]$ is an eigenvector.

If this example looks familiar, this is actually the same question as the first example from this section. Continuing, below are two interesting theorems on eigenvectors and eigenvalues.

Theorem 16.3.7

For a linear transformation $T : V \to V$, $v \in V$ is an eigenvector with the corresponding eigenvalue λ if and only if $(v)_B$ is an eigenvector with the corresponding eigenvalue λ for A_B^B, the matrix representation of T for any basis B.

Proof.

Notice that $Tv = \lambda v$ if and only if $A_B^B(v)_B = (Tv)_B = (\lambda v)_B = \lambda(v)_B$. Thus, the theorem holds.

This theorem will come in handy as we can either find all eigenvectors and corresponding eigenvalues of the transformation or its matrix representation. Continuing, below is another theorem.

Theorem 16.3.8

For a linear transformation $T : V \to V$ and eigenvalue λ, $S_\lambda = \{v \in V | Tv = \lambda v\}$ is a subspace of V.

Proof.

First, notice that because $S_\lambda \subset V$, it suffices to show that the set is closed under vector addition and scalar multiplication. Consider the following equations for $u, v \in S_\lambda$ and some scalars α and β.

$$\begin{aligned} T(\alpha u + \beta v) &= \alpha Tu + \beta Tv = \alpha\lambda u + \beta\lambda v \\ &= \lambda(\alpha u + \beta v) \end{aligned}$$

In other words, $\alpha u + \beta v \in S_\lambda$ and the theorem holds.

As you may have guessed, we can establish a new definition from the

theorem above.

Definition 16.3.9

For a transformation $T : V \to V$ and its eigenvalue λ, $S_\lambda = \{v \in V | Tv = \lambda v\}$ defined as the **eigenspace** of T for the eigenvalue λ.

Just as we redefined eigenvectors and eigenvalues with matrices, we could do the same for eigenspace. Now that we discussed what eigenvectors and eigenvalues are, let's discuss how to find one.

Theorem 16.3.10

For a square matrix $A_{n\times n}$, λ is its eigenvalue if and only if $\det(A - \lambda I_n) = 0$.

Proof.

First notice that by definition, λ is an eigenvalue of A if and only if $Ax = \lambda x$ holds for some nonzero vector x. Continuing, $Ax = \lambda x$ holds if and only if $Ax = \lambda I_n x$ and $(A - \lambda I_n)x = 0$. In other words, the columns of the matrix $A - \lambda I_n$ are linearly dependent and invertible. Therefore by Theorem 14.5.13, the theorem holds.

Being one of the most important theorems in linear algebra, we can establish another definition from the theorem.

Definition 16.3.11

The **characteristic equation** of a square matrix A is defined as $\det(A - \lambda I_n) = 0$ for scalar variable λ.

An important thing to note from this definition is that $\det(A - \lambda I_n)$ is a polynomial in λ with degree n. Moreover, the real roots of the polynomial can be interpreted as eigenvalues of A. Let's take a look at an example of how the characteristic equation can be used to find all eigenvalues of a matrix.

Exercise 16.3.12

Find all eigenvalues and the corresponding eigenspaces of matrix $A = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$.

Solution.

To find all eigenvalues of A, it suffices to find all solutions for the characteristic equation of A by Theorem 16.3.10.

$$A - \lambda I_2 = \begin{bmatrix} 1-\lambda & 0 \\ 0 & -1-\lambda \end{bmatrix}$$

Therefore, $\det(A - \lambda I_2) = (1-\lambda)(-1-\lambda)$ and $\lambda = 1$ or $\lambda = -1$ are the solutions to the characteristic equation.

Continuing, the corresponding eigenspaces could be found. For $\lambda = 1$, the eigenspace is defined as $S_1 = \{x | Ax = x\}$. Solving for the equation $(A - I)x = 0$, the following equations hold.

$$\begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$
$$-2x_2 = 0$$

Therefore, the eigenspace for the eigenvalue 1 can be represented as the following

$$S_1 = \text{span} \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$$

Similarly, $S_{-1} = \{x | Ax = -x\}$ and $(A + I)x = 0$ could be solved.

$$\begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$
$$2x_1 = 0$$

Therefore,

$$S_{-1} = \text{span}\left\{\begin{bmatrix}0\\1\end{bmatrix}\right\}$$

and the corresponding eigenspaces for all eigenvalues of A are found.

We got a nice matrix here, but it's worth noting that λ need not be real. As demonstrated from the simple example above, solving the characteristic equation will lead to all eigenvalues for a matrix. I will leave more complex and interesting examples in the practice problems below. In the next part of the section, we will discuss the properties of eigenvalues and eigenvectors.

16.3.1 Properties of Eigenvalues

Eigenvalues and eigenvectors have interesting properties and relationship with vector spaces. Consider the following theorem.

Theorem 16.3.13

For two square matrices A and B such that $A \cong B$, the characteristic equation and eigenvalues for each matrix are equal.

Proof.

By definition of similar matrices, there exists some invertible matrix P such that $B = P^{-1}AP$. Therefore, the following equations hold.

$$P^{-1}(A - \lambda I)P = P^{-1}AP - P^{-1}\lambda IP = B - \lambda I$$

Therefore, $\det(B - \lambda I) = \det(P^{-1}(A - \lambda I)P) = \det(A - \lambda I)$ and the characteristic equations are the same. In other words, the eigenvalues for A and B are equal and the theorem holds.

Below is another theorem on the eigenvalues of triangular matrices.

Theorem 16.3.14

The eigenvalues of triangular (upper or lower) matrices are the diagonal elements.

Proof.

By definition, notice that for a triangular matrix A, $A - \lambda I$ is also triangular. Therefore by Theorem 14.5.4,

$$\det(A - \lambda I) = \prod_{k=1}^{n} (\lambda_k - \lambda)$$

and $\det(A - \lambda I_n) = 0$ if and only if $\lambda = \lambda_k$. Because λ_k are the diagonal elements of A, the theorem holds.

Continuing, below is another theorem on the relationship between eigenvalues and invertibility.

Theorem 16.3.15

A square matrix A contains a zero eigenvalue if and only if it is not invertible.

Proof.

A square matrix A contains a zero eigenvalue if and only if $\det(A - 0I) = 0$ and $\det(A) = 0$. Because A is invertible if and only if $\det(A) \neq 0$ by Theorem 14.5.13, the theorem holds.

If you think this theorem is nice, below is another theorem on invertibility and eigenvalues.

Theorem 16.3.16

For an invertible matrix A with eigenvector x and its corresponding eigenvalue λ, x is also an eigenvalue of A^{-1} with corresponding eigenvalue $\frac{1}{\lambda}$.

Proof.

By definition, $Ax = \lambda x$ and $x = A^{-1}\lambda x$. Continuing, $A^{-1}x = \frac{1}{\lambda} \cdot x$. Therefore, x is also an eigenvector of A^{-1} with the corresponding eigenvalue $\frac{1}{\lambda}$.

Continuing, below is a theorem on the relationship between eigenvalues and linear independence.

Theorem 16.3.17

If $x_1, \ldots, x_k$ are eigenvectors of a square matrix A and the corresponding eigenvalues $\lambda_1, \ldots, \lambda_k$ are all distinct, then $\{x_1, \ldots, x_k\}$ is linearly independent.

Proof.

First, notice that $\{x_1\}$ is linearly independent. Therefore, it suffices to show that if the theorem holds for $k - 1 > 0$, then the statement holds for k. Assume $\{x_1, \ldots, x_{k-1}\}$ is a linearly independent set of eigenvectors with distinct corresponding eigenvalues $\lambda_1, \ldots, \lambda_{k-1}$. Consider the following equations for some scalars c_i.

$$\sum_{i=1}^{k} c_i x_i = 0$$
$$\sum_{i=1}^{k} c_i A x_i = 0$$
$$\sum_{i=1}^{k} c_i \lambda_i x_i = 0$$
$$\sum_{i=1}^{k-1} c_i \lambda_k x_i = 0$$

Subtracting the last two equations, $\sum_{i=1}^{k-1} c_i(\lambda_i - \lambda_k)x_i = 0$ is obtained. Because $\{x_1, \ldots, x_{k-1}\}$ is linearly independent, $c_i(\lambda_i - \lambda_k) = 0$ for all integer $i \in [1, k)$. Continuing, because $\lambda_i - \lambda_k \neq 0$ by construction, $c_i = 0$ for all integer $i \in [1, k)$ and $c_k = 0$ since $x_k \neq 0$.

Thus $\{x_1, \ldots, x_k\}$ is linearly independent and the theorem holds by induction.

Naturally, we can consider the dimension of an eigenspace.

Theorem 16.3.18

For a matrix $A_{n\times n}$ with an eigenvalue λ, the following equation holds.

$$\dim(S_\lambda) = n - \mathrm{r}(A - \lambda I)$$

Proof.

By definition, the eigenspace for λ is the set of all vectors v that satisfy $Av = \lambda v$. Because $(A - \lambda)v = 0$, S_λ can be interpreted as $\ker(A - \lambda I)$. Moreover by Theorem 16.2.29, the following equation holds.

$$\mathrm{r}(A - \lambda I) + \mathrm{null}(A - \lambda I) = n$$

Because $\dim(\ker(A - \lambda I)) = \mathrm{null}(A - \lambda I)$ holds by definition, $\dim(S_\lambda) = n - \mathrm{r}(A - \lambda I)$.

We could really go on forever with interesting theorems, but before we conclude this part of the note, I would like to discuss two more theorems.

Theorem 16.3.19

Consider the following statements for a matrix A with an eigenvector x with its corresponding eigenvalue λ.

1. For an arbitrary scalar k, x is also an eigenvector of kA with corresponding eigenvalue $k\lambda$.
2. For some $n \in \mathbb{N}$, x is also an eigenvector of A^n with corresponding eigenvalue λ^n.

Let's start by proving the first statement.

Proof.

Consider the following equation.

$$(kA)x = k(Ax) = k\lambda x = (k\lambda)x$$

Therefore, x is an eigenvector of kA and its corresponding eigenvalue is $k\lambda$.

Below is the proof for the second statement.

Proof.

Assume that if the statement holds for $n-1$, then the statement holds for $n > 1$. Consider the following equations.

$$\begin{aligned} A^n x &= A\left(A^{n-1}x\right) = A\left(\lambda^{n-1}x\right) = \lambda^{n-1}Ax \\ &= \lambda^{n-1}\lambda x = \lambda^n x \end{aligned}$$

Because $Ax = \lambda x$, the statement holds by induction.

Before we move on to the final content with the last theorem for this part, below is a definition for the theorem.

Definition 16.3.20

The **trace** of a matrix $A = [a_{ij}]$, denoted as $\text{tr}(A)$, is the sum of the diagonal elements of A.

With the definition, we can continue with the theorem.

Theorem 16.3.21

For a square matrix $A_{n\times n}$ with eigenvalues $\lambda_1, \ldots, \lambda_n$, the following equations hold.

1. $\text{tr}(A) = \sum_{k=1}^{n} \lambda_k$
2. $\det(A) = \prod_{k=1}^{n} \lambda_k$

Let's start by proving the first equation.

Proof.

By the Fundamental Theorem of Algebra, the following equation holds.

$$\det(A - \lambda I) = \prod_{k=1}^{n}(\lambda_k - \lambda) = 0$$

Therefore, all eigenvalues are the diagonal elements and by the definition of trace of A, the sum of all eigenvalues are the trace of A.

Continuing, below is the proof for the second equation.

Proof.

Substituting $\lambda = 0$ into the equation

$$\det(A - \lambda I) = \prod_{k=1}^{n}(\lambda_k - \lambda),$$

$\det(A) = \prod_{k=1}^{n} \lambda_k$ and the equation holds.

It is important to note that the second equation counts multiplicity. In other words, even if $\lambda_p = \lambda_q$, you still count both. Wow! I can't believe that we are already almost done with the notes! For the final concept of my notes on linear algebra, we will discuss diagonalization.

16.3.2 Diagonalization

Diagonalization in linear algebra is a technique that I think really makes our lives easier in calculations. To get into it, let's get started with definition and theorems.

Definition 16.3.22

A square matrix that is similar to a diagonal matrix is known to be **diagonalizable**.

Now we cannot always use a brute force method to see if two matrices are similar. Fortunately, the following theorem will help in checking whether a square matrix is diagonalizable.

Theorem 16.3.23

A square matrix $A_{n\times n}$ is diagonalizable if and only if it retains a set of n eigenvectors that is linearly independent.

Proof.

First and foremost, notice that A and a diagonal matrix D of the same order with entries λ_i are similar if and only if there exists an invertible matrix P such that $P^{-1}AP = D$ or $AP = PD$. Let x_j and e_j be the j^{th} column of P and I respectively. Then, the following equations hold.

$$\begin{aligned} Ax_j &= APe_j = PDe_j = P\lambda_j e_j \\ &= \lambda_j Pe_j = \lambda_j x_j \end{aligned}$$

Thus, x_j and λ_j are an eigenvector and the corresponding eigenvalue of A. By definition, because M is invertible, x_j are all linearly independent and A has n linearly independent eigenvectors. In other words, if A is diagonalizable, then A has n linearly independent eigenvectors.

Continuing, the converse can be proven. Let A have n linearly independent eigenvectors denoted as x_j with corresponding eigenvalues λ_j. Construct matrices P and D such that the j^{th} column of P is x_j and the diagonal element of D is λ_j. If e_j is the j^{th} column of I, then the j^{th} column of AP can be written as the following.

$$APe_j = Ax_j = \lambda_j x_j$$

Similarly, the j^{th} column of PD can be written as $PDe_j = M\lambda_j e_j = \lambda_j Me_j = \lambda_j x_j$ and $AP = PD$ holds. Moreover, because P is invertible by construction, $P^{-1}AP = D$ and A is diagonalizable.

Continuing from the theorem, we can establish the following corollary.

Corollary 16.3.24

A matrix $A_{n\times n}$ with n distinct real eigenvalues is diagonalizable.

Proof.

First and foremost, notice that by Theorem 16.3.17, the set of corresponding eigenvectors $\{x_1, \ldots, x_n\}$ of A is linearly independent. Thus by Theorem 16.3.23, the corollary holds.

This theorem also naturally leads to the question if a square matrix can be similar to more than one diagonal matrix. The answer to that question can be found with the theorem below.

Theorem 16.3.25

If a square matrix is similar to both diagonal matrices D_1 and D_2, then the diagonal entries of D_1 and D_2 are the same with possibly different orderings.

Proof.

First and foremost, notice that by Theorem 16.3.13, because A and D_1 are similar, they have the same characteristic equation. Similarly, because A and D_2 are similar, they also have the same characteristic equation and the characteristic equations of D_1 and D_2 are equal.

$$\det(D_1 - \lambda I) = \det(D_2 - \lambda I) = 0$$

By construction, notice that $D_1 - \lambda I$ and $D_2 - \lambda I$ are diagonal matrices. Thus by Corollary 14.5.5, the determinant is the product of all $(\lambda - \lambda_k$ for integer $k \in [1, n]$ where λ_k are the diagonal entries for D_1. Because multiplication is commutative, the diagonal entries for D_1 and D_2 are the same.

Naturally, we can see that if $A \cong D_1$ and $A \cong D_2$, then $D_1 \cong A$ and $D_1 \cong D_2$. In other words, if A is similar to multiple diagonal matrices, then the diagonal matrices are similar to each other.

Now that we discussed when a matrix is diagonalizable, let's discuss how to diagonalize and its applications.

Definition 16.3.26

Diagonalization is the process of writing a diagonalizable matrix A in the form of $A = PDP^{-1}$.

Diagonalization is particularly useful as it makes calculations easier. Before concluding our notes, below is the final theorem!

Theorem 16.3.27

Let $A_{n\times n}$ be a matrix with real and distinct eigenvalues $\lambda_1, \ldots, \lambda_k$. The matrix A is diagonalizable if and only if the following equation holds.

$$\sum_{i=1}^{k} \dim(S_{\lambda_i}) = n$$

Proof.

First, the statement if $\sum_{i=1}^{k} \dim(S_{\lambda_i}) = n$, then A is diagonalizable can be proven. Let B_i be a basis for eigenspace S_{λ_i}. By definition, vectors v in eigenspace S_{λ_i} satisfy $Av = \lambda_i v$. Because λ_i are all distinct, no eigenspace shares the same eigenvector and thus no basis of two different eigenspaces shares a vector. In other words, B contains n distinct eigenvectors of A. Let $v_i \in S_{\lambda_i}$ be some linear combination of vectors in B_i. Notice that if $\sum_{i=1}^{k} v_i = 0$, then the coefficients must be zero since $\{v_1, \ldots, v_k\}$ is linearly independent by Theorem 16.3.17. In other words, all coefficients for the linear combination of B_i that forms v_i must be zero for $\sum_{i=1}^{k} v_i = 0$ to hold. Thus, B is linearly independent and A is diagonalizable by Theorem 16.3.23.

Continuing, the converse can be proven. Assume that $\sum_{i=1}^{k} \dim(S_{\lambda_i}) > n$. Continuing from the previous definition, $B \subset \mathbb{R}^n$ and the inequality $\sum_{i=1}^{k} \dim(S_{\lambda_i}) > n$ leads to contradiction. Moreover, if

> $\sum_{i=1}^{k} \dim(S_{\lambda_i}) < n$ then A has no set that contains n linearly independent eigenvectors. Thus, A is not diagonalizable if $\sum_{i=1}^{k} \dim(S_{\lambda_i}) \neq n$.

This is it for eigenvalues and eigenvectors! This note was a particularly long one, but I think it also means that the relationships between matrices and vector spaces are one of the most essential understandings for linear algebra. Below are a few practice problems on the topics that we discussed.

§16.4 Practice Problems

1. Determine whether a transformation defined as $T(\langle a, b\rangle) = a + b$ is linear. How about $T(\langle a, b\rangle) = ab$?

2. For square matrices A and B of order $n \times n$, show that if λ is an eigenvalue of AB, then the eigenvalues of BA also include λ [Erd20].

3. Find bases for the kernel and the image of $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$.

4. Show that $A \cong B$ if and only if $A^\intercal \cong B^\intercal$.

5. Find the eigenvalues and corresponding eigenspace for $\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & 0 \\ 0 & 1 & 2 \end{bmatrix}$.

Note 17

Solutions

Here are the solutions to the practice problems at the end of each note.

§17.1 Solutions for Note 1

1. It suffices to show that $\alpha(x,y,z) + \beta(x',y',z') \in S$ for any scalars α and β. With standard vector addition and scalar multiplication, $\alpha(x,y,z) + \beta(x',y',z') = (0, \alpha y + \beta y', \alpha z + \beta z') \in S$. Thus, the set is a vector space.

2. Let f and g be functions in $\mathbb{R}$ such that $f'(0) = 0$ and $g'(0) = 0$. Notice that $(f+g)'(0) = f'(0) + g'(0) = 0$ and $f + g \in S$. Moreover, for any scalars α, $(\alpha f)'(0) = \alpha f'(0) = 0$. Thus $\alpha f \in S$ and S is a vector space.

3. First, for any scalars α, notice that $\alpha(x_1, \ldots, x_n) = (\alpha x_1, \ldots, \alpha x_n)$. If each coefficient is divided by $\alpha \neq 0$, the scalar multiplication holds. The case where $\alpha = 0$ naturally holds. Continuing, notice that $(x_1, \ldots, x_n) + (x_1', \ldots, x_n') = (x_1 + x_1', \ldots, x_n + x_n') \in S$ due to the property of addition. Thus, S is a vector space.

4. Let $V = W_1 \cap \cdots \cap W_n$ where W_i for integer $i \in [1, n]$ is a subspace of V. For $u, v \in V$, notice that since $u, v \in W_i$ for all i and $\alpha u + \beta v \in W_i$ for any scalars α and β by construction, $\alpha u + \beta v \in V$ and the intersection of subspaces of a vector space is also a subspace of the vector space.

5. First and foremost, it is evident that $e^{a_1 x}$ is linearly independent since $e^{a_1 x} > 0$ for all a_1 and x. Therefore, it suffices to show that if the set is linearly independent for $n - 1$, then it is linearly independent for n. Consider the following equations for some c_k.

$$\sum_{k=1}^{n} c_k e^{a_k x} = 0$$
$$\sum_{k=1}^{n} c_k a_k e^{a_k x} = 0$$
$$\sum_{k=1}^{n} c_k a_n e^{a_k x} = 0$$

Subtracting the third equation, which is obtained by multiplying a_n in both sides, to the second equation obtained by differentiating both sides,

$$\sum_{k=1}^{n} c_k (a_k - a_n) e^{a_k x} = \sum_{k=1}^{n-1} c_k (a_k - a_n) e^{a_k x} = 0$$

is obtained. Because $a_k \neq a_n$ for integer $k \in [1, n-1]$ and $\{e^{a_1 x}, \ldots, e^{a_{n-1} x}\}$ is linearly independent $c_1 = \cdots = c_{n-1} = 0$. In other words, $c_n e^{a_n x} = 0$ and $c_n = 0$. Therefore, $\{e^{a_1 x}, \ldots, e^{a_n x}\}$ is linearly independent by induction.

6. By definition, note that $C = A + B$. Therefore, it suffices to show that $A \cap B = \{0\}$. Notice that A and B can be written as $A = \text{span}\{(1, 1, 1)\}$ and $B = \text{span}\{(20, 15, 12)\}$. Because $\lambda(1, 1, 1) = (20, 15, 12)$ if and only if $\lambda = 0$, $A \cap B = \{0\}$ and $C = A \oplus B$.

7. Consider the following equations for some scalars k_1, k_2, k_3.

$$k_1(a+b) + k_2(b+c) + k_3(c+a) = 0$$
$$a(k_3+k_1) + b(k_1+k_2) + c(k_2+k_3) = 0$$

The linear combinations of $\{a+b, b+c, c+a\}$ equal zero if and only if the linear combinations of a, b, and c equal zero. Moreover, from the equation above, $k_1 = -k_2 = k_3 = -k_1$. In other words, $k_1 = k_2 = k_3 = 0$ and $\{a+b, b+c, c+a\}$ is linearly independent.

8. Let f and g be polynomials such that $f(0) = g(0) = 0$. Therefore, $(f+g)(0) = 0$ and $f+g$ is in the set. Moreover, $(\alpha f)(0) = 0$ for any scalars α. Therefore, the set of all polynomials in $\mathbb{P}^3$ that includes the origin is a subspace of $\mathbb{P}^3$.

§17.2 Solutions for Note 2

1. (a) Consider the following computation.

$$A - 2B = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} - 2\begin{bmatrix} -1 & 2 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} + \begin{bmatrix} 2 & -4 \\ 0 & -6 \end{bmatrix}$$
$$= \begin{bmatrix} 3 & -2 \\ 2 & -6 \end{bmatrix}$$

(b) Consider the following equation.

$$AB = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} -1 & 2 \\ 0 & 3 \end{bmatrix} = \begin{bmatrix} -1 & 8 \\ -2 & 7 \end{bmatrix}$$

(c) Because AB is found above, BA could be found.

$$BA = \begin{bmatrix} -1 & 2 \\ 0 & 3 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ 6 & 0 \end{bmatrix}$$

Therefore, the following equation is true.

$$(AB - BA)^\intercal = \left(\begin{bmatrix} -1 & 8 \\ -2 & 7 \end{bmatrix} - \begin{bmatrix} 3 & -2 \\ 6 & 0 \end{bmatrix} \right)^\intercal = \left(\begin{bmatrix} -4 & 10 \\ -8 & 7 \end{bmatrix} \right)^\intercal = \begin{bmatrix} -4 & -8 \\ 10 & 7 \end{bmatrix}$$

2. By the definition of matrix multiplication, $a_2 B = [4, 5, 6]$.

3. Notice that A is a 2×2 square matrix. Therefore, define A as the following.

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Therefore, the following equation is obtained.

$$2A - 3A^\intercal = \begin{bmatrix} 2a_{11} & 2a_{12} \\ 2a_{21} & 2a_{22} \end{bmatrix} - \begin{bmatrix} 3a_{11} & 3a_{21} \\ 3a_{12} & 3a_{22} \end{bmatrix} = \begin{bmatrix} 2a_{11} - 3a_{11} & 2a_{12} - 3a_{21} \\ 2a_{21} - 3a_{12} & 2a_{22} - 3a_{22} \end{bmatrix} = \begin{bmatrix} -1 & -5 \\ 0 & -4 \end{bmatrix}$$

Continuing, the elements of A could be found.

$$\begin{aligned} 2a_{11} - 3a_{11} &= -1 \\ 2a_{12} - 3a_{21} &= -5 \\ 2a_{21} - 3a_{12} &= 0 \\ 2a_{22} - 3a_{22} &= -4 \end{aligned}$$

Therefore, $a_{11} = 1$, $a_{12} = 2$, $a_{21} = 3$, and $a_{22} = 4$. Thus, A can be defined as the following.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

4. For any square matrix S, define A and B as the following.

$$A = \frac{S + S^\intercal}{2}, \quad B = \frac{S - S^\intercal}{2}$$

Notice that $S = A + B$. Therefore, it suffices to show that A is symmetric and B is skew-symmetric. By Theorem 14.4.4, the following equations hold.

$$A^\intercal = \frac{S + S^\intercal}{2}, \quad B^\intercal = \frac{S^\intercal - S}{2} = -B$$

Thus, all square matrices can be represented as the sum of a symmetric and skew-symmetric matrices.

5. By definition, there exist $n, m \in \mathbb{Z}$ such that $A^n = 0$ and $B^m = 0$. Because $AB = BA$, $A^{nm}B^{nm} = 0$ can be written as $(AB)^{nm} = 0$ and AB is nilpotent.

6. By Theorem 14.4.4, $(AB + (AB)^\intercal)^\intercal = (AB)^\intercal + AB = AB + (AB)^\intercal$. Therefore, $AB + (AB)^\intercal$ is symmetric.

7. An easy trap for this problem is that you might be tempted to do $\det(A^2 - B^2) = \det(A + B)\det(A - B)$. However, notice that this does not hold in general as $AB \neq BA$ in general.

Consider the following matrix P and its inverse P^{-1}.

$$P = \begin{bmatrix} I & I \\ I & -I \end{bmatrix}, \quad P^{-1} = \frac{1}{2}\begin{bmatrix} I & I \\ I & -I \end{bmatrix}$$

Continuing,

$$\begin{aligned} P^{-1}\begin{bmatrix} A & B \\ B & A \end{bmatrix} P &= \frac{1}{2}\begin{bmatrix} A+B & B+A \\ A-B & B-A \end{bmatrix}\begin{bmatrix} I & I \\ I & -I \end{bmatrix} \\ &= \begin{bmatrix} A+B & 0 \\ 0 & A-B \end{bmatrix} \end{aligned}$$

holds and $\begin{bmatrix} A & B \\ B & A \end{bmatrix} \cong \begin{bmatrix} A+B & 0 \\ 0 & A-B \end{bmatrix}$. Therefore, the determinant of the original matrix is $\det(A+B)\det(A-B)$.

8. By definition, A is skew-symmetric if $A^\intercal = -A$. Notice that $(A^3)^\intercal = (A^\intercal)^3$ by Theorem 14.4.4. Therefore, $(A^3)^\intercal = (A^\intercal)^3 = -A^3$. Thus, if A is skew-symmetric, then A^3 is also skew-symmetric.

§17.3 Solutions for Note 3

1. If A is invertible, then $A^{-1}Ax = A^{-1}b$ and $x = A^{-1}b$. Therefore, the system $Ax = b$ is always consistent.

2. Consider the following representation of the system.

$$\begin{aligned} x + 2y &= 1 \\ 2x + 4y &= k \end{aligned}$$

Notice that if $k = 2$, then there exist infinitely many solutions while $k \neq 2$ gives no solutions. Thus the system never has a unique solution.

3. First and foremost, consider the following augmented matrix.

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 10 \\ 3 & 4 & -1 & 2 \\ -1 & -3 & 1 & 4 \end{array}\right]$$

Using elementary row operations, the augmented matrix can be transformed as following.

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 10 \\ 3 & 4 & -1 & 2 \\ -1 & -3 & 1 & 4 \end{array}\right] \to \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10 \\ 0 & 1 & -4 & -28 \\ -1 & -3 & 1 & 4 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10 \\ 0 & 1 & -4 & -28 \\ 0 & -2 & 2 & 14 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|c} 1 & 1 & 1 & 10 \\ 0 & 1 & -4 & -28 \\ 0 & 0 & -6 & -42 \end{array}\right]$$

Therefore, $-6z = -42$ and $z = 7$. Moreover, $y - 4z = -28$ and $y = 0$. Finally, $x + y + z = 10$ and $x = 3$. Therefore, the solution to the system is $(3, 0, 7)$.

4. First and foremost, let $A = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix}$. Using elementary row operations, A can be transformed as following.

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ -1 & -3 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & -2 & 2 \end{bmatrix}$$
$$\rightarrow \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -6 \end{bmatrix}$$

In other words, A can be represented as the matrix multiplication below.

$$\begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -6 \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -6 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -1 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -6 \end{bmatrix}$$

Consider the following system for equation for $Ly = b$.

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ -1 & -2 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 10 \\ 2 \\ 4 \end{bmatrix}$$

It is evident that $x' = 10$, $y' = 2 - 3x' = -28$, and $z' = 4 + x' + 2y' = -42$. Continuing, the system for $Ux = y$ can be solved.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -4 \\ 0 & 0 & -6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ -28 \\ -42 \end{bmatrix}$$

Therefore, $z = 7$, $y = 0$, and $x = 3$ holds and the solution for the system is $(3, 0, 7)$ as shown in the previous problem.

5. To find the inverse of $A = \begin{bmatrix} 1 & 1 & 1 \\ 3 & 4 & -1 \\ -1 & -3 & 1 \end{bmatrix}$, consider the following augmentation and transformations.

$$\left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 3 & 4 & -1 & 0 & 1 & 0 \\ -1 & -3 & 1 & 0 & 0 & 1 \end{array}\right] \rightarrow \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & -4 & -3 & 1 & 0 \\ -1 & -3 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & -4 & -3 & 1 & 0 \\ 0 & -2 & 2 & 1 & 0 & 1 \end{array}\right]$$

$$\rightarrow \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & -4 & -3 & 1 & 0 \\ 0 & 0 & -6 & -5 & 2 & 1 \end{array}\right]$$

$$\to \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & -4 & -3 & 1 & 0 \\ 0 & 0 & 1 & \frac{5}{6} & -\frac{1}{3} & -\frac{1}{6} \end{array}\right]$$

$$\to \left[\begin{array}{ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & \frac{1}{3} & -\frac{1}{3} & -\frac{2}{3} \\ 0 & 0 & 1 & \frac{5}{6} & -\frac{1}{3} & -\frac{1}{6} \end{array}\right]$$

$$\to \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & -\frac{1}{6} & \frac{2}{3} & \frac{5}{6} \\ 0 & 1 & 0 & \frac{1}{3} & -\frac{1}{3} & -\frac{2}{3} \\ 0 & 0 & 1 & \frac{5}{6} & -\frac{1}{3} & -\frac{1}{6} \end{array}\right]$$

Therefore, $A^{-1} = \begin{bmatrix} -\frac{1}{6} & \frac{2}{3} & \frac{5}{6} \\ \frac{1}{3} & -\frac{1}{3} & -\frac{2}{3} \\ \frac{5}{6} & -\frac{1}{3} & -\frac{1}{6} \end{bmatrix}$ and the system could be solved by multiplying A^{-1} and b.

$$\begin{bmatrix} -\frac{1}{6} & \frac{2}{3} & \frac{5}{6} \\ \frac{1}{3} & -\frac{1}{3} & -\frac{2}{3} \\ \frac{5}{6} & -\frac{1}{3} & -\frac{1}{6} \end{bmatrix} \begin{bmatrix} 10 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 7 \end{bmatrix}$$

Thus, the solution is $(3, 0, 7)$, which is consistent with the previous problems.

§17.4 Solutions for Note 4

1. For transformation $T(\langle a, b\rangle) = a + b$, consider the following equations for some scalars α and β.

$$\begin{aligned} T(\alpha\langle a_1, b_1\rangle + \beta\langle a_2, b_2\rangle) &= T(\langle \alpha a_1, \alpha b_1\rangle + \langle \beta a_2, \beta b_2\rangle) \\ &= \alpha a_1 + \alpha b_1 + \beta a_2 + \beta b_2 \\ &= \alpha(a_1 + b_1) + \beta(a_2 + b_2) \\ &= \alpha T(\langle a_1, b_1\rangle) + \beta T(\langle a_2, b_2\rangle) \end{aligned}$$

Therefore by the definition of linear transformation, T is linear.

For $T(\langle a, b\rangle) = ab$, consider the following for arbitrary scalars α and β.

$$\begin{aligned} T(\alpha\langle a_1, b_1\rangle + \beta\langle a_2, b_2\rangle) &= T(\langle \alpha a_1, \alpha b_1\rangle + \langle \beta a_2, \beta b_2\rangle) \\ &= T(\langle \alpha a_1 + \beta b_2, \alpha b_1 + \beta b_2\rangle) \\ &= (\alpha a_1 + \beta b_2)(\alpha b_1 + \beta b_2) \end{aligned}$$

Moreover,

$$\alpha T(\langle a_1, b_1\rangle) + \beta T(\langle a_2, b_2\rangle) = \alpha a_1 b_1 + \beta a_2 b_2$$

holds. Because $(\alpha a_1 + \beta b_2)(\alpha b_1 + \beta b_2) \neq \alpha a_1 b_1 + \beta a_2 b_2$ in general, the transformation is not linear.

2. By definition, if λ is an eigenvalue of AB, then there exists some nonzero vector v such that $ABv = \lambda v$. First, let $\lambda \neq 0$. Notice that $u = Bv \neq \mathbf{0}$. Therefore, $BAu = BA(Bv) = B(ABv) = B(\lambda v) = \lambda(Bv) = \lambda u$ and λ is also an eigenvalue of BA. Continuing if $\lambda = 0$, then $\det(AB) = \det(A)\det(B) = 0 = \det(B)\det(A) = \det(BA)$. In other words, if AB is singular then BA is also singular and contains the eigenvalue λ. Therefore, λ is an eigenvalue of AB if and only if it is an eigenvalue of BA.

3. First, to find bases for the kernel of the matrix, the following equation must be solved.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving the system, $y = 0$ and $x = -z$. Therefore the basis for the kernel is $\left\{ \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \right\}$. Continuing, the basis for the image can be

found. Consider the following equation.

$$\begin{bmatrix}1 & 1 & 1\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\ y\\ z\end{bmatrix} = \begin{bmatrix}x+y+z\\ y\end{bmatrix} = x\begin{bmatrix}1\\ 0\end{bmatrix} + y\begin{bmatrix}1\\ 1\end{bmatrix} + z\begin{bmatrix}1\\ 0\end{bmatrix}$$

Therefore,

$$\left\{\begin{bmatrix}1\\ 0\end{bmatrix}, \begin{bmatrix}1\\ 1\end{bmatrix}\right\}$$

is a basis of the image of the given matrix.

4. By definition, $A \cong B$ if and only if there exists an invertible matrix P such that $A = P^{-1}BP$. Continuing, $A = P^{-1}BP$ if and only if $A^\intercal = (P^{-1}BP)^\intercal = P^\intercal B^\intercal (P^{-1})^\intercal = P^\intercal B^\intercal (P^\intercal)^{-1}$. In other words, $A \cong B$ if and only if $A^\intercal \cong B^\intercal$.

5. To find the eigenvalues for the given matrix say A, it suffices to solve $\det(A - \lambda I) = 0$ where λ is an eigenvalue.

$$\det\left(\begin{bmatrix}1-\lambda & 2 & 3\\ 0 & 1-\lambda & 0\\ 0 & 1 & 2-\lambda\end{bmatrix}\right) = 0$$

Continuing with the expansion along the first column,

$$\begin{aligned}&\det\left(\begin{bmatrix}1-\lambda & 2 & 3\\ 0 & 1-\lambda & 0\\ 0 & 1 & 2-\lambda\end{bmatrix}\right)\\ &= (1-\lambda)\det\left(\begin{bmatrix}1-\lambda & 0\\ 1 & 2-\lambda\end{bmatrix}\right)\\ &= (1-\lambda)(1-\lambda)(2-\lambda)\end{aligned}$$

Therefore, the eigenvalues are 1 and 2. For eigenvalue 1, it suffices to solve the equation $Av = v$ or $(A - I)v = 0$. Consider the

following system.

$$\begin{bmatrix} 0 & 2 & 3 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Therefore, $2y + 3z = 0$ and $y + z = 0$. In other words, $y = z = 0$. Thus, $S_1 = \text{span}\left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}$ holds for $\lambda = 1$. Continuing with $\lambda = 2$, it suffices to solve $Av = 2v$ or $(A - 2I)v = 0$.

$$\begin{bmatrix} -1 & 2 & 3 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Solving the system, $y = 0$ and $x = 3z$. Thus, $S_2 = \text{span}\left\{ \begin{bmatrix} 3 \\ 0 \\ 1 \end{bmatrix} \right\}$ holds for $\lambda = 2$.

§17.5 Final Thoughts

I wanted to include my final thoughts on these notes and notice on some topics that were not covered here, and I guess I can use the help of appendix.

First, as mentioned in the prefix, these are the notes that I took while learning the essence of linear algebra in 8 weeks. That being said, while most of the fundamental concepts were covered, other topics like dual space, cross product, euclidean inner product, and more advanced concepts including Markov chains and Jordan canonical forms are not included here. After the initial release, I will continue to update these notes for the second edition as I will likely take linear algebra course again in college as a student who will major in pure math. In the meantime, I

highly recommend watching 3Blue1Brown series on linear algebra as it really helped me visually understand different concepts and formulas.

As I also study different concepts in machine learning, I realized how important linear algebra is outside "mathematics." Literally, vectors are used everywhere including, but not limited to encoding/decoding and backpropagation. One of the reasons why I find math so fascinating is due to duality and applications that just seems so perfect. Matrices can be interpreted with heavy computation while it can also be interpreted geometrically with change in area. The two seemingly unrelated topics actually imply the same exact concept. Similarly, what else can we use instead of matrix multiplication and addition for forward pass in machine learning? The way it seems so intuitive and "perfect" is I think one of the most beautiful side of mathematics. Really, if you completed these notes, I highly recommend learning math behind machine learning if you like finding different applications of mathematics. More information can be found in the next part of LibreNotebook.

Part III

Multivariable Calculus

Contents

Introducing the Notes **339**

18 Vectors and Geometry **341**
18.1 Basic Terms and Intuitions 341
18.1.1 Geometric Intuition in 3-Dimensions 344
18.2 Vectors 345
18.2.1 Dot Products and Projections 350
18.2.2 The Cross Product and Geometry 354
18.3 Geometry in 3-Space 362
18.3.1 Lines in 3-Space 362
18.3.2 Planes in 3-Space 364
18.3.3 Surfaces in 3-Space 369
18.4 Cylindrical and Spherical Coordinates 374
18.4.1 Cylindrical Coordinates 374
18.4.2 Spherical Coordinates 376

19 Vector-Valued Functions and Parametrization **379**
19.1 Vector-Valued Functions and Calculus 379
19.1.1 Calculus for Vector-Valued Functions 381
19.2 Parameterizations of Curves 384
19.3 Curvature 387

Introducing the Notes

To be written

[AS26] [Ant98]

Note 18

Vectors and Geometry

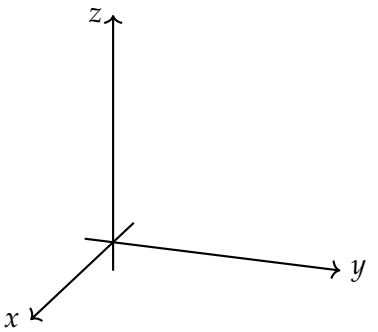
On my other notes on linear algebra, we discussed vectors and vector spaces in detail. However, I wish to discuss some parts again as linear algebra and multivariable calculus have different perspectives on how we view them. In this note, let's discuss more on the geometric interpretation and less on abstractness of vector spaces.

§18.1 Basic Terms and Intuitions

First, why should we even bother higher dimension for multivariable calculus? As the name suggests, multivariable calculus studies calculus over many variables. With generalization from the single variable calculus concepts, it is natural to consider planes and spaces of higher dimension. For instance, say we have a function $f(x, y) = x + y$. We cannot graph this in the normal 2 dimensional Cartesian plane as we have two inputs and an output while the plane only has the space for two. As we have discussed in the notes for linear algebra, we can graph such functions in a 3 dimensional space.

The standard and conventional orientation of the 3D Cartesian Coordinate System is **the Right-hand Rule**. The orientation of the axis below

follows the right-hand rule.



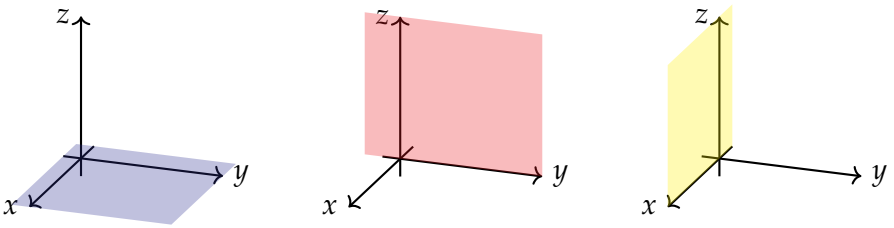
One way to memorize this is to have your curled right hand face up. Then, the direction where your hand is pointing is the direction of the x-axis and the direction of your thumb is the direction of the z-axis. This is a very common explanation; however, to be honest it is quite hard to remember for me. Another explanation that really helped me is to think of a clock at 3 P.M. The hour hand is the x-axis and the minute hand is the y-axis. Then, if you draw a line from the center of the clock pointing at you, that would be the z-axis.

Besides having the name **3-dimensional Cartesian coordinate system**, it is often called as **rectangular coordinate system for 3-space**. When depicting a specific point, we will use the ordered triple (x, y, z).

Some fun facts that we can notice from the notation is that we can represent x, y, and z axis as $\{(x, 0, 0) \mid x \in \mathbb{R}\}$, $\{(0, y, 0) \mid y \in \mathbb{R}\}$, and $\{(0, 0, z) \mid z \in \mathbb{R}\}$ respectively assuming that we only consider real numbers.

An extension of the idea above are planes perpendicular to an axis. We can see that the xy-plane, yz-plane, and zx-plane can be represented as the following.

xy-plane	yz-plane	zx-plane
$\{(x, y, 0) \mid x, y \in \mathbb{R}\}$	$\{(0, y, z) \mid y, z \in \mathbb{R}\}$	$\{(x, 0, z) \mid x, z \in \mathbb{R}\}$



We can also notice that depending on what we have in the place for 0, we can have the planes $x = a$, $y = b$, and $z = c$.

One other extension that we can have from 2D plane is the distance formula between two points.

Theorem 18.1.1: Distance Formula

The distance between two points (x_1, y_1, z_1) and (x_2, y_2, z_2) on a 3D plane can be represented as the following equation.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Proof.

First, construct a right triangle with the points $A(x_1, y_1, z_1)$, $B(x_2, y_2, z_2)$, and $C(x_2, y_2, z_1)$. Notice that the length of $BC = |z_2 - z_1|$ and the length of the segment $AC = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Using the Pythagorean theorem,

$$\begin{aligned} d &= \sqrt{\left(\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}\right)^2 + (z_2 - z_1)^2} \\ &= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \end{aligned}$$

holds and the distance between the two points can be written as $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$.

Using this, we can define a sphere just as we do for circles.

18.1.1 Geometric Intuition in 3-Dimensions

Consider equations of the following form with constants G, H, I, J.

$$x^2 + y^2 + z^2 + Gx + Hy + Iz + J = 0$$

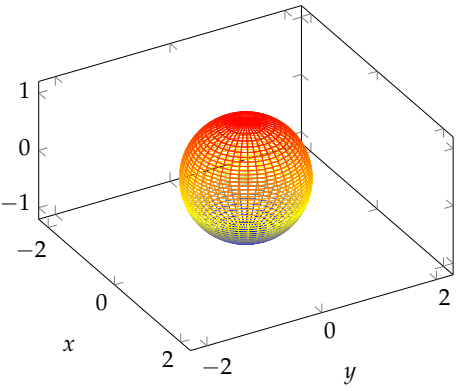
Just as we use completing squares for circles, we can rewrite the equation above as the following form with constants a, b, c, K.

$$(x-a)^2 + (y-b)^2 + (z-c)^2 = K$$

Square rooting both sides, we obtain

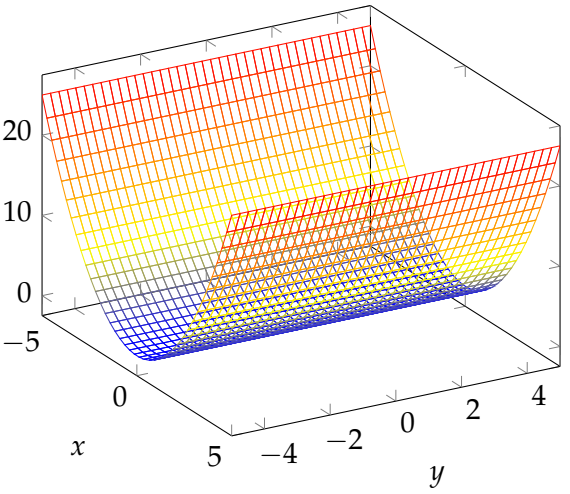
$$\sqrt{(x-a)^2 + (y-b)^2 + (z-c)^2} = \sqrt{K}$$

which is equivalent of saying the locus of points (x, y, z) that is in the distance $\sqrt{K}$ from (a, b, c) by Theorem 18.1.1. Naturally, we can notice that if $K > 0$, such sphere exists. If $K = 0$, the equation has one solution and the graph is a single point. If $K < 0$, there exists no such graph. Graphing a square with the center (x_0, y_0, z_0) and radius r, we can obtain the following by plotting the locus of (x, y, z) that satisfy $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$.



Now that we can graph spheres, how about equations with two variables? When you think about it, it is actually pretty simple since it is the

same as graphing in 2 dimensional plane and stretching it with respect to the remaining axis. Such equations are known as **cylindrical surface**.



The graph above is $z = x^2$. This is the same as graphing the quadratic function in xz-plane and making its copy across the y-axis.

Continuing with this intuition, we can implement the ideas with vectors.

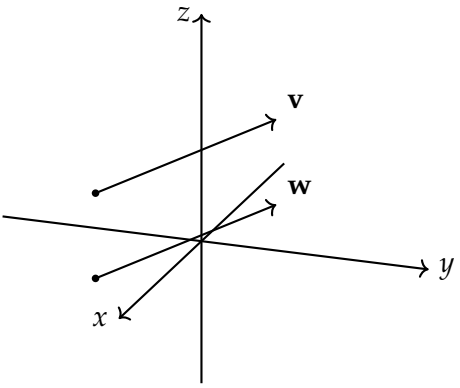
§18.2 Vectors

We have discussed on vectors more in-depth in the part for Linear Algebra. In this section however, let's focus more on the geometric intuition than the abstract nature of vectors.

Definition 18.2.1

Vectors are quantities with both magnitude and a direction.

Some example of vectors include force, velocity, and acceleration because they all have a specific direction and also a magnitude. When we graph them, we can think of them as a arrow with starting and ending points.

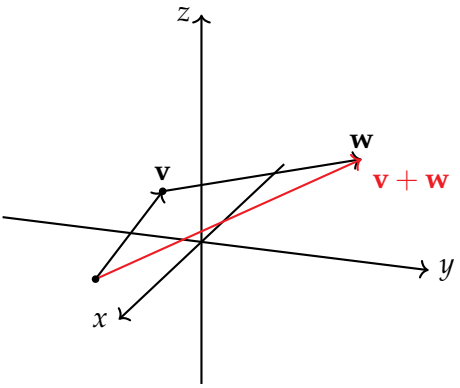


From now, let's represent vectors with $\mathbf{u}, \mathbf{v}, \mathbf{w}$ and scalars, or real numbers, with a, b, α, β. From the graph above, we can define another definition.

Definition 18.2.2

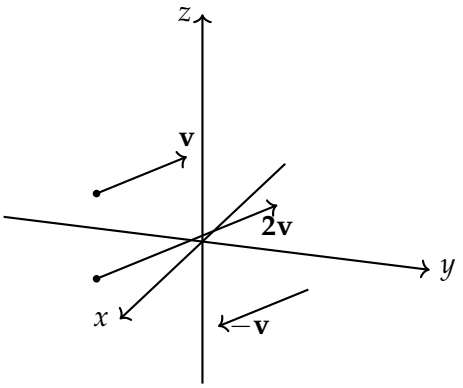
The **terminal point** of a vector is the point where the vector ends. The **initial point** is the point where the vector begins.

When we want to add two vectors $\mathbf{v}$ and $\mathbf{w}$, we would connect the terminal point of $\mathbf{v}$ by the starting point of $\mathbf{w}$ and obtain a new vector by connecting the initial point of $\mathbf{v}$ and the terminal point of $\mathbf{w}$ as shown by the figure below.



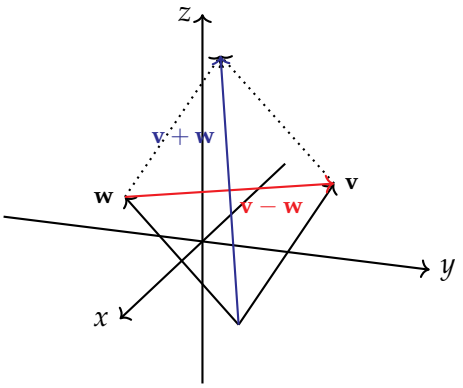
Multiplying a vector by scalar k can be though of as lengthening the

vector by a factor of k. If $k < 0$, the vector is pointing the opposite direction.



Also for the purpose of the notes, let's consider the vectors with same magnitude and direction but different initial vectors as the same vectors. The being said, we can write any vectors in n-dimension as $\langle v_1, v_2, v_3, \ldots, v_n \rangle$ where $(v_1, v_2, v_3, \ldots, v_n)$ is the terminal point and $(0, 0, , 0 \ldots, 0)$ is the initial point. Such numbers $v_1, \ldots, v_n$ are known as the **components** of $\mathbf{v}$.

Now that we have defined $-\mathbf{v}$, we can consider vector subtraction. We can consider vector subtraction as adding two vectors where one is scaled by a negative factor.



Using components, we can write addition, subtraction and scalar multi-

plication as the following for $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$ and $\mathbf{w} = \langle w_1, w_2, w_3 \rangle$.

$$\begin{aligned} \mathbf{v} + \mathbf{w} &= \langle v_1 + w_1, v_2 + w_2, v_3 + w_3 \rangle \\ \mathbf{v} - \mathbf{w} &= \langle v_1 - w_1, v_2 - w_2, v_3 - w_3 \rangle \\ k\mathbf{v} &= \langle kv_1, kv_2, kv_3 \rangle \end{aligned}$$

As you can see from the equations above, we can see that vector addition is commutative and associative and scalar multiplication is distributive. I will keep them brief here because we already discussed about these in-depth on my other notes for linear algebra. Now this property leads to the natural question of subtracting a vector by itself.

Definition 18.2.3

The **zero vector** is a vector with magnitude of zero.

Intuitively, we can think of a zero vector as vector with the same initial and terminal points. Another way of thinking about a zero vector is a vector with zero as its components. This will naturally leads to $\mathbf{v} + \mathbf{0} = \mathbf{v}$ for any vector $\mathbf{v}$. For the purpose of this notes, let's consider zero vector as the vector that is parallel to every vectors.

Continuing with the definition of components and the magnitude of a vector, we can define an important terminology.

Definition 18.2.4

The magnitude of a vector $\mathbf{v}$ is known as the **norm** of $\mathbf{v}$, denoted as $\|\mathbf{v}\|$.

Notice that the magnitude, or length, cannot be negative. Therefore, we can see that the norm of any vector is non-negative. Moreover, $\|\mathbf{0}\| = 0$. Also note that scalar multiplication is defined element-wise. Using the distance formula, we can also obtain the following very important equation for a vector $\mathbf{v}$ and scalar k.

$$\|k\mathbf{v}\| = |k| \cdot \|\mathbf{v}\|$$

Let's take a look at a quick example. Consider the vector $\mathbf{v} = \langle 1, 2, 3 \rangle$. Notice that the following equation holds.

$$\|\mathbf{v}\| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{14}$$

We can also see that the following equation holds with scalar 2 and $2\langle 1, 2, 3 \rangle = \langle 2, 4, 6 \rangle$.

$$\begin{aligned} \|2\mathbf{v}\| &= \sqrt{2^2 + 4^2 + 6^2} = \sqrt{2^2(1^2 + 2^2 + 3^2)} \\ &= 2\sqrt{1^2 + 2^2 + 3^2} \\ &= 2 \cdot \|\mathbf{v}\| \end{aligned}$$

With this equation in mind, let's consider the following question.

> Given a vector $\mathbf{v}$, how do we find a unit vector, or vector with length 1, with the same direction as $\mathbf{v}$?

One way of thinking this is rearranging our equation $\|k\mathbf{v}\| = |k| \cdot \|\mathbf{v}\|$. Consider the following equations with unit vector $\mathbf{u}$ and $\|\mathbf{v}\| = k \neq 0$.

$$\left|\frac{1}{k}\right| \cdot \|\mathbf{v}\| = \frac{1}{k} \cdot k = 1 = \|\mathbf{u}\|$$

This solves it! We can find a unit vector with the same direction $\mathbf{v}$ by dividing it by its norm. Another variation of this equation would be the following.

$$\|\mathbf{v}\| = |k| \cdot \|\mathbf{u}\|$$

> This implies that any vector $\mathbf{v}$ can be represented as the product of its norm and a unique unit vector with the same direction as $\mathbf{v}$. Such process of multiplying a nonzero vector by the reciprocal of its norm is known as **normalizing**.

Finally, another standard way of writing vectors is by linear combination

of the standard basis vectors defined as the following.

$$\mathbf{i} = \langle 1, 0, 0 \rangle, \quad \mathbf{j} = \langle 0, 1, 0 \rangle, \quad \mathbf{k} = \langle 0, 0, 1 \rangle$$

With this, we can write $\mathbf{v} = \langle 1, 2, 3 \rangle$ as $\mathbf{i} + 2\mathbf{j} + 3\mathbf{k}$.

More detailed and abstract introduction of vectors are included in the linear algebra part of LibreNotebook, and I highly recommend checking that out! From here, we will discuss more foundational and geometric interpretations of vectors that we have not discussed in my notes for linear algebra.

18.2.1 Dot Products and Projections

As always, let's start with definition.

Definition 18.2.5

The **dot product** of two vectors $\mathbf{u}$ and $\mathbf{v}$ of the same dimension is defined as the sum of all products of elements in the same index.

Writing them with mathematical notations, we can write them as the following.

$$\langle u_1, u_2, \ldots, u_n \rangle \cdot \langle v_1, v_2, \ldots, v_n \rangle = \sum_{i=1}^{n} u_i v_i$$

Here is a quick example.

$$\langle 1, 2, 3 \rangle \cdot \langle -1, 0, 1 \rangle = -1 + 0 + 3 = 2$$

Continuing, below are some properties of dot products. The statements below are quite self-evident as they mirror the properties of addition and subtraction.

Theorem 18.2.6

For vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ of the same dimension and an arbitrary scalar λ, the following equations hold.

1. $\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{u}$
2. $\lambda(\mathbf{u} \cdot \mathbf{v}) = (\lambda \mathbf{u}) \cdot \mathbf{v} = \mathbf{u} \cdot (\lambda \mathbf{v})$
3. $\mathbf{u} \cdot (\mathbf{v} + \mathbf{w}) = \mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{w}$
4. $\mathbf{v} \cdot \mathbf{v} = \|v\|^2$

One of the reasons why the dot product of two vectors is so important is because it helps us determine the angle between the two vectors. Consider the following theorem.

Theorem 18.2.7

Let $\mathbf{u}, \mathbf{v}$ be two nonzero vectors. Let θ be the smaller angle between the two vectors when their initial point coincide. Then, the following equation holds.

$$\mathbf{u} \cdot \mathbf{v} = \|u\| \|v\| \cos\theta$$

Proof.

First and foremost, consider the following equation by the law of cosines.

$$\cos\theta = \frac{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - \|\mathbf{v} - \mathbf{u}\|^2}{2\|\mathbf{u}\|\|\mathbf{v}\|}$$

By Theorem 18.2.6, the following equations hold.

$$\begin{aligned} \cos\theta &= \frac{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - (\mathbf{v} - \mathbf{u}) \cdot (\mathbf{v} - \mathbf{u})}{2\|\mathbf{u}\|\|\mathbf{v}\|} \\ &= \frac{\|\mathbf{u}\|^2 + \|\mathbf{v}\|^2 - (\mathbf{v} \cdot \mathbf{v} - 2\mathbf{u} \cdot \mathbf{v} + \mathbf{u} \cdot \mathbf{u})}{2\|\mathbf{u}\|\|\mathbf{v}\|} \\ &= \frac{2\mathbf{u} \cdot \mathbf{v}}{2\|\mathbf{u}\|\|\mathbf{v}\|} \\ \therefore \|u\|\|v\| \cos\theta &= \mathbf{u} \cdot \mathbf{v} \end{aligned}$$

Thus, the theorem holds.

Now this theorem tells us the relationship between the angle between two vectors and their dot product.

If $0 \leq \theta < \frac{\pi}{2}$, then $\mathbf{u} \cdot \mathbf{v} > 0$. If $\theta = \frac{\pi}{2}$, then $\mathbf{u} \cdot \mathbf{v} = 0$. If $\frac{\pi}{2} < \theta < \pi$, then $\mathbf{u} \cdot \mathbf{v} < 0$.

This somewhat simple observation with the cosine function and the fact that $\|\mathbf{u}\|, \|\mathbf{u}\| > 0$ reveals the relationship between the dot product of two vectors and their angles. Continuing from this observation, we can establish a definition.

Definition 18.2.8

Two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ are **orthogonal** if they form a right angle. In other words, $\mathbf{u} \cdot \mathbf{v} = 0$.

Continuing with the relationship and definition, let's discuss more on the angle of vectors.

Definition 18.2.9

The **direction angles** of a nonzero vector $\mathbf{v}$ is the angle between $\mathbf{v}$ and the standard basis vectors.

To find the direction angles of $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$, let's return to Theorem 18.2.7. Consider the following equation for the angle α between $\mathbf{v}$ and the standard basis vector $\mathbf{i}$.

$$\begin{aligned} \mathbf{v} \cdot \mathbf{i} &= \|\mathbf{v}\| \|\mathbf{i}\| \cos \alpha \\ v_1 &= \|\mathbf{v}\| \cos \alpha \\ \therefore \alpha &= \cos^{-1} \left(\frac{v_1}{\|\mathbf{v}\|} \right) \end{aligned}$$

Following the same pattern for β and γ, the can establish the following

equations.

$$\cos\alpha = \frac{v_1}{\|\mathbf{v}\|}, \quad \cos\beta = \frac{v_2}{\|\mathbf{v}\|}, \quad \cos\gamma = \frac{v_3}{\|\mathbf{v}\|}$$

Indeed, we have special term for the values above. The values above are known as the **direction cosines** of $\mathbf{v}$. For vectors of direction cosines, the can write the following equation.

$$\langle \cos\alpha, \cos\beta, \cos\gamma \rangle = \left\langle \frac{v_1}{\|\mathbf{v}\|}, \frac{v_2}{\|\mathbf{v}\|}, \frac{v_3}{\|\mathbf{v}\|} \right\rangle = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

Continuing, let's consider the case when $\|\mathbf{v}\| = 1$. In that case,

$$\langle \cos\alpha, \cos\beta, \cos\gamma \rangle = \left\langle \frac{v_1}{\|\mathbf{v}\|}, \frac{v_2}{\|\mathbf{v}\|}, \frac{v_3}{\|\mathbf{v}\|} \right\rangle = \mathbf{v}$$

also hold! Finally, here is another interesting properties of the direction cosines.

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = \left(\frac{v_1}{\|\mathbf{v}\|}\right)^2 + \left(\frac{v_2}{\|\mathbf{v}\|}\right)^2 + \left(\frac{v_3}{\|\mathbf{v}\|}\right)^2 = \frac{\|\mathbf{v}\|^2}{\|\mathbf{v}\|^2} = 1$$

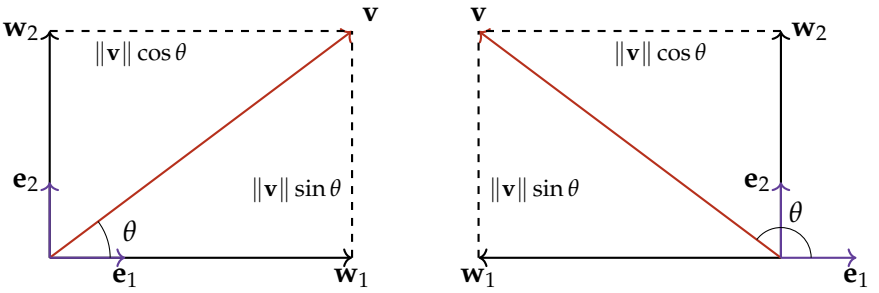
We can see that for nonzero $\mathbf{v}$, we can generalize this for any n. Therefore the sum of the squares of direction cosines are always 1.

The last part that I wish to discuss about dot products is projections. First, consider a vector $\mathbf{v} = \mathbf{w}_1 + \mathbf{w}_2$ in a plane where $\mathbf{w}_1$ and $\mathbf{w}_2$ are along the standard basis $\mathbf{i}$ and $\mathbf{j}$ respectively. By construction, we can write $\mathbf{w}_1 = \mathbf{v} \cdot \mathbf{i}$ and $\mathbf{w}_2 = \mathbf{v} \cdot \mathbf{j}$.

> In other words, for any nonzero vector $\mathbf{v}$ in a plane, it can be represented as the following.
>
> $$\mathbf{v} = (\mathbf{v} \cdot \mathbf{i})\mathbf{i} + (\mathbf{v} \cdot \mathbf{j})\mathbf{j}$$
>
> Such vectors $(\mathbf{v} \cdot \mathbf{i})\mathbf{i}$ and $(\mathbf{v} \cdot \mathbf{j})\mathbf{j}$ are known as **vector components** of $\mathbf{v}$ along $\mathbf{i}$ and $\mathbf{j}$ where the dot products $\mathbf{v} \cdot \mathbf{i}$ and $\mathbf{v} \cdot \mathbf{j}$ are **scalar components** of $\mathbf{v}$ along $\mathbf{i}$ and $\mathbf{j}$.

This idea can be generalized for higher dimensions. Moreover, although we used standard basis vectors here, this works for any orthogonal unit vectors.

Another way of expanding this is through geometric interpretation.



As you can see from the diagrams above, we can write the scalar components with respect to the norm of $\mathbf{v}$ and the angle θ. Moreover, we can define a new definition.

Definition 18.2.10

The **orthogonal projection** of a vector $\mathbf{v}$, denoted as $\text{proj}_{\mathbf{u}}\,\mathbf{v}$, on a unit vector $\mathbf{u}$ is the vector component of $\mathbf{v}$ that is parallel to $\mathbf{u}$.

From the figures above, we can see that $\text{proj}_{\mathbf{e}_1} = \mathbf{w}_1$ and $\text{proj}_{\mathbf{e}_2} = \mathbf{w}_2$. Finally, this leads us to the conclusion that for any vector $\mathbf{v}$ and respective orthogonal unit vectors $\mathbf{e}_1$ and $\mathbf{e}_2$, the following equation holds.

$$\mathbf{v} = \text{proj}_{\mathbf{e}_1}\,\mathbf{v} + \text{proj}_{\mathbf{e}_2}\,\mathbf{v}$$

This is it for dot products and orthogonal projections! Before we conclude our section on vectors, we will discuss about cross products and its geometric interpretations.

18.2.2 The Cross Product and Geometry

Unlike how we defined vector addition and scalar multiplication, we had a special way of defining "multiplication" of vectors with the dot

product. However specifically for 3-dimensional space, we can also define cross products.

Definition 18.2.11

For $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$, the **cross product**, denoted as $\mathbf{u} \times \mathbf{v}$ and read as "$\mathbf{u}$ cross $\mathbf{v}$", is defined as the following.

$$\mathbf{u} \times \mathbf{v} = (u_2v_3 - u_3v_2)\mathbf{i} - (u_1v_3 - u_3v_1)\mathbf{j} + (u_1v_2 - u_2v_1)\mathbf{k}$$

As you can see from the definition above, unlike the dot product that returns a scalar, the cross product of two vector return a vector. Also, if it is difficult to memorize the formula above, we can also think of cross product as the following.

$$\begin{aligned}\mathbf{u} \times \mathbf{v} &= \det\left(\begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix}\right) \\ &= (u_2v_3 - u_3v_2)\mathbf{i} - (u_1v_3 - u_3v_1)\mathbf{j} + (u_1v_2 - u_2v_1)\mathbf{k}\end{aligned}$$

If you are unfamiliar with the operation above, I highly recommend reading my notes on determinants, which is in Section 14.5!

> The cross product $\mathbf{u} \times \mathbf{v}$ is only defined for $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$. The other dimension does not return meaningful properties and definition of the cross product.

Unlike the dot product, the properties of the cross product are not so nice.

Theorem 18.2.12

For $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbb{R}^3$ and some scalar λ, the following equations hold.

1. $\mathbf{u} \times \mathbf{v} = -(\mathbf{v} \times \mathbf{u})$
2. $\mathbf{u} \times \mathbf{u} = \mathbf{0}$
3. $\mathbf{u} \times \mathbf{0} = \mathbf{0}$
4. $\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) + (\mathbf{u} \times \mathbf{w})$
5. $\lambda(\mathbf{u} \times \mathbf{v}) = (\lambda\mathbf{u}) \times \mathbf{v} = \mathbf{u} \times (\lambda\mathbf{v})$

There are a lot to prove, but let's start with the first one.

Proof.

Note that by definition, the following equations hold.

$$\begin{aligned}\mathbf{u} \times \mathbf{v} &= (u_2v_3 - u_3v_2)\mathbf{i} - (u_1v_3 - u_3v_1)\mathbf{j} + (u_1v_2 - u_2v_1)\mathbf{k} \\ \mathbf{v} \times \mathbf{u} &= (v_2u_3 - v_3u_2)\mathbf{i} - (v_1u_3 - v_3u_1)\mathbf{j} + (v_1u_2 - v_2u_1)\mathbf{k} \\ &= -(v_3u_2 - v_2u_3)\mathbf{i} + (v_3u_1 - v_1u_3)\mathbf{j} - (v_2u_1 - v_1u_2)\mathbf{k}\end{aligned}$$

Thus, $\mathbf{u} \times \mathbf{v} = -(\mathbf{v} \times \mathbf{u})$.

Below is the proof for the second one.

Proof.

By the first property, $\mathbf{u} \times \mathbf{u} = -(\mathbf{u} \times \mathbf{u})$. In other words, $2\mathbf{u} \times \mathbf{u} = \mathbf{0}$ and $\mathbf{u} \times \mathbf{u} = \mathbf{0}$.

Continuing, here is the proof for the third property.

Proof.

By definition, $\mathbf{u} \times \mathbf{0} = (u_2v_3 - u_3v_2)\mathbf{i} - (u_1v_3 - u_3v_1)\mathbf{j} + (u_1v_2 - u_2v_1)\mathbf{k} = 0\mathbf{i} + 0\mathbf{j} + 0\mathbf{k} = \mathbf{0}$.

Below is the proof for the fourth property.

Proof.

First and foremost, consider the following equation for the first

component only.

$$\begin{aligned}\langle 1,0,0\rangle \cdot (\mathbf{u} \times (\mathbf{v}+\mathbf{w})) &= u_2(v_3+w_3) - u_3(v_2+w_2) \\ &= (u_2v_3 + u_2w_3) - (u_3v_2 + u_3w_2) \\ &= u_2v_3 - u_3v_2 + u_2w_3 - u_3w_2 \\ &= \langle 1,0,0\rangle \cdot \mathbf{u} \times \mathbf{v} + \langle 1,0,0\rangle \cdot \mathbf{u} \times \mathbf{w}\end{aligned}$$

Similarly, the second and the third entry hold with analogous argument. Thus, the property holds.

Lastly, here is the proof for the last property.

Proof.

By definition, the following equations hold.

$$\begin{aligned}k(\mathbf{u} \times \mathbf{v}) &= k(u_2v_3 - u_3v_2)\mathbf{i} - k(u_1v_3 - u_3v_1)\mathbf{j} + k(u_1v_2 - u_2v_1)\mathbf{k} \\ &= ((ku_2)v_3 - (ku_3)v_2)\mathbf{i} - ((ku_1)v_3 - (ku_3)v_1)\mathbf{j} \\ &\qquad + ((ku_1)v_2 - (ku_2)v_1)\mathbf{k} \\ &= (k\mathbf{u})\mathbf{v}\end{aligned}$$

The analogous computations apply for $k(\mathbf{u} \times \mathbf{v}) = \mathbf{u} \times (k\mathbf{v})$.

Continuing, it is worth memorizing the cross product between the standard basis vectors.

$$\mathbf{i} \times \mathbf{j} = \mathbf{k}, \quad \mathbf{j} \times \mathbf{k} = \mathbf{i}, \quad \mathbf{k} \times \mathbf{i} = \mathbf{j}$$

I personally think and easy way to memorize this is through cycle. For cycle counter clockwise, we get a negative one.

$$\mathbf{j} \times \mathbf{i} = -\mathbf{k}, \quad \mathbf{k} \times \mathbf{j} = -\mathbf{i}, \quad \mathbf{i} \times \mathbf{k} = -\mathbf{j}$$

We can also notice that the cross product returns a vector that are orthogonal to those being crossed!

Parenthesis is very important! As you can see from the standard basis vectors above, cross products are generally not associative. Meaning,

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) \neq (\mathbf{u} \times \mathbf{v}) \times \mathbf{w}$$

is true in general and parenthesis is necessarily to avoid ambiguity.

However when we have a dot product and a cross product like $\mathbf{u} \cdot (\mathbf{v} \times \mathbf{w})$, then parenthesis is not necessary as the cross product is defined only between two three dimensional vectors. Speaking of such forms, consider the following theorem.

Theorem 18.2.13

For two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in the third dimension, $\mathbf{u} \cdot \mathbf{u} \times \mathbf{v}$ and $\mathbf{v} \cdot \mathbf{u} \times \mathbf{v}$ holds.

Proof.

First, let $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$ and consider the following equations.

$$\begin{aligned}
\mathbf{u} \cdot \mathbf{u} \times \mathbf{v} &= \mathbf{u} \cdot \langle u_2 v_3 - u_3 v_2, u_3 v_1 - u_1 v_3, u_1 v_2 - u_2 v_1 \rangle \\
&= u_1(u_2 v_3 - u_3 v_2) + u_2(u_3 v_1 - u_1 v_3) + u_3(u_1 v_2 - u_2 v_1) \\
&= u_1 u_2 v_3 - u_1 u_3 v_2 + u_2 u_3 v_1 - u_1 u_2 v_3 + u_1 u_3 v_2 - u_2 u_3 v_1 \\
&= 0
\end{aligned}$$

Therefore, the first equation holds. Continuing, the following equations hold for the second equation by Theorem 18.2.12.

$$\mathbf{v} \cdot \mathbf{u} \times \mathbf{v} = \mathbf{v} \cdot (-\mathbf{v} \times \mathbf{u}) = -\mathbf{v} \cdot \mathbf{v} \times \mathbf{u} = 0$$

Thus the theorem holds.

From the theorem, we can notice that $\mathbf{u}$ is orthogonal to $\mathbf{u} \times \mathbf{v}$, which is orthogonal to $\mathbf{v}$. These types of operation is known as scalar triple product, and we will be discussing more on that soon! In addition to the

orthogonal nature of $\mathbf{u} \times \mathbf{v}$ and its adherence to the right-hand rule, we can also find the relationship between the norm of $\mathbf{u} \times \mathbf{v}$ and $\mathbf{u}$ and $\mathbf{v}$.

Theorem 18.2.14

For two nonzero vectors $\mathbf{u}$ and $\mathbf{v}$ in the third dimension with angle θ, $\|\mathbf{u} \times \mathbf{v}\| = \|\mathbf{u}\|\|\mathbf{v}\| \sin\theta$.

Proof.

Let $\mathbf{u} = \langle u_1, u_2, u_3 \rangle$ and $\mathbf{v} = \langle v_1, v_2, v_3 \rangle$. Notice that because both the left-hand side and the right-hand side of the equation are both positive, it suffices to show that the square of each side equals. Consider the following equations.

$$\begin{aligned}
\|\mathbf{u} \times \mathbf{v}\| &= (u_2v_3 - u_3v_2)^2 + (u_3v_1 - u_1v_3)^2 + (u_1v_2 - u_2v_1)^2 \\
&= u_2^2v_3^2 + u_3^2v_2^2 - 2u_2v_3u_3v_2 + u_3^2v_1^2 + u_1^2v_3^2 - 2u_3v_1u_1v_3 \\
&\quad + u_1^2v_2^2 + u_2^2v_1^2 - 2u_1v_2u_2v_1 \\
&= u_1^2v_2^2 + u_2^2v_3^2 + u_3^2v_1^2 + v_1^2u_2^2 + v_2^2u_3^2 + v_3^2u_1^2 \\
&\quad - 2u_1u_2v_1v_2 - 2u_2u_3v_2v_3 - 2u_3u_1v_3v_1
\end{aligned}$$

Continuing, the right-hand side can be rearranged as the following.

$$\begin{aligned}
&\|\mathbf{u}\|^2\|\mathbf{v}\|^2 \sin^2\theta \\
&= \|\mathbf{u}\|^2\|\mathbf{v}\|^2 - \|\mathbf{u}\|^2\|\mathbf{v}\|^2 \cos^2\theta \\
&= \|\mathbf{u}\|^2\|\mathbf{v}\|^2 - (\mathbf{u} \times \mathbf{v})^2 \\
&= (u_1^2 + u_2^2 + u_3^2)(v_1^2 + v_2^2 + v_3^2) - (u_1v_1 + u_2v_2 + u_3v_3)^2 \\
&= u_1^2v_1^2 + u_1^2v_2^2 + u_1^2v_3^2 + u_2^2v_1^2 + u_2^2v_2^2 + u_2^2v_3^2 + u_3^2v_1^2 + u_3^2v_2^2 + u_3^2v_3^2 \\
&\quad - (u_1^2v_1^2 + u_2^2v_2^2 + u_3^2v_3^2) - 2(u_1u_2v_1v_2 + u_2u_3v_2v_3 + u_3u_1v_3v_1) \\
&= u_1^2v_2^2 + u_2^2v_3^2 + u_3^2v_1^2 + v_1^2u_2^2 + v_2^2u_3^2 + v_3^2u_1^2 \\
&\quad - 2u_1u_2v_1v_2 - 2u_2u_3v_2v_3 - 2u_3u_1v_3v_1
\end{aligned}$$

Thus, the theorem holds.

This theorem implies that the norm of the cross product of the two vec-

tors is the product of the norms of the vectors and sine of the angle between them. With this, we can also find another geometric representation of cross products.

> The area of the parallelogram with sides $\mathbf{u}$ and $\mathbf{v}$ can be represented as $\|\mathbf{u} \times \mathbf{v}\|$.

This is somewhat simple to notice as if we consider $\mathbf{u}$ as the base, then $\|\mathbf{v}\| \sin\theta$ is the height where θ is the angle between them. Therefore by Theorem 18.2.14, the area of the parallelogram is $\|\mathbf{u}\|\|\mathbf{v}\| \sin\theta = \|\mathbf{u} \times \mathbf{v}\|$.

Before we move on to the next section, let's wrap up our discussion on vectors with the scalar triple product and its geometric representations.

Definition 18.2.15

The **scalar triple product** of three vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$ in 3-space is defined as $\mathbf{u} \cdot \mathbf{v} \times \mathbf{w}$.

Notice that the output of the scalar triple product of three vectors is a scalar. Also, using the definition of cross products and dot products, we can notice that the scalar triple product is equivalent to the following.

$$\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix}$$

From the matrix operation above, we can notice that the following equations hold.

$$\begin{aligned} \mathbf{u} \cdot \mathbf{v} \times \mathbf{w} &= \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ w_1 & w_2 & w_3 \end{bmatrix} = - \begin{bmatrix} u_1 & u_2 & u_3 \\ w_1 & w_2 & w_3 \\ v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & w_3 \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix} \\ &= \mathbf{w} \cdot \mathbf{u} \times \mathbf{v} \end{aligned}$$

Therefore, we could see that $\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = \mathbf{w} \cdot \mathbf{u} \times \mathbf{v}$ and $\mathbf{w} \cdot \mathbf{u} \times \mathbf{v} =$

$\mathbf{v} \cdot \mathbf{w} \times \mathbf{u}$ both holds by cycling through the vectors.

For any vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ in 3-space, the following equation holds.

$$\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = \mathbf{v} \cdot \mathbf{w} \times \mathbf{u} = \mathbf{w} \cdot \mathbf{u} \times \mathbf{v}$$

Moreover because $\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = \mathbf{w} \cdot \mathbf{u} \times \mathbf{v} = \mathbf{u} \times \mathbf{v} \cdot \mathbf{w}$, the following equation also holds.

$$\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = \mathbf{u} \times \mathbf{v} \cdot \mathbf{w}$$

Lastly before wrapping up, let's take a look at geometric interpretation of the scalar triple product. First, consider a parallelepiped with edges $\mathbf{u}, \mathbf{v}, \mathbf{w}$.

From our previous observation, we can see that the area of the base of the parallelepiped can be represented as $\|\mathbf{v} \times \mathbf{w}\|$. Moreover the height can be represented as $\|\mathbf{u}\||\cos\theta|$ where θ is the angle between $\mathbf{u}$ and $\mathbf{v} \times \mathbf{w}$. Therefore, the volume can be represented as $\|\mathbf{u}\||\cos\theta|\|\mathbf{v} \times \mathbf{w}\| = |\mathbf{u} \times \mathbf{v} \times \mathbf{w}|$. Continuing with this idea, we can establish an important theorem.

Theorem 18.2.16

$\mathbf{u}, \mathbf{v}, \mathbf{w}$ are coplanar if and only if $\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = 0$.

Proof.

From the previous observation on the relationship between the scalar triple product and the volume of the parallelepiped with the edges $\mathbf{u}, \mathbf{v}, \mathbf{w}$, it is evident that the volume equals zero if and only if $\mathbf{u} \cdot \mathbf{v} \times \mathbf{w} = 0$. Moreover, the volume equals zero if and only if the vectors are coplanar.

This is it for the vector section! There were lots of topics and geometric interpretations of vectors that were not included in the notes for linear algebra. In our next section, we will continue our discussion through more extensions of geometric representations of vectors.

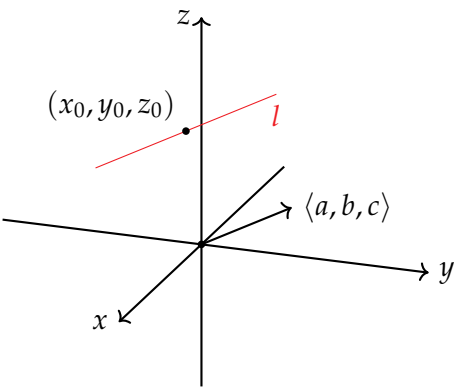
§18.3 Geometry in 3-Space

In this section of the note, we will discuss more geometric analysis in 3-space with vectors. I personally think it was very interesting to see different interpretations of vectors in 3-space because they could be very abstract, yet a fundamental component that helps analyzing geometry in 3-dimensions.

18.3.1 Lines in 3-Space

Continuing with our prior discussion on vectors, let's discuss more on lines and planes in 3-space and how they relate to vectors.

First, the components of lines in the coordinate plane can be thought of as a point and a direction, which would be slope. Given a point that a line passes through and the slope, or direction, of the line, we can find the unique line. Similarly, we could do the same for 3-space. The direction in this case would be a nonzero vector. In other words, given a point and a nonzero vector parallel to the line, we can find the unique line.



Looking at the graph above, we can derive a very important concept of representing lines in parametric equations.

A line l in 3-space that passes through the point (x_0, y_0, z_0) and parallel to vector $\mathbf{v} = \langle a, b, c \rangle$ can be represented as the following parametric equation for $t \in (-\infty, \infty)$.

$$x = x_0 + at, \quad y = y_0 + bt, \quad z = z_0 + ct$$

To prove this, we can see that the line actually contains infinitely many vectors that passes through the point (x_0, y_0, z_0) and (x, y, z) while being parallel to $\langle a, b, c \rangle$. In other words, $\langle x - x_0, y - y_0, z - z_0 \rangle = t\mathbf{v} = \langle ta, tb, tc \rangle$ for some t. Comparing the vectors element-wise, we obtain the parametric equation.

Another standard way of writing the parametric equation $x = x_0 + at, y = y_0 + bt, z = z_0 + ct$ is the following.

$$\langle x, y, z \rangle = \langle x_0, y_0, z_0 \rangle + t\langle a, b, c \rangle$$

This is known as a **vector equation**.

Let's take a look at a quick example.

Exercise 18.3.1

Find a parametric equation and a vector equation of the line that passes through the point $(1, 2, 3)$ and parallel to $\langle 1, 2, 3 \rangle$.

Solution.

Because the line passes through the point $(1, 2, 3)$ and is parallel to $\langle 1, 2, 3 \rangle$, a vector equation can be written as $\langle x, y, z \rangle = \langle 1, 2, 3 \rangle + t\langle 1, 2, 3 \rangle$. Similarly, a parametric equation can be written as $x = 1 + t$, $y = 2 + 2t$, and $z = 3 + 3t$.

However, this is not the only answer. Because the line is also parallel to $\langle 1, 2, 3 \rangle$, it is also parallel to $\langle 2, 4, 6 \rangle$. Using this instead, we get a vector equation $\langle 1, 2, 3 \rangle + t\langle 2, 4, 6 \rangle$. This shows that there are different parametrizations and vector equations for a given line.

One way vector equation can be very helpful than parametric equation is that we can directly see if two lines are parallel to each other. If the two vectors that are being scaled by t are scale multiples of each other, then the two lines are similar. This observation also leads us to define as new term.

Definition 18.3.2

The **skew** lines are two lines in 3-space that are not parallel to each other and that do not intersect.

Unlike 2-space, we can see that not being parallel does not necessarily imply that the two lines intersect.

> Segments are very similar to lines. When we have to obtain a segment, then we can just limit our parameter t from the line obtained from the methods above.

Now that we have took a look at the lines in 3-space, let's discuss its extension: planes.

18.3.2 Planes in 3-Space

First, let's define the terms **perpendicular** and **parallel** in the context of planes.

Definition 18.3.3

A plane A and a vector $\mathbf{v}$ are **perpendicular** if any vector lying on the plane A is orthogonal to $\mathbf{v}$. Moreover, if there exists a vector that is perpendicular to two planes, then the planes are **parallel**. Finally, a vector and a plane is **parallel** if some normal vector of the plane is orthogonal to the vector.

Similar to how we found vector equations of a line, we could extend our understandings on plane.

The plane in 3-space that is perpendicular to a nonzero vector $\mathbf{v} = \langle a, b, c \rangle$ and contains the point (x_0, y_0, z_0) can be represented as the following equation.

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0$$

Such vectors $\mathbf{v}$ that are perpendicular to the plane are known as **normal vectors**. Moreover, the equation above is called the **point-normal form**.

Similar to how we defined a line, we can notice that the plane contains all vectors $\langle x - x_0, y - y_0, z - z_0 \rangle$ that is perpendicular to $\mathbf{v}$. Therefore using the dot product, we know that the vectors are orthogonal if and only if their dot product is zero.

$$\langle x - x_0, y - y_0, z - z_0 \rangle \cdot \langle a, b, c \rangle = 0$$

Expanding the equation, we obtain the equations of a plane. Besides this point-normal form of the equation of the plane, we obtain the following form by expanding.

$$ax + by + cz + d = 0$$

Here, $d = -ax_0 - by_0 - cz_0$ and the from above is called the **general form**. Continuing, we can establish an important theorem on the general form.

Theorem 18.3.4

For a plane $ax + by + cz + d = 0$ with $\langle a, b, c \rangle \neq \langle 0, 0, 0 \rangle$, the normal vector is $\langle a, b, c \rangle$.

Proof.

First, consider the case when $a \neq 0$. Rearranging the equation,

$$ax + by + cz + d = a\left(x + \frac{d}{a}\right) + by + cz$$

is obtained and $a\left(x+\frac{d}{a}\right)+by+cz=0$, which represents the plane with the normal vector $\langle a,b,c\rangle$ and passes through the point $(-\frac{d}{a},0,0)$. The case when $b\neq 0$ or $c\neq 0$ is proven in analogous way. Lastly, if all a,b,c are nonzero, then it is a plane that is shifted from the origin.

Let's take a look at quick example.

Exercise 18.3.5

Find the equation of the plane that contains $\langle 1,2,3\rangle$ and perpendicular to $\langle 1,2,3\rangle$ in point-normal form and in general form.

Solution.

Using the form $a(x-x_0)+b(y-y_0)+c(z-z_0)=0$, the equation of the plane can be represented as $(x-1)+2(y-2)+3(z-3)=0$, or $x+2y+3z-14=0$.

Similar to lines, there can be multiple representations of the same plane. Now before we move on to curves, let's discuss few meaningful geometric interpretations of planes.

The two cases that we will check are the angle between two planes and the distance between two parallel planes.

Definition 18.3.6

The smaller angle between two planes are known as the **acute angle of intersection** between the two planes.

I know we have the term "acute" in it, but $\frac{\pi}{2}$ is included as acute angle of intersection if the two planes make a right angle. Now to find an interesting equation like Theorem 18.2.7, let's prove a lemma.

Lemma 18.3.7

For two intersecting lines L_A and L_B in 3-space, let θ be $\angle(L_A, L_B)$. If n_A and n_B are vectors that are parallel to the lines respectively, then the following equation holds.

$$\cos\theta = \frac{|n_A \cdot n_B|}{\|n_A\|\|n_B\|}$$

Proof.

Let α be the angle between n_A and n_B. If $0 < \alpha < \frac{\pi}{2}$, then the following equation holds by Theorem 18.2.7.

$$\cos\theta = \cos\alpha = \frac{n_A \cdot n_B}{\|n_A\|\|n_B\|} = \frac{|n_A \cdot n_B|}{\|n_A\|\|n_B\|}$$

Similarly, consider the case when $\frac{\pi}{2} \leq \alpha < \pi$. By the property of cosine functions, the following equation holds.

$$\cos\theta = -\cos\alpha = |\cos\alpha| = \frac{|n_A \cdot n_B|}{\|n_A\|\|n_B\|}$$

Thus, the lemma holds.

With the lemma shown, we can derive an important theorem regarding the angle between two intersecting planes.

Theorem 18.3.8

For two intersecting planes A and B with normal vectors n_A and n_B, if θ is the acute angle of intersection between A and B, then the following equation holds.

$$\cos\theta = \frac{|n_A \cdot n_B|}{\|n_A\|\|n_B\|}$$

Proof.

Let l_A and l_B be two lines normal to planes A and B respectively that

also passes through the intersection of the planes. Let α and β be the $\angle(l_A, l_B)$ and $\angle(l_B, A)$ respectively. Notice that $\alpha + \beta = \frac{\pi}{2} = \beta + \theta$ through geometric interpretations. Therefore, $\alpha = \theta$ and it suffices to find $\cos\alpha$. Because the angle between l_A and l_B and the angle between n_A and n_B equal, the theorem holds by Lemma 18.3.8.

Continuing, let's take a look at the case when two different planes do not intersect. To begin, consider the following theorem.

Theorem 18.3.9

For a plane $ax + by + cz + d = 0$ and a point (x_0, y_0, z_0), the distance D between the plane and the point is represented with the following formula.

$$D = \frac{|ax_0 + by_0 + cz_0 + d|}{\sqrt{a^2 + b^2 + c^2}}$$

Proof.

Let $Q(x_1, y_1, z_1)$ be any point in the plane. If the angle between the normal vector $\mathbf{n}$ of the plane from Q and the vector from Q to P is θ, then the distance D can be represented as $\|\vec{QP}\||\cos\theta|$. Thus, the following equations hold by Theorem 18.2.7.

$$D = \|\vec{QP}\|\cos\theta = \frac{\|\mathbf{n}\|\|\vec{QP}\||\cos\theta|}{\|\mathbf{n}\|} = \frac{|\mathbf{n}\cdot\vec{QP}|}{\|\mathbf{n}\|}$$

Defining $\mathbf{n} = \langle a, b, c\rangle$, the following equations hold by construction.

$$\begin{aligned} |\mathbf{n}\cdot\vec{QP}| &= |\langle a, b, c\rangle \cdot \langle x_0 - x_1, y_0 - y_1, z_0 - z_1\rangle| \\ &= |ax_0 + by_0 + cz_0 + d| \end{aligned}$$

Because $\|\mathbf{n}\| = \sqrt{a^2 + b^2 + c^2}$, the theorem satisfy.

Continuing from the theorem above, we could see that the distance between two planes is the distance between a plane and a point on the other plane. Note that both planes $ax + by + cz + d_1 = 0$ and $ax + by +$

$cz + d_2 = 0$ are parallel. Therefore, the distance D between the two planes can be represented as the following by Theorem 18.3.9.

$$D = \frac{|ax_0 + by_0 + cz_0 + d_1|}{\sqrt{a^2 + b^2 + c^2}} = \frac{|d_1 - d_2|}{\sqrt{a^2 + b^2 + c^2}}$$

Notice that this holds because (x_0, y_0, z_0) lies on $ax + by + cz + d_2 = 0$.

The distance D between two parallel planes $ax + by + cz + d_1 = 0$ and $ax + by + cz + d_2 = 0$ can be represented as the following.

$$D = \frac{|d_1 - d_2|}{\sqrt{a^2 + b^2 + c^2}}$$

This is it for this part! In the next part, let's continue from this linear ideas to expand them to curves.

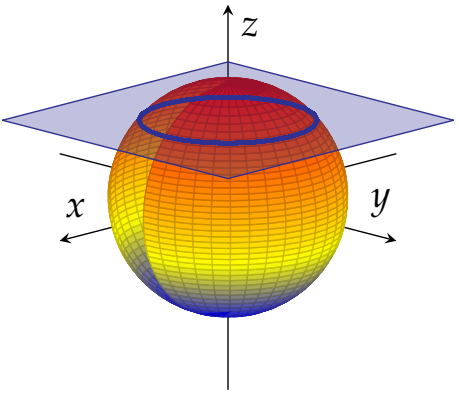
18.3.3 Surfaces in 3-Space

Before we start graphing surfaces, let's think about how we may achieve that. In the coordinate plane, we graph curves by knowing the locus of points that satisfy the given equation of a curve. Graphing surfaces is similar! In 3-space, we find which curves lie on the surface to find the general shape of a given surface.

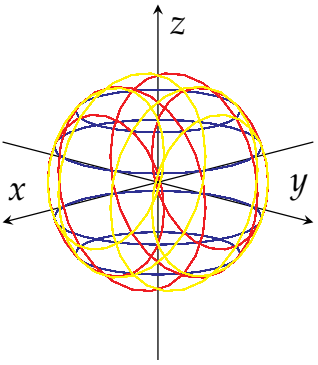
Definition 18.3.10

The **trace** of the surface in a plane is the set of points of intersection of the surface and the plane.

Let's take a look at an example. It is self-evident that the trace of a sphere in a plane is always a circle or a point assuming that they intersect. However for an algebraic approach, let's forget how the surface S defined as $x^2 + y^2 + z^2 = 9$ looks like. Given plane $z = 2$, we can see that the trace of S in the plane $z = 2$ can be found by substituting the value. In other words, S and the plane intersect if and only if $z = 2$ and the trace is $x^2 + y^2 = 5$. This trace is indeed a circle!



Continuing with this idea, we can try to construct the general shape of a surface by finding the traces. Let's take a look at an example from our previous example.



For surface S defined as $x^2 + y^2 + z^2 = 9$, we can find the trace of S in $x = k$, $y = k$, and $z = k$. For each case, we can see that there is no intersection for $|k| > 3$. Therefore considering the equations for $|k| \leq 3$, we obtain the following general structure. This indeed looks like a sphere!

Continuing, here is another example.

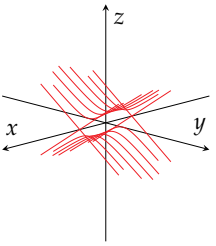
Exercise 18.3.11

Find the general shape of the surface $\frac{x^2}{36} - \frac{y^2}{9} + \frac{z^2}{4} = 1$.

Solution.

To find the general shape of the surface, we can try to find different traces in different planes to later combine them for a general shape.

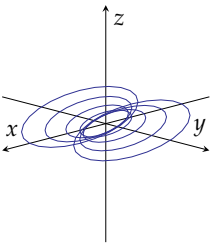
First, let's take a look at the traces in the plane $x = k$. Substituting, we obtain an equation representing hyperbolas.



$$\frac{z^2}{4} - \frac{y^2}{9} = 1 - \frac{k^2}{36}$$

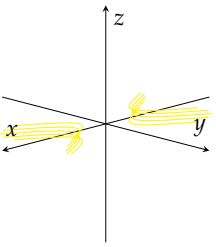
Continuing, let's take a look at traces in the plane $y = k$.

Substituting, we obtain $\frac{x^2}{36} + \frac{z^2}{4} = 1 + \frac{k^2}{9}$. Rearranging the equation, the following equation of ellipse is obtained.



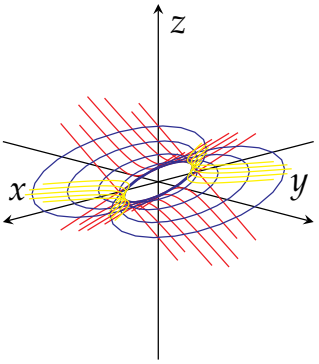
$$\frac{x^2}{36\left(1 + \frac{k^2}{9}\right)} + \frac{z^2}{4\left(1 + \frac{k^2}{9}\right)} = 1$$

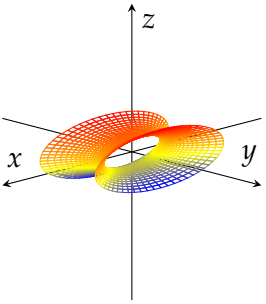
Finally for $z = k$, we obtain the following equations just like the case for $x = k$.



$$\frac{x^2}{36} - \frac{y^2}{9} = 1 - \frac{k^2}{4}$$

Combining the traces, we obtain the general shape of the surface.





Such surfaces are known as **hyperboloid**. Continuing with surfaces, let's discuss quadratic surfaces.

Note that in coordinate plane, we call equations of the form

$$Ax^2 + bxy + Cy^2 + Dx + Ey + F = 0$$

a conic section. Some examples include parabola, hyperbola, circle, and ellipse. Now, we could also generalize such equations in 3-space.

Definition 18.3.12

A **quadric surface** is known as the graph whose equation is of the following form.

$$Ax^2 + By^2 + Cz^2 + Dxy + Eyz + Fzx + Gx + Hy + Iz + J = 0$$

For instance, the graph from our previous example is a quadric surface.

To investigate in some quadric surfaces, let's take a look at few examples. First, consider the examples of the following form.

$$z^2 = \frac{x^2}{a^2} + \frac{y^2}{b^2}$$

Considering the traces in planes $z = k$, we can notice that the equation becomes

$$\frac{x^2}{a^2k^2} + \frac{y^2}{b^2k^2} = 1$$

which are ellipses or $(0, 0)$ if $k = 0$.

Similarly for $x = k$, we obtain

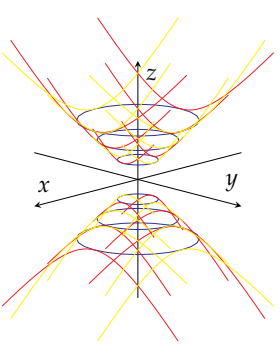
$$\frac{z^2}{\left(\frac{k}{a}\right)^2} - \frac{y^2}{\left(\frac{bk}{a}\right)^2} = 1$$

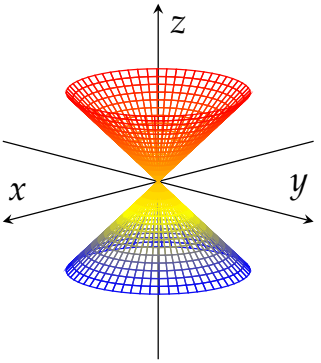
which are hyperbolas. Continuing for $y = k$, we obtain the following

equation for hyperbolas.

$$\frac{z^2}{\left(\frac{k}{b}\right)^2} - \frac{x^2}{\left(\frac{ak}{b}\right)^2} = 1$$

With the found traces, we could try finding the general shape of the surface by combining them.





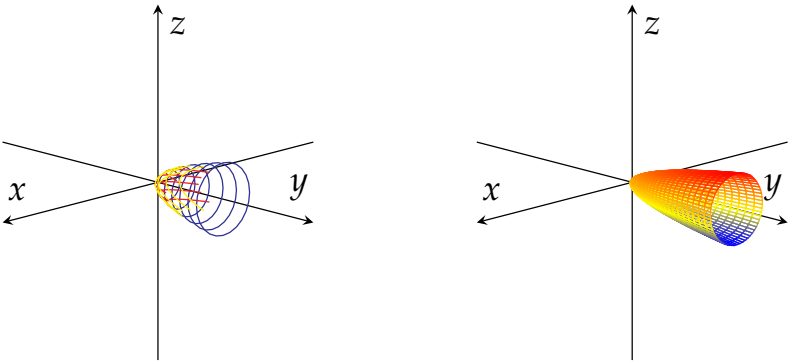
Such graphs are known as **elliptic cone**. Continuing, let's try and graph surfaces of the following form with $a, b > 0$.

$$y = \frac{x^2}{a^2} + \frac{z^2}{b^2}$$

First, consider the trace in $y = k$. Note that if $k < 0$, then there are no points since right-hand side is sum of squares. Moreover rearranging the equations, we obtain the following equation for ellipses.

$$\frac{x^2}{ka^2} + \frac{z^2}{kb^2} = 1$$

Continuing, we can consider traces in $x = k$ and $z = k$. For each case, we can easily notice that it is the locus of quadratic functions of y with respect to x and z. Graphing, we obtain the general sketch.



This is known as **elliptic paraboloid**. There are much more surfaces to discuss. For instance, **hyperboloid of two sheets** are represented by $\frac{z^2}{c^2} - \frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$ while **hyperbolic paraboloid** is described by $\frac{z}{c} = \frac{x^2}{a^2} - \frac{y^2}{b^2}$. Moreover, **ellipsoid** are represented by $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$. For all such surfaces, the following fact holds.

> Given a surface S with equation $E(x, y, z)$, the equation of the surface S', obtained by translating S by a, b, c units in the x, y, and z-direction respectively, is $E(x - a, y - b, z - c)$.

This is pretty straightforward with the same intuition we use for 2-space. This is it for this section! With different surfaces in mind, let's discuss different ways to represent coordinates in 3-space.

§18.4 Cylindrical and Spherical Coordinates

Recall that in 2-space, we have different ways to represent a point. Using Cartesian coordinates, we represent a point P as (x, y). In polar coordinate system, we write the point as (r, θ) where r is the radius, or distance from the origin, and θ is the angle from the positive x-axis. In 3-space, we can represent a coordinate in a very similar way.

18.4.1 Cylindrical Coordinates

As always, let's get started with the definition.

Definition 18.4.1

Given a point P in 3-space with rectangular coordinate (x, y, z), the **cylindrical coordinates** of point P is (r, θ, z) if (r, θ) is the polar coordinate of (x, y).

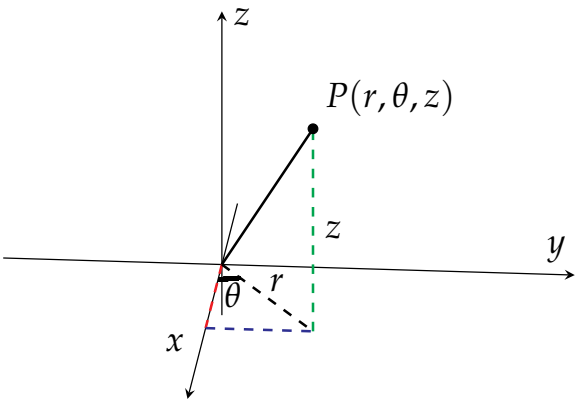
Just like the polar coordinates, we can see that the rectangular coordinate of P with cylindrical coordinates (r, θ, z) adhere to the following equations.

$$x = r\cos\theta, \quad y = r\sin\theta, \quad z = z$$

Similarly, given point P with rectangular coordinate (x, y, z), we can see that the cylindrical coordinate adhere to the following.

$$r = \sqrt{x^2 + y^2}, \quad \tan\theta = \frac{y}{x}, \quad z = z$$

However, it is important to note that $\tan\theta = \frac{y}{x}$ may not hold depending on the position of the point. It is important to see which quadrant the point lies in first before using it.

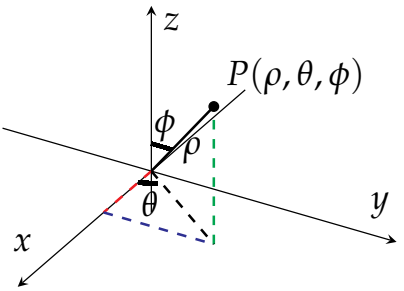


Sometimes being able to represent coordinates in cylindrical coordinates helps writing surfaces in a more simple way. Consider the surface defined by $z = \sqrt{9x^2 + 9y^2}$. This surface is quite cumbersome to write. However using $r = x^2 + y^2$, we can see that $z = 3r$. Therefore the surface can be represented as $r = \frac{z}{3}$, which in my opinion is way easier to

write. Continuing, let's discuss more on the second way of representing: spherical coordinates.

18.4.2 Spherical Coordinates

Another extension of polar coordinates for 3-space besides cylindrical coordinate is spherical coordinate. To understand the coordinates, consider the following diagram.



Using the figure above, we can define the spherical coordinate of a point P.

Definition 18.4.2

Given a point P with $\rho \geq 0$, $0 \leq \theta < 2\pi$, and $0 \leq \phi \leq \pi$, the **spherical coordinate** of P is (ρ, θ, ϕ).

Using this definition, let's take a look at how the spherical coordinates can be converted to cylindrical and rectangular coordinates. From the diagram, it is evident that the following equations hold.

$$r = \rho \sin\phi, \quad x = \rho \sin\phi \cos\theta, \quad y = \rho \sin\phi \sin\theta, \quad z = \rho \cos\phi$$

Similarly, the following equations holds for converting rectangular coordinates to spherical coordinates.

$$\rho = \sqrt{x^2 + y^2 + z^2}, \quad \tan\theta = \frac{y}{x}, \quad \phi = \cos^{-1}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right)$$

Just like the cylindrical coordinates, spherical coordinates can help in representing certain surfaces. Consider the following example.

Exercise 18.4.3

Rewrite the hyperboloid $x^2 + y^2 - z^2 = 1$ using spherical coordinates.

Solution.

First, notice that $\rho = \sqrt{x^2 + y^2 + z^2}$ and $z = \rho \cos \phi$. Substituting, $\rho^2 - 2\rho^2 \cos^2 \phi = 1$ is obtained. Therefore, the hyperboloid can be written as $\rho^2 = \frac{1}{1 - 2\cos^2 \phi} = -\sec 2\phi$.

This is it for different coordinate representations of a point in 3-space! From our next notes, we will begin discussing about different functions and derivatives.

Note 19

Vector-Valued Functions and Parametrization

In our previous note, we reviewed about vectors and their geometry. For calculus in higher dimension, we also study vector-valued functions in addition to real-valued ones that we discussed for calculus in 2-space. In this note, we will further extend our discussion on vectors with vector-valued functions, parametric curves, and calculus in higher dimension.

§19.1 Vector-Valued Functions and Calculus

As always, let's get started with the definition.

Definition 19.1.1

A function with real inputs is a **vector-valued function** if its outputs are vectors. The **implicit domain** is the largest set of reals for which the function is well-defined when the domain of a vector-valued function is not explicitly stated.

Below are some examples of vector-valued functions.

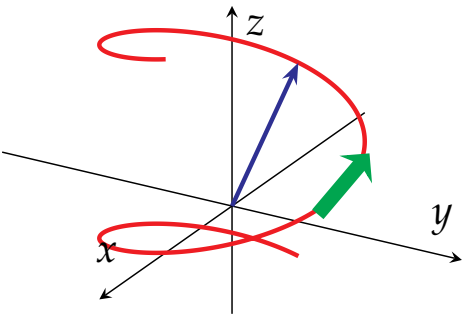
$$\mathbf{F}(t) = \langle t+2, 3t^2, 4t^3 \rangle, \quad -10 \leq t \leq 10$$
$$\mathbf{r}(t) = t\mathbf{i} + 5t\mathbf{j} + \cos t\mathbf{k}, \quad -2\pi \leq t \leq 10\pi$$

Note that the range need not be vectors in 3-space, but it could be any vectors. From the example above, we could define another term.

Definition 19.1.2

Consider a vector-valued function $\mathbf{r} = f_1(t)\mathbf{i} + f_2(t)\mathbf{j} + f_3(t)\mathbf{k}$. The functions f_1, f_2, f_3 of t are known as the **component functions** of $\mathbf{r}$.

Graphing such functions is pretty intuitive. For each input t, we would draw the resulting vector from the origin, which is called the **radius vector** or **position vector**. Below is an example of graphing $\mathbf{r} = \cos t\mathbf{i} + \sin t\mathbf{j} + t\mathbf{k}$ for $t \in [-\pi, 2\pi]$.



As you can see from the figure above, the blue arrow from the origin is a radius vector of the helix. The arrowhead on the curve is known as the **orientation** of the graph, which represents the direction in which the curve traverse as the input increase.

One more important thing to note is that the parametrization of a curve is *not* unique. For instance, the curve above $\mathbf{r} = \cos t\mathbf{i} + \sin t\mathbf{j} + t\mathbf{k}$ for $t \in [-\pi, 2\pi]$ can also be represented as $\mathbf{s} = \cos 2t\mathbf{i} +$

$\sin 2t\mathbf{j} + 2t\mathbf{k}$ for $t \in [-\frac{\pi}{2}, \pi]$.

Extending our example, we can see that the curve also lies on an circular curve since for $x(t) = \cos t$ and $y(t) = \sin t$, $x^2 + y^2 = 1$.

19.1.1 Calculus for Vector-Valued Functions

Now that we discussed what vector-valued functions are and how to graph them, it is natural to see how calculus work for such functions. Just like graphing, it is quite natural to see how calculus works given the properties of limits, derivatives, and integrals.

Given a vector-valued function $\mathbf{r}(t) = f_1(t)\mathbf{i} + f_2(t)\mathbf{j} + f_3(t)\mathbf{k}$, its limits, derivative, and integrals are defined as the following.

- $\lim\limits_{t\to a} \mathbf{r}(t) = \left(\lim\limits_{t\to a} f_1(t)\right)\mathbf{i} + \left(\lim\limits_{t\to a} f_2(t)\right)\mathbf{j} + \left(\lim\limits_{t\to a} f_3(t)\right)\mathbf{k}$
- $\mathbf{r}'(t) = f_1'(t)\mathbf{i} + f_2'(t)\mathbf{j} + f_3'(t)\mathbf{k}$
- $\int \mathbf{r}(t)\,dt = \int f_1(t)\,dt\,\mathbf{i} + \int f_2(t)\,dt\,\mathbf{j} + \int f_3(t)\,dt\,\mathbf{k}$
- $\int_a^b \mathbf{r}(t)\,dt = \int_a^b f_1(t)\,dt\,\mathbf{i} + \int_a^b f_2(t)\,dt\,\mathbf{j} + \int_a^b f_3(t)\,dt\,\mathbf{k}$

Continuing with the definitions, it is natural to notice that the limit, derivatives, and integrals exist if and only if they exist for the component functions. In other words, $\mathbf{r}$ is continuous at a if and only if its component functions are continuous at a.

Using the definition and properties of derivatives and integrals for real-valued functions, it is self-evident that the following rules also hold for vector-valued functions $\mathbf{r}$ under the appropriate conditions.

- $(k\mathbf{r})' = k\mathbf{r}'$
- $(\mathbf{r_1} \pm \mathbf{r_2})' = \mathbf{r_1}' \pm \mathbf{r_2}'$
- For real-valued function f, $(f \cdot \mathbf{r})' = f' \cdot \mathbf{r} + f \cdot \mathbf{r}'$

- $\int k\mathbf{r}(t)\,dt = k\int \mathbf{r}(t)\,dt$
- $\int \mathbf{r_1}(t) \pm \mathbf{r_2}(t)\,dt = \int \mathbf{r_1}(t)\,dt \pm \int \mathbf{r_2}(t)\,dt$
- For $\mathbf{R}'(t) = \mathbf{r}(t)$ satisfying for $t \in [a,b]$, $\int_a^b \mathbf{r}(t)\,dt = \mathbf{R}(b) - \mathbf{R}(a)$

Before concluding this section of the note, let's discuss how we could geometrically interpret derivatives of vector-valued functions.

First, consider the following equations.

$$\begin{aligned}
&\mathbf{r}'(t)\\
&= f_1'(t)\mathbf{i} + f_2'(t)\mathbf{j} + f_3'(t)\mathbf{k}\\
&= \left(\lim_{h\to 0}\frac{f_1(t+h)-f_1(t)}{h}\right)\mathbf{i} + \left(\lim_{h\to 0}\frac{f_2(t+h)-f_2(t)}{h}\right)\mathbf{j}\\
&\qquad\qquad + \left(\lim_{h\to 0}\frac{f_3(t+h)-f_3(t)}{h}\right)\mathbf{k}\\
&= \lim_{h\to 0}\frac{f_1(t+h)\mathbf{i} + f_2(t+h)\mathbf{j} + f_3(t+h)\mathbf{k} - (f_1(t)\mathbf{i} + f_2(t)\mathbf{j} + f_3(t)\mathbf{k})}{h}\\
&= \lim_{h\to 0}\frac{\mathbf{r}(t+h)-\mathbf{r}(t)}{h}
\end{aligned}$$

From our equation, it is evident that $\mathbf{r}'(t)$ is getting more parallel with line tangent to the curve at the terminal point of $\mathbf{r}(t)$ as $h \to 0$. In other words, $\mathbf{r}'(t)$ is a vector that is tangent to the curve at $r(t)$ for some value t. Such vectors are called **tangent vector** and the line that passes through tangent vectors is known as **tangent line**. Let's take a look at an example.

Exercise 19.1.3

Given $\mathbf{r}(t) = \langle \frac{t^2}{2}, \sin t, e^t \rangle$, find the tangent vector and tangent line at $t = \pi$.

Solution.

First, notice that $\mathbf{r}'(t) = \langle t, \cos t, e^t \rangle$. Therefore, $\mathbf{r}'(\pi) = \langle \pi, -1, e^\pi \rangle$, which is the tangent vector.

Continuing, note that $\mathbf{r}(\pi) = \langle \frac{\pi^2}{2}, 0, e^\pi \rangle$. Therefore, the tangent line can be represented as $L(t) = \langle \frac{\pi^2}{2}, 0, e^\pi \rangle + t\langle \pi, -1, e^\pi \rangle$.

Before concluding this section of the note, consider the following theorem.

Theorem 19.1.4

Given vector-valued functions $\mathbf{r_1}$ and $\mathbf{r_2}$, the following equations hold.

- $(\mathbf{r_1} \cdot \mathbf{r_2})' = \mathbf{r_1}' \cdot \mathbf{r_2} + \mathbf{r_1} \cdot \mathbf{r_2}'$
- $(\mathbf{r_1} \times \mathbf{r_2})' = \mathbf{r_1}' \times \mathbf{r_2} + \mathbf{r_1} \times \mathbf{r_2}'$

Proof.

For the first equation, it suffices to show for vectors in 3-space without loss of generality. Define $\mathbf{r_1} = \langle f_1, g_1, h_1 \rangle$ and $\mathbf{r_2} = \langle f_2, g_2, h_2 \rangle$. By definition, $\mathbf{r_1} \cdot \mathbf{r_2} = f_1 f_2 + g_1 g_2 + h_1 h_2$. Therefore, $(\mathbf{r_1} \cdot \mathbf{r_2})' = (f_1' f_2 + g_1' g_2 + h_1' h_2) + (f_1 f_2' + g_1 g_2' + h_1 h_2')$, which is equal to $\langle f_1', g_1', h_1' \rangle \cdot \langle f_2, g_2, h_2 \rangle + \langle f_1, g_1, h_1 \rangle \cdot \langle f_2', g_2', h_2' \rangle$. Thus, the first equation holds.

Note that the second equation holds for $\mathbf{r_1}$ and $\mathbf{r_2}$ in 3-space. Therefore, define $\mathbf{r_1} = \langle f_1, g_1, h_1 \rangle$ and $\mathbf{r_2} = \langle f_2, g_2, h_2 \rangle$. Consider the following equations.

$$\begin{aligned}
\mathbf{r_1} \times \mathbf{r_2} &= \langle g_1 h_2 - h_1 g_2, h_1 f_2 - f_1 h_2, f_1 g_2 - g_1 f_2 \rangle \\
(\mathbf{r_1} \times \mathbf{r_2})' &= \langle g_1' h_2 - h_1' g_2 + g_1 h_2' - h_1 g_2', h_1' f_2 - f_1' h_2 \\
&\qquad + h_1 f_2' - f_1 h_2', f_1' g_2 - g_1' f_2 + f_1 g_2' - g_1 f_2' \rangle \\
&= \langle g_1' h_2 - h_1' g_2, h_1' f_2 - f_1' h_2, f_1' g_2 - g_1' f_2 \rangle \\
&\qquad + \langle g_1 h_2' - h_1 g_2', h_1 f_2' - f_1 h_2', f_1 g_2' - g_1 f_2' \rangle
\end{aligned}$$

$$= \mathbf{r_1}' \times \mathbf{r_2} + \mathbf{r_1} \times \mathbf{r_2}'$$

Thus, the theorem holds.

One fun fact that we can notice from the first equation is that if we assume $\mathbf{r}(t) \cdot \mathbf{r}(t) = a$ for some constant a, then $\mathbf{r}'(t) \cdot \mathbf{r}(t) + \mathbf{r}(t) \cdot \mathbf{r}'(t) = 0$ and $\mathbf{r}(t) \cdot \mathbf{r}'(t) = 0$. In other words, $\mathbf{r}(t)$ and $\mathbf{r}'(t)$ are orthogonal! This is it for this section, and let's discuss more on parametrization in the next section.

§19.2 Parameterizations of Curves

As you could have noticed from the previous section, there are many different ways to represent a curve. In this section, we will discuss another method that uses the length of arc of a curve. But before doing that, let's get started with definitions.

Definition 19.2.1

A **parametrization** of a curve is a way of representing the curve in a continuous vector-valued function. A parametrization is **smooth** if its derivative is nonzero and continuous at the domain of the curve.

Notice that a smooth parametrization of a curve is not unique because we can always scale the parameter with accordingly adjusted domain or construct with different orientations. Continuing from the definition, we can establish an important theorem.

Theorem 19.2.2

Given a curve C, let $\mathbf{r_1}$ be a smooth parametrization of C with the domain U. If s is a differentiable real-valued function with domain V and range U such that s' is continuous and nonzero, then a smooth parametrization of C can also be represented as $\mathbf{r_2} = \mathbf{r_1} \circ s$.

Proof.

Without loss of generality, assume $\mathbf{r_1}$ is a parameterization in 3-space defined as $\mathbf{r_1}(t) = \langle x(t), y(t), z(t) \rangle$. Because it is self-evident that $\mathbf{r_2}$ is a parametrization of C by definition, it suffices to show that $\mathbf{r_2}$ is smooth. Consider the following equations.

$$\begin{aligned}
\mathbf{r_2}'(k) &= \frac{d}{dk}\langle x(s(k)), y(s(k)), z(s(k)) \rangle \\
&= s'(k)\langle x'(s(k)), y'(s(k)), z'(s(k)) \rangle \\
&= s'(k)\mathbf{r_1}'(s(k))
\end{aligned}$$

Notice that because s' and $\mathbf{r_1}'$ are both nonzero and continuous function by construction, $\mathbf{r_2} = \mathbf{r_1} \circ s$ is a smooth parametrization of C.

We actually have a special name for s. Such functions are known as a **smooth change in parameter**. With this concept in mind, we can find a special way of parametrizing a curve.

Before continuing with the new method, we need to understand how arc lengths are defined in a curve.

For a curve defined as $\mathbf{r}(t) = \langle x(t), y(t), z(t) \rangle$, the **arc length** s of the curve in the interval $a \leq t \leq b$ is defined as the following.

$$s = \int_a^b \sqrt{x'(t)^2 + y'(t)^2 + z'(t)^2}\, dt = \int_a^b \|\mathbf{r}'(t)\|\, dt$$

A way of understanding this equation is cutting the curve in the given interval into arbitrarily small pieces so that each piece becomes a linear segment. Then, it becomes obvious that we can simply use Pythagorean theorem to derive the formula just as we do for 2-space. Also, using the definition of a norm of a vector, we can rewrite the integral.

Now with this idea in mind, we can construct arc length parametrization of a curve. The basic idea of this method is to define each point on the curve with the distance from a reference point $t_0 = f(0)$. Consider the

following equation.

$$s = \int_{t_0}^{f(s)} \|\mathbf{r_1}'(t)\| \, dt$$

From here it is evident that we can define a new t-value with some distance s since there are only two possible directions depending on the position of $f(s)$. Another fun fact that we can notice by letting $s = f^{-1}(u)$ for some independent values of u is that

$$f^{-1}(u) = \int_{t_0}^{f(f^{-1}(u))} \|\mathbf{r_1}'(t)\| \, dt = \int_{t_0}^{u} \|\mathbf{r_1}'(t)\| \, dt$$
$$f^{-1}(t) = \int_{t_0}^{t} \|\mathbf{r_1}'(u)\| \, du$$

holds and the following equation is also true by FTC.

$$\left(f^{-1}\right)'(t) = \|\mathbf{r_1}'(t)\|$$

This shows that because f is differentiable and f' is continuous and nonzero from the equation $f'(s) = \frac{1}{f^{-1\prime}(f(s))}$, f is a smooth change of parameter.

> In other words, the arc length function of a curve with parametrization $\mathbf{r}$ can be represented as the following with the reference point t_0.
>
> $$f(t) = \int_{t_0}^{t} \|\mathbf{r}'(u)\| \, du$$

This is how we represent a curve with arc length! Before we conclude this section, here is an important theorem to consider.

Theorem 19.2.3

Given an arc length parametrization $\mathbf{r}(s)$ of a curve, $\|\mathbf{r}'(s)\| = 1$ always holds.

Proof.

Note that by definition,

$$s = \int_0^s \|\mathbf{r}'(u)\|\, du$$

and we obtain $1 = \|\mathbf{r}'(s)\|$ by differentiating with respect to s.

This is it for this section and this note! In our next section, we will discuss about curvature.

§19.3 Curvature

Before we get into curvature, we need to discuss some important topics and notation that will be used throughout the notes. As always, let's get started with definition.

Definition 19.3.1

Given a vector-valued function $\mathbf{r}$ of a smooth curve C, the **unit tangent vector** to C is defined as the following.

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{\|\mathbf{r}'(t)\|}$$

As the name suggests, $\|\mathbf{T}(T)\| = \frac{\|\mathbf{r}'(t)\|}{\|\mathbf{r}'(t)\|} = 1$. Also, it is worth noting that the unit tangent vector at a point of all smooth parametrizations of a curve are the same because the tangent vectors are normalized. Continuing, we can define another term.

Definition 19.3.2

Given a smooth parametrization $\mathbf{r}$ of a curve and the unit tangent vector $\mathbf{T}(t)$, the **unit normal vector** to the curve at t is defined as the following.

$$\mathbf{N}(t) = \frac{\mathbf{T}'(t)}{\|\mathbf{T}'(t)\|}$$

Notice that because $\mathbf{T}'(t) \equiv 0$ if a curve is a straight line, $\mathbf{N}(t)$ is not

defined. Moreover, notice that because $\mathbf{T}(t) \cdot \mathbf{T}(t) = 1$ and $2\mathbf{T}'(t) \cdot \mathbf{T}(t) = 0$, $\mathbf{N}(t)$ is orthogonal to $\mathbf{T}(t)$ as the name suggests! However, there are different ways for $\mathbf{N}$ to be orthogonal to $\mathbf{T}$ geometrically. For instance, in 2-space, we have two directions and we have infinitely many possibilities in 3-space. To clarify the direction in which $\mathbf{N}$ is orthogonal in 2-space, consider the following theorem.

Theorem 19.3.3

Given a smooth curve C in 2-space, the unit normal vector at a point on the curve is towards the concave side.

Proof.

Notice that to show that $\mathbf{N}(t)$ point towards the concave side of the curve, it suffices to show that the curve bends towards the direction of $\mathbf{T}'(t)$. Let $\theta(t)$ be the angle of $\mathbf{T}$ from the positive x-axis. Notice that because $\mathbf{T}$ is a unit vector, $\mathbf{T}$ can be written as $\mathbf{T} = (\cos\theta(t), \sin\theta(t))$.

Continuing, $\mathbf{T}'(t) = \theta'(t)(-\sin\theta(t), \cos\theta(t))$, which can be obtained by rotating $\mathbf{T}$ by 90° and multiplying by $\theta'(t)$. Note that if $\theta'(t) > 0$, then it is pointing towards the left, or towards the concave side, of the curve. Similarly, if $\theta'(t)$, then $\mathbf{T}'(t)$ is pointing towards the left, or towards the concave side, of the curve. Because the normal vector is not defined when $\theta'(t) = 0$, the theorem holds.

Before we get into curvature, here is one last definition.

Definition 19.3.4

Given a smooth curve in 3-space, the **binormal vector** to the curve at t is defined as $\mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t)$.

From the definition, we can infer that $\mathbf{B}$ is orthogonal to both unit tangent vector and unit normal vector, and is also a unit vector. The coordinate system using $\mathbf{N}$, $\mathbf{T}$, and $\mathbf{B}$ is known as **Frenet frame** or **TNB-frame**.

One more thing that we can notice from the definition is that $\mathbf{B}(t)$ can be written as the following at t and s if the parametrization is $\mathbf{r}(s)$.

$$\mathbf{B}(t) = \frac{\mathbf{r}'(t) \times \mathbf{r}''(t)}{\|\mathbf{r}'(t) \times \mathbf{r}''(t)\|}$$
$$\mathbf{B}(s) = \frac{\mathbf{r}'(s) \times \mathbf{r}''(s)}{\|\mathbf{r}'(s) \times \mathbf{r}''(s)\|} = \frac{\mathbf{r}'(s) \times \mathbf{r}''(s)}{\|\mathbf{r}''(s)\|}$$

Because the derivation is fairly straightforward with definition and $\mathbf{r}' \times \mathbf{r}' = 0$, let's skip the proof. Now let's discuss about curvature!

Just for the sake of intuition, consider a circle and a square root function. From those two, the circle has larger curvature because it curves more, and we can represent that by the rate of changes in the direction of the tangent vector. Here is more rigorous definition.

Definition 19.3.5

Given an arc length parametrization of a curve, the **curvature** of the curve at s is defined as $\kappa(s) = \|\mathbf{T}'(s)\|$.

In other words, curvature is the rate of change of the tangent vector with respect to its arc length. Notice that $\kappa(s)$ is a function in terms of s.

If the parametrization is not arc-length, then we write it as $\kappa(t = t_0)$ for clarity and the following equation holds.

$$\kappa(t = t_0) = \frac{\|\mathbf{T_1}'(t_0)\|}{\|\mathbf{r_1}'(t_0)\|}$$

To see why this is true, consider the arc-length parametrization $\mathbf{r_2}(s) = \mathbf{r_1}(g(s))$. Similarly, let $\mathbf{T_1}$ and $\mathbf{T_2}$ be the respective unit tangent vectors. Then, the following equations hold.

$$\kappa(t = t_0) = \kappa(s = s_0) = \|\mathbf{T_2}'(s_0)\| = \|\mathbf{T_1}'(g(s_0)) \cdot g'(s_0)\|$$
$$= \frac{\|\mathbf{T_1}'(g(s_0)) \cdot g'(s_0)\|}{\|\mathbf{r_2}'(s_0)\|} = \frac{\mathbf{T_1}'(g(s_0))}{\mathbf{r_1}'(g(s_0))}$$

Therefore, the equation holds.

Continuing, here is another interesting theorem. This is quite important because many times curvature is defined and used with this theorem, or formula, below.

Theorem 19.3.6

Let $\mathbf{r}(t)$ be a smooth parametrization of a curve in 3-space. If $\mathbf{r}''(t_0)$ is defined, then the following equation holds.

$$\kappa(t = t_0) = \frac{\|\mathbf{r}'(t_0) \times \mathbf{r}''(t_0)\|}{\|\mathbf{r}'(t_0)\|^3}$$

Proof.

Define $\mathbf{r_1}(s) = \mathbf{r}(g(s))$ where g is an arc length change. Then, it suffices to show that $\|\mathbf{r_1}''(s_0)\| = \frac{\|\mathbf{r}'(t_0)\times\mathbf{r}''(t_0)\|}{\|\mathbf{r}'(t_0)\|^3}$.

Notice that by construction, the following equations hold.

$$\begin{aligned}\mathbf{r_1}'(s) &= \mathbf{r}'(g(s))g'(s)\\ \mathbf{r_1}''(s) &= \mathbf{r}''(g(s))g'(s)^2 + \mathbf{r}'(g(s))g''(s)\end{aligned}$$

Therefore, the following equations hold.

$$\begin{aligned}\frac{\|\mathbf{r}'(t_0) \times \mathbf{r}''(t_0)\|}{\|\mathbf{r}'(t_0)\|^3} &= \frac{\|\mathbf{r}'(g(s_0)) \times \mathbf{r}''(g(s_0))\|}{\|\mathbf{r}'(g(s_0))\|^3}\\ &= \frac{\|\mathbf{r}'(g(s_0)) \times \mathbf{r}''(g(s_0))\||g'(s_0)|^3}{\|\mathbf{r}'(g(s_0))g'(s_0)\|^3}\\ &= \|\mathbf{r_1}'(s_0) \times \mathbf{r_1}''(s_0)\|\\ &= \|\mathbf{r_1}''(s_0)\|\end{aligned}$$

Therefore, the equation holds.

Notice that if $\mathbf{r}(t) = \langle x(t), y(t), 0\rangle$, then this theorem also holds for 2-space. Let's take a look at an example for computing curvature at a points of a curve.

Exercise 19.3.7

Find the curvature of the helix $\mathbf{r}(t) = \langle a\cos t, a\sin t, bt\rangle$ at t given that $a > 0$.

Solution.

First, recall that $\kappa(t_0) = \frac{\|\mathbf{r}'(t_0)\times\mathbf{r}''(t_0)\|}{\|\mathbf{r}'(t_0)\|^3}$. Moreover, the following equations hold.

$$\begin{aligned}\mathbf{r}'(t) &= \langle -a\sin t, a\cos t, b\rangle\\ \mathbf{r}''(t) &= \langle -a\cos t, -a\sin t, b\rangle\\ \mathbf{r}'(t)\times\mathbf{r}''(t) &= \langle ab\sin t, -ab\cos t, a^2\rangle\end{aligned}$$

Therefore, $\|\mathbf{r}'(t)\times\mathbf{r}''(t)\| = a\sqrt{a^2+b^2}$ and $\|\mathbf{r}'(t)\| = \sqrt{a^2+b^2}$. In other words, $\kappa(t_0) = \frac{a}{a^2+b^2}$.

In this case, we could see that the curvature is constant because it is a circular helix. The similar idea apply for 2-space.

Exercise 19.3.8

Given a circle $\mathbf{r}(t) = \langle a\cos t, a\sin t\rangle$ with $a > 0$, find the curvature at each point on the circle.

Solution.

First, notice that

$$\begin{aligned}\mathbf{r}'(t) &= \langle -a\sin t, a\cos t, 0\rangle\\ \mathbf{r}''(t) &= \langle -a\cos t, -a\sin t, 0\rangle\end{aligned}$$

holds. Therefore, $\kappa(t) = \frac{a^2}{(a^2\sin^2 t + a^2\cos^2 t)^{\frac{3}{2}}} = \frac{1}{a}$.

Again, the curvature is constant! Another interesting property of circle is that its curvature at each point is the reciprocal of its radius. Continuing with this observation, we can establish the following definitions.

Definition 19.3.9

Given a point P on a curve C with curvature κ_0, there exists a unique circle ω that is tangent to the line that is tangent to C at P and is on the concave side of C. Such circle is known as **osculating circle** to C at P where the radius $\frac{1}{\kappa_0}$ and its center are known as the **radius of curvature** and the **center of curvature** of C at P.

From the definition and construction, we can notice that the osculating circle can be used to best approximate the curve at P.

Another interesting property that we can notice from curvature for an arc-length parametrization in 2-space is its relationship with the angle that the unit tangent vector makes with the positive x-axis. Notice that $\mathbf{T}(\theta) = \langle \cos\theta, \sin\theta \rangle$. More $\mathbf{T}(s) = \mathbf{T}(\theta(s)) = \langle \cos\theta(s), \sin\theta(s) \rangle$. Because $\|\frac{d\mathbf{T}}{d\theta}\| = 1$, $\kappa(s) = \|\frac{d\mathbf{T}}{ds}\| = \|\frac{d\mathbf{T}}{d\theta} \cdot \frac{d\theta}{ds}\| = |\frac{d\theta}{ds}|$.

> Therefore, $\kappa(s) = |\frac{d\theta}{ds}|$ if θ is the angle that the unit tangent vector makes with the positive x-axis given an arc-length parameterization of a curve.

Before we wrap up this note, here are some interesting connections we can make with curvature and physics.

First, notice that $\mathbf{v}(t) = \frac{ds}{dt}\mathbf{T}(t)$ given an arc-length parameterization. Therefore, the following equations hold.

$$\begin{aligned}\mathbf{a}(t) &= \frac{d^2s}{dt^2}\mathbf{T}(t) + \frac{ds}{dt}\mathbf{T}'(t) = \frac{d^2s}{dt^2}\mathbf{T}(t) + \|\mathbf{r}'(t)\|\mathbf{T}'(t) \\ &= \frac{d^2s}{dt^2}\mathbf{T}(t) + \|\mathbf{r}'(t)\|\mathbf{T}'(t) = \frac{d^2s}{dt^2}\mathbf{T}(t) + \frac{\|\mathbf{T}'(t)\|}{\|\mathbf{r}'(t)\|} \cdot \|\mathbf{r}'(t)\|^2 \cdot \frac{\mathbf{T}'(t)}{\|\mathbf{T}'(t)\|} \\ &= \frac{d^2s}{dt^2}\mathbf{T}(t) + \kappa(t)\|\mathbf{r}'(t)\|^2\mathbf{N}(t)\end{aligned}$$

Therefore, we can conclude the following.

> Given an arc-length parametrization of position of a particle, its

acceleration at t can be represented as the following.

$$\mathbf{a}(t) = \frac{d^2s}{dt^2}\mathbf{T}(t) + \kappa(t)\|\frac{ds}{dt}\|^2\mathbf{N}(t)$$

We can also notice that this equation is consistent with centripetal force and acceleration in linear path! Continuing from this equation, we can establish the following definition.

Definition 19.3.10

Let $a_T(t) = \frac{d^2s}{dt^2}$ and $a_N(t) = \kappa(t)\|\frac{ds}{dt}\|^2$. Then $\mathbf{a}(t) = a_T\mathbf{T} + a_N\mathbf{N}$. The terms a_T and a_N are known as the **tangential component of acceleration** and the **normal component of acceleration**. Moreover, $a_T\mathbf{T}$ and $a_N\mathbf{N}$ are known as the **tangential vector component of acceleration** and the **normal vector component of acceleration** respectively.

Continuing from this notation and definition, we can notice the following.

Given a smooth parametrization of the position of a particle, the following equations hold.

$$a_T = \frac{\mathbf{v}\cdot\mathbf{a}}{\|\mathbf{v}\|}$$
$$a_N = \frac{\mathbf{v}\times\mathbf{a}}{\|\mathbf{v}\|}$$
$$\kappa = \frac{\mathbf{v}\times\mathbf{a}}{\|\mathbf{v}\|^3}$$
$$\|\mathbf{a}\|^2 = a_T^2 + a_N^2$$

The derivation of these equations are fairly straightforward, so let's skip

the proof. Finally, consider the following equations for projectile motion.

$$\begin{aligned}\mathbf{a} &= -g\mathbf{j}\\ \mathbf{v} &= -gt\mathbf{j} + \mathbf{v_0}\\ \mathbf{r} &= \left(-\frac{gt^2}{2} + \mathbf{s_0}\right)\mathbf{j} + t\mathbf{v_0}\end{aligned}$$

Substituting $\mathbf{v_0} = \langle x, y\rangle$, we can see that the following equation holds.

$$y = \left(-\frac{g}{2a^2}\right)x^2 + \frac{b}{a}x + \mathbf{s_0}$$

Indeed projectile motions are parabolas, and this is it for this note! In the next note, we will discuss about multivariable functions.

Part IV

Machine Learning

Contents

What is AI and Why Should We Learn It? **399**

20 Fundamental Math for Machine Learning **401**
20.1 Calculus 401
20.1.1 Foundations 402
20.2 Linear Algebra 406
20.2.1 Foundations 406
20.2.2 CONTINUE 408
20.3 Probability Theory 408

21 Python Basics for Machine Learning **413**
21.1 Basic Syntax 413
21.2 Modules and Libraries 419
21.2.1 NumPy 419
21.2.2 Matplotlib 422

22 Foundations and Intuitions **425**
22.1 Fundamental Terms and Concepts 425
22.1.1 ML Paradigms 426
22.2 Introduction to LLMs 429
22.2.1 Tokenization 430
22.2.2 Language Model Output 431
22.3 Improving Large Language Models 433

22.3.1 Introduction to Post-Training 433
22.3.2 Reasoning 434

23 Deep Dive into Deep Neural Networks 437
23.1 Definitions and High Level Overview 437
23.1.1 High Level Architecture 437
23.1.2 Training 440
23.1.3 Gradient Descent and Learning Rate 442
23.2 Mathematical Interpretations 446
23.2.1 Deriving Equations and Backpropagation 448
23.3 Building Vanilla NN From Scratch 452
23.3.1 Simple Stochastic Gradient Descent-Based Learning Engine 452
23.3.2 Vanilla Neural Network 463

24 Building Language Models 477
24.1 Bigram Language Model 477
24.2 Attention and Transformers 477
24.3 Building an LLM with PyTorch 478

25 AI and The Future 479
25.1 AI Safety 479

26 Further Resources 481

What is AI and Why Should We Learn It?

Unless you've been living under a rock, you probably have heard of the term "AI" and "Machine Learning". Just as the name suggests Artificial Intelligence is any technology that mimics human intelligence. That being said, AI is a very, very broad term. Under AI, there is machine learning (ML), which quite literally means techniques, or machines, that can learn from data to better mimic human intelligence. Now, just as us humans learn from data with our brain, a subset of ML is Deep Learning (DL), which uses neural networks inspired from our brains. We could continue more, but let's save that for later since this is a brief introduction.

AI has been studied, and will continue to develop. In October 2015, a legendary match between an AI model AlphaGo developed by DeepMind and one of the greatest Go player Lee Sedol was conducted. During the game, AlphaGo demonstrated its creativity through numerous moves including Move 37. Ultimately, AlphaGo won the game with 4-1 victory. More information can be found in AlphaGo.

Despite the long history, the modern AI boom in the public occurred from late 2022 when OpenAI released its GPT-3.5. Nevertheless, LLMs

have significantly transformed the lives of countless people. For instance, as the concept of "vibe coding" emerged, the concept of "coding" became more accessible to the public. Moreover, job markets are transforming with the rise of AI. Now despite all "AI bubbles" concerns and anti-AI movements, it is somewhat self-evident that AI and the concept of AGI (Artificial General Intelligence) will not suddenly disappear in near future. Then, the real question is "do you want to become a consumer or producer?"

To briefly share why I began studying ML, the reason is quite simple. As a student who wishes to advance as a quantitative researcher, not learning ML is just not in the option. Moreover, learning to develop AI will widen the future opportunities even if I decided different path than quantitative finance. Although the ideas on an AI model developing another model is rising, it is imperative that a human engineer must superintend the architecture and development. Finally, it is fun to learn. I mean what is more intriguing than to learn about your favorite tool and how to contribute to its development?

Note 20

Fundamental Math for Machine Learning

This note covers fundamental math necessary for machine learning. Please note that I won't nor have the capacity to include details on the concepts that are not strictly necessary to learn ML, which I believe are out of the scope of the notes. With that in mind, let's get started!

§20.1 Calculus

Ah yes, that one friend who's everywhere, where ever you go. Broadly speaking, calculus is one of the foundational math for machine learning. For instance, backpropagation, which we will discuss further in latter notes, is fundamentally based on the chain rule. There are actually a lot to cover on calculus and there are some advanced topics. Good news is that you probably don't need calculus concepts outside multivariable calculus (a.k.a. Calculus III) unless you are doing theoretical works, and they are out of my capacity. That being said, I will include most calculus concepts that you will need for general purpose, but concepts that are out of scope of these notes will not be included.

20.1.1 Foundations

I am assuming most of you have some background in calculus, so I will be skipping basic concepts from Calculus AB/BC (a.k.a Calculus I and II) as this section is about calculus for machine learning. Please don't worry if you don't as you probably won't need all concepts from the courses to understand the topics here. Now, let's start by building from terminologies.

Definition 20.1.1

The **limit** of a function is the value that the function approaches as the input gets arbitrarily close to a value.

Below are some examples.

$$\lim_{x \to 3} x^2 = 3^2 = 9, \quad \lim_{x \to \infty} \frac{1}{x} = 0, \quad \lim_{x \to 1} \frac{1}{x-1} \text{ DNE}$$

Definition 20.1.2

Derivative of a function respect to a variable is the measure of the rate of change of the output based on the change of the variable(s). **Gradient** is a vector of partial derivatives. **Differentiation** is the process of finding the derivative.

With equations, derivative and partial derivatives can be written as the following for function $f(x)$ and $f(x_1, \ldots, x_n)$.

$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$\frac{\partial}{\partial x_i} f(a) = \lim_{h \to 0} \frac{f(a_1, \ldots, a_{i-1}, a_i + h, a_{i+1}, \ldots, a_n) - f(a_1, \ldots, a_n)}{h}$$

For gradient ∇f, read as "nabla f", the mathematical expression follows.

$$\nabla f(a) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(a) \\ \vdots \\ \frac{\partial f}{\partial x_n}(a) \end{bmatrix}$$

For instance, the gradient of $f(x, y) = x^2 + 2y$ can be written as $\nabla f = \langle 2x, 2 \rangle$ and the gradient at $(1, 1)$ can be written as $\nabla f(1, 1) = \langle 2, 2 \rangle$.

Now that we discussed about derivatives, let's take a look at common rules that will appear for machine learning. Starting with the power rule, below is the list of common rules.

Theorem 20.1.3: Common Rules for Derivatives

1. $\frac{d}{dx} x^n = nx^{n-1}$
2. $\frac{d}{dx} e^x = e^x$
3. $\frac{d}{dx} \ln(x) = \frac{1}{x}$
4. $\frac{d}{dx} a = 0$
5. $\frac{d}{dx} cf(x) = cf'(x)$
6. $\frac{d}{dx} \left(f(x) \pm g(x) \right) = \frac{d}{dx} f(x) \pm \frac{d}{dx} g(x)$
7. $\frac{d}{dx} \left(f(x) g(x) \right) = f'(x) g(x) + f(x) g'(x)$
8. $\frac{d}{dx} \left(\frac{f(x)}{g(x)} \right) = \frac{f'(x) g(x) - f(x) g'(x)}{\left(g(x) \right)^2}$
9. $\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x} \left(\frac{\partial f}{\partial x} \right)$
10. $\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y} \left(\frac{\partial f}{\partial x} \right)$

Let's skip the proofs for this one! In addition to derivative rules, one of the most primary use case of calculus in machine learning is the chain rule. Indeed, the neural network is fundamentally based on the chain rule in calculus.

Theorem 20.1.4: The Chain Rule of Calculus

The derivative of a composite function $h(x) = f \circ g(x)$ can be represented as $h'(x) = f'(g(x)) \cdot g'(x)$. Alternatively, for $y = f(u)$ and $u = g(x)$, the following equation holds.

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Proof.

Consider the following equation by the definition of derivatives.

$$h'(x) = \lim_{n \to 0} \frac{h(x+n) - h(x)}{n}$$

Substituting $h(x) = f(g(x))$, the following equations are obtained.

$$\begin{aligned} h'(x) &= \lim_{n \to 0} \frac{f(g(x+n)) - f(g(x))}{n} \\ &= \lim_{n \to 0} \frac{f(g(x+n)) - f(g(x))}{g(x+n) - g(x)} \cdot \lim_{n \to 0} \frac{g(x+n) - g(x)}{n} \end{aligned}$$

Let $n' = g(x+n) - g(x)$. Assuming that g is continuous, as $n \to 0$, $n' \to 0$. Therefore, the following equation is obtained.

$$h'(x) = \lim_{n' \to 0} \frac{f(u+n') - f(u)}{n'} \cdot \lim_{n \to 0} \frac{g(x+n) - g(x)}{n} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Thus, the chain rule holds.

For a brief example, let's take a look at an example with sigmoid function.

Exercise 20.1.5: The Derivative of Sigmoid Function

Sigmoid function is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. Find the derivative of $\sigma(x)$.

Solution.

Consider the following process with the product and chain rule.

$$\begin{aligned}\sigma'(x) &= \frac{d}{dx}\left(1+e^{-x}\right)^{-1} = -\left(1+e^{-x}\right)^{-2}\cdot\left(-e^{-x}\right)\\ &= \left(1+e^{-x}\right)^{-2}\cdot e^{-x} = \left(1+e^{-x}\right)^{-1}\cdot\frac{e^{-x}}{1+e^{-x}}\\ &= \sigma(x)\left(1-\sigma(x)\right)\end{aligned}$$

Therefore, the derivative $\sigma'(x) = \sigma(x)(1-\sigma(x))$.

In addition to the chain rule above, we could expand it for multivariable functions.

Theorem 20.1.6: The Chain Rule for One Independent Variable

For differentiable functions $x = a(t)$, $y = b(t)$, and $z = f(x,y)$, the following equation holds.

$$\frac{dz}{dt} = \frac{\partial z}{\partial x}\cdot\frac{dx}{dt} + \frac{\partial z}{\partial y}\cdot\frac{dy}{dt}$$

Similarly, we could expand it for cases where x and y are functions of two variables.

Theorem 20.1.7: The Chain Rule for Two Independent Variable

For differentiable functions $x = a(u,v)$, $y = b(u,v)$, and $z = f(x,y)$, the following equations hold.

$$\begin{aligned}\frac{\partial z}{\partial u} &= \frac{\partial z}{\partial x}\frac{\partial x}{\partial u} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial u}\\ \frac{\partial z}{\partial v} &= \frac{\partial z}{\partial x}\frac{\partial x}{\partial v} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial v}\end{aligned}$$

For the purpose of these notes, let's skip the proof for the two theorems above.

§20.2 Linear Algebra

Linear algebra is a foundational building block of ML as majority of the components are written in very high dimensional arrays and tensors. Without exaggeration, training and using a model can be considered as performing billions of linear algebra operations. Therefore, it is encouraged to get familiar with the notations as they will be used throughout the future notes. That being said, because many of the contents were introduced in detail in the linear algebra part of LibreNotebook, I will keep it brief here for topics that were not introduced in the previous part.

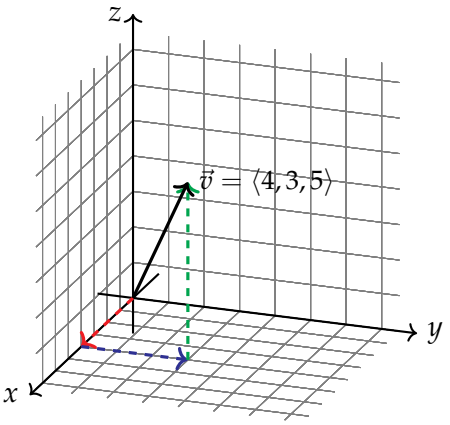
20.2.1 Foundations

Many mathematicians might not like the definition below, but I think it is worth considering different perspectives of how we can see vectors and matrices.

Definition 20.2.1

A 1-dimensional vector is simply called **vector**. A 2D vector is called **matrix**. For vectors with 3+ dimensions, the term **tensor** is used.

Now scalars are not considered vectors because it is only defined with magnitude. Geometrically speaking, the dimensions here can be interpreted as the dimension that we usually say in science. For example, a matrix can be plotted in a 2D space with x and y components. Similarly, a 3D tensor can be drawn in 3-dimensional plane with x, y, and z coordinates. Consider the following example.



Continuing, we can see an operation that vectors can perform that we have not discussed in the notes for linear algebra.

Definition 20.2.2

A **dot product** of two vectors is the sum of products of the corresponding elements.

Dot product will be quite important when we discuss attention and transformers because it can tell us how "related" two vectors are. Indeed, consider the vectors A and B with the angle between the two vectors as θ. The following equation holds.

$$A \cdot B = ||A||||B|| \cos\theta$$

Here, $||V||$ for $V \in \mathbb{R}^n$ represents the magnitude of the vector, i.e. $\sqrt{\sum_{k=1}^{n} V_k^2}$. Let's take a look at a quick example.

$$\begin{aligned} \langle 1, 0 \rangle \cdot \langle 1, \sqrt{3} \rangle &= 1 \cdot 1 + 0 \cdot \sqrt{3} = 1 \\ &= 1 \cdot 2 \cdot \cos 60^\circ = 1 \end{aligned}$$

One thing to keep in mind is that we have a special term for the sign $||V||$.

Definition 20.2.3

The **vector norm** is the non-negative value that represents the magnitude of a vector.

With that in mind, let's move on to a more abstract and advanced topics in linear algebra that are constantly used in machine learning.

20.2.2 CONTINUE

§20.3 Probability Theory

Ultimately, ML models are prediction engines. In other words, it is very crucial to understand fundamental probability theory. Therefore, let's get started with basic terminologies and concepts.

Definition 20.3.1

Probability distribution is a process of describing probability and possible cases.

For example, using functions or tables to represents probability of getting a certain rolling sum after rolling a fair standard die two times is using probability distribution.

Few symbols to keep in mind are is used to represent the following of the probability distribution, and $\propto$ is used to illustrate direct proportionality.

Definition 20.3.2

Sample space, denoted as Ω, is a set of possible outcomes. **Probability measure** is a function that maps such outcome events to probability in $[0, 1]$. **Random Variables** are functions that map the outcomes to certain numbers.

Notice that the sum of all measures in a sample space is 1. Now sometimes, probabilities depend on the previous events.

Definition 20.3.3

If A is the event that occurs after B, the probability of A occurring after B is known as the **posterior probability**, denoted as $P(A \mid B)$. Moreover, $P(A)$ is the **prior** probability.

For instance, if A is the event where you see a snail outside and B is the event of raining, then the probability $P(A \mid B)$ is the probability of you seeing a snail outside given that it is raining. With this in mind, we can observe the following important theorem that will come very handy when we discuss language models.

Theorem 20.3.4: The Chain Rule of Probability

The chain rule of probability states the following equation.

$$P(A, B) = P(A \mid B)P(B)$$

The equation above is read as the probability of A and B is equal to the probability of A given B multiplied by the probability of B. The direct corollary of the chain rule is Bayes' Theorem.

Theorem 20.3.5: Bayes' Theorem

Conditional probabilities can be found with Bayes' Theorem defined with the equation below for $P(B) \neq 0$.

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

As you may have guessed, we can scale the chain rule above with generalization. Consider the following generalized version of the chain rule.

Theorem 20.3.6: Generalized Chain Rule

The chain rule of probability states the following.

$$\begin{aligned} P(x_1, x_2, \ldots, x_n) &= P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2) \\ &\qquad \cdots P(x_n \mid x_1, x_2, \ldots, x_{n-1}) \\ &= \prod_{i=1}^{n} P\left(x_i \,\middle|\, \bigcap_{k=1}^{i-1} x_k\right) \end{aligned}$$

Proof.

Deriving this generalization from the chain rule with two variables is rather simple. We can use the definition multiple times to multiply and telescope. Here is what I mean. (Note that mathematicians like to use the standard set notations while ML engineers likes to use a more "colloquial notation". For the purpose of these notes, let's use the notations that ML engineers use.)

Consider the following equations by the definition of conditional probability.

$$\begin{aligned} P(x_1) &= P(x_1) \\ P(x_2 \mid x_1) &= \frac{P(x_1, x_2)}{P(x_1)} \\ P(x_3 \mid x_1, x_2) &= \frac{P(x_1, x_2, x_3)}{P(x_1, x_2)} \\ &\vdots \\ P(x_n \mid x_1, \ldots, x_{n-1}) &= \frac{P(x_1, \ldots, x_n)}{P(x_1, \ldots, x_{n-1})} \end{aligned}$$

Multiplying the equations, the following equation is obtained.

$$\begin{aligned} &P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)P(x_n \mid x_1, \ldots, x_{n-1}) \\ &\qquad = P(x_1) \cdot \frac{P(x_1, x_2)}{P(x_1)} \cdot \frac{P(x_1, x_2, x_3)}{P(x_1, x_2)} \cdots \frac{P(x_1, \ldots, x_n)}{P(x_1, \ldots, x_{n-1})} \end{aligned}$$

Telescoping the right hand side, we get our final equation.

$$P(x_1, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)P(x_n \mid x_1, \ldots, x_{n-1})$$

Alternatively, we could try going backward from the final equation from the definition, but I found this proof a little more satisfying.

Note 21

Python Basics for Machine Learning

To develop a model represented in abstract math, we need a programming language. If you ask any ML engineer on the language they use the most, I can guarantee you that it's Python (well most likely). The reason is simple: syntax and libraries. Although all heavy computations in ML is mostly done by C++, Python is way easier to write than C++. You don't have to deal with segmentation faults and spend more time adding the missing semicolon than actually debugging. Moreover, Python has popular libraries like NumPy, Matplotlib, and PyTorch that will make our lives easier. Therefore, we will discuss about basic syntax, NumPy, and Matplotlib for ML in this note. Don't worry, we will discuss about PyTorch in latter notes as it requires ML concepts.

§21.1 Basic Syntax

As always, let's get started by being more familiar with the nature of Python. Unlike most other languages, indentation is a vital factor in Python. Two spaces, four spaces, doesn't really matter, let's use four

spaces as it is more common and readable. Here is the Hello World example.

```
1 print("Hello World")
```

```
Hello World
```

Please note that this Jupyter Notebook can be found in `jupyter/` directory in GitHub under the root directory for these notes. Please feel free to download and follow along! The next example is a list of simple operations that we can do with Python.

```
1  import math
2  import numpy as np
3  import matplotlib.pyplot as plt
4  %matplotlib inline
5
6  print(1 + 2)        # Addition
7  print(3 - 9)        # Subtraction
8  print(4 * 3)        # Multiplication
9  print(5 ** 2)       # Power
10 print(math.log(5)) # log(a, base), default base is
       e
11 print(math.exp(2)) # e^2
12 print(math.e)       # Accessing constants
```

```
3
-6
12
25
1.6094379124341003
7.38905609893065
2.718281828459045
```

As you can see in Cell 2, we need module "math" to access constants and functions like e, π, and $\log_a b$.

Definition 21.1.1

We can achieve the same effect with **variables**, or a place with name that stores a value. A **function** is a reusable block of code dedicated for specific tasks.

Let's take a look at an example of how function and variable in addition to the operations above can be used to find the approximation of the derivative of $f(x) = x^2 + 5$ at $x = 3$.

```
def f(x):
    return x**2 + 5

a = 3
h = 0.0001
der = (f(a + h) - f(a))/h
print(der)
```

```
6.000100000012054
```

Let's check if our answer is reasonable. Using the Power Rule, we know that $f'(x) = 2x$. Because $f(3) = 2 \cdot 3 = 6$ and 6.000100000012054 and 6 are somewhat close, our result is reasonable.

Definition 21.1.2

Lists are dynamic and ordered collection of elements of potentially different data types. An **index** is the position of an element in the list. The index of the first item is 0.

Below are examples of what we can do with lists.

```
# A list can be empty
a = []
print(a)

```

```
# We can append a value at the end
a = [1, 'hi', 3]
a.append('bye')
print(a)

# Remove an element
a.remove('hi')
print(a)

# Access an element with index
print(a[2])

# Find the number of elements
print(len(a))
```

```
[]
[1, 'hi', 3, 'bye']
[1, 3, 'bye']
bye
3
```

What do we do if we need to access multiple elements? This is where **slicing** shines. Consider the following example.

```
# a[i:j] to access from index 'i' to index 'j-1'
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[1:5])

# Omit 'i' or 'j' to define only the start or end
    values
print(a[:6])
print(a[3:])
```

```
[2, 3, 4, 5]
[1, 2, 3, 4, 5, 6]
[4, 5, 6, 7, 8, 9]
```

Now that we know basic operations, lists, and how we can use functions,

let's discuss about conditions and loops.

Definition 21.1.3

Conditional statements are statements that performs certain tasks under specific conditions. **Loops** are structure that repeats a block of code until they are explicitly told to end. One more definition that will make our lives easier is **increment**, which is an assignment operator that augments certain values.

Here is an example of conditional statements.

```
# If, Else, Elif (else if)
banana_num = 10
if banana_num < 10:
    print("There are less than ten bananas")
elif banana_num == 10:
    print("There are ten bananas")
else:
    print("There are more than ten bananas")
```

```
There are ten bananas
```

Here, because "=" is reserved, we use "==" for "equal to." We could also do less than or equal to, greater than or equal to, and not equal to with $<=$, $>=$, and $!=$ respectively.

Regarding loops, "for loop" and "while loop" are two primary methods. Below are examples of how both loops can be used.

```
# While loop
love_banana = True
banana_num = 0

while love_banana and banana_num < 5:
    banana_num += 1 # Increment
```

```
    print(banana_num)

print()

# For loop
my_banana = ['yellow', 'green', 'mushy', 'sweet']

for i in range(len(my_banana)):
    print(my_banana[i])
```

```
1
2
3
4
5

yellow
green
mushy
sweet
```

First, `love_banana = True` is a Boolean value. In the while loop, what it does is while `love_banana` is True, we will increase the value `banana_num` by 1 and print them. Notice that the increment with `banana_num += 1` is a shorter form of writing `banana_num = banana_num + 1`. Moreover, we use `and` to make the loop continue if `love_banana` is true and `banana_num` is less than five.

In the second loop, we listed all elements in the list `my_banana`. The loop `for i in range()` tells Python to go through the code `print(my_banana[i])` from $i = 0$ to range(len(my_banana)) $- 1$.

We discussed very basic Python syntax that we must know to understand the following notes. Let's discuss core libraries NumPy and Matplotlib now, and save PyTorch for later.

§21.2 Modules and Libraries

First, what are modules and libraries?

Definition 21.2.1

Modules in Python are files that consists of definitions and statements for organization and reusability. **Libraries** are collection of modules and pre-written packages that can later be imported and distributed.

One of the reason why Python is so popular in ML and data science is because of the extensive libraries. Having the pre-written libraries specifically for ML and tensors like PyTorch simply makes our lives easier.

21.2.1 NumPy

NumPy is a Python package that helps handle multidimensional arrays. Let's get started by importing NumPy to our code. We could import it as whatever alias we want, but `np` is the standard for ML, so let's stick with that.

```
import numpy as np
```

Assuming that you have read the previous note or have strong foundations in linear algebra, we can implement the mathematical properties in Python with NumPy. Below is an example of how we can define vectors in different dimensions. Note that `.ndim` can be used to find the dimension of a vector.

```
arr_1d = np.array([1, 2, 3])
arr_2d = np.array([[1, 2, 3],
                   [4, 5, 6]])
arr_3d = np.array([[[1, 2], [3, 4]],
```

```
                    [[5, 6], [7, 8]],
                    [[9, 10], [11, 12]]])

print(f"Dimension: {arr_1d.ndim}, Shape: {arr_1d.
    shape}")
print(f"Dimension: {arr_2d.ndim}, Shape: {arr_2d.
    shape}")
print(f"Dimension: {arr_3d.ndim}, Shape: {arr_3d.
    shape}")
```

```
Dimension: 1, Shape: (3,)
Dimension: 2, Shape: (2, 3)
Dimension: 3, Shape: (3, 2, 2)
```

A simple way to understand multidimensional array in NumPy is stacking previous layers. For instance, `arr_2d` has the shape $(2,3)$. We can think of it as stacking two 1D array with three elements. Similarly, `arr_3d` has the shape $(3,2,2)$. We could think of it as stacking three layers of 2D arrays that consists of two 1D arrays. It is quite hard to visualize higher dimensional arrays, so let's take a look at different things that NumPy can do.

Below is an example where we utilize arange, reshape, dtype, and slicing techniques in NumPy. Let's create an array first.

```
# Making an array with numbers 0-17 and
# rerranging it to shape (3, 3, 2)
a = np.arange(18).reshape(3, 3, 2)
a
```

```
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],

       [[ 6,  7],
        [ 8,  9],
        [10, 11]],
```

```
       [[12, 13],
        [14, 15],
        [16, 17]]])
```

We can also find the data type with the attribute `.dtype`.

```
a.dtype.name
```

```
'int64'
```

This means that the individual elements in array `a` are 8 bytes integers. Just as we index and slice 1D array, we could do the same with NumPy. Consider the following example for a matrix.

```
a = np.arange(9).reshape(3, 3)
print(f"{a}\n")

# First row, second column element
print(a[0, 1])
# Entire second row
print(a[1, :])
# Entire third column
print(a[:, 2])
```

```
[[0 1 2]
 [3 4 5]
 [6 7 8]]

1
[3 4 5]
[2 5 8]
```

This was the basic introduction to NumPy. Please check out the documentation for further information.

21.2.2 Matplotlib

Do you want to visualize something with Python? Matplotlib is here for your need! Matplotlib is a Python library for visualization. Let's first import the library to take a look at an example.

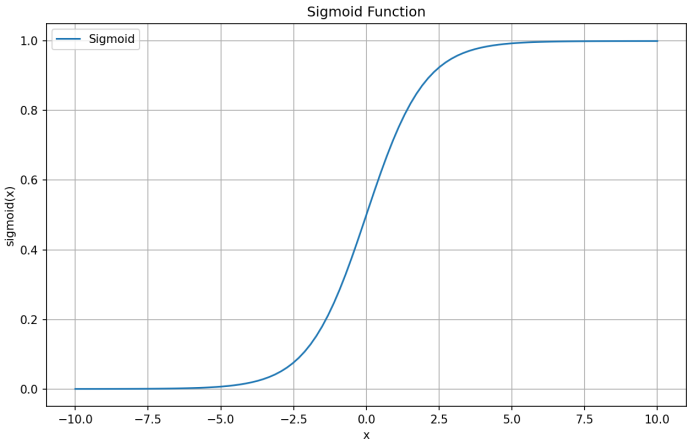
```
import matplotlib.pyplot as plt
%matplotlib inline
```

If you are using Jupyter Notebook like the examples in these notes, it is highly recommended to keep `%matplotlib inline` to make our lives easier by plotting directly in the notebook without an external window. Although Matplotlib is an essential library for visualizing model performance and analyzing, let's stick with basic plotting for now as I believe more information would be an overkill for this note.

For this brief introduction, let's plot Sigmoid function, which we will discuss in later notes.

```
# Define domain and the function
x = np.linspace(-10, 10, 100)
y = 1 / (1 + np.exp(-x))

plt.figure()
plt.plot(x, y, label='Sigmoid')
plt.grid(True)
plt.xlabel('x')
plt.ylabel('sigmoid(x)')
plt.title('Sigmoid Function')
plt.legend()
plt.show()
```



I know this intro was quite brief, but we will continue using amazing features in Matplotlib! For more information, please visit the official tutorial page.

In the next note, we will discuss foundational terms and concepts in ML. It will also include a high level introduction to large language model (LLM). Understanding the high level overview of how LLMs operate under the hood really helped me grasp the intuition. Without further ado, let's get started!

Note 22

Foundations and Intuitions

This note covers basic concepts as well as intuitions through the introduction of Large Language Models.

§22.1 Fundamental Terms and Concepts

This section is an introduction to basic terms and concepts that will be used throughout my notes. That being said, more advanced concepts will be introduced in latter notes. Please feel free to skip over this section if you already know the bold words.

Definition 22.1.1

Machine learning is a subset of artificial intelligence that learns from data. **Deep learning** is a subset of machine learning that utilizes neural networks.

The primary purpose of ML is to make AI "smarter". For instance, calculators are "dumb" in a sense that it requires an input and can only do certain tasks such as outputting calculated input. However, we can make it "smarter" by teaching data. We can clearly see that WolframAlpha is smarter than a normal scientific calculator. (Please note

that this is just an analogy not an example because WolframAlpha is a knowledge engine, not a machine learning model. But I mean you get the point.)

On the high level, you can think of an AI model as a giant function with billions of array of numbers, called parameters.

Definition 22.1.2

Parameters are the internal values in a model composed with **weights** and **biases** that are learned by the model. After the parameters are finalized with the training phase, the models generate **predictions** through the process called **inference**.

Indeed, models are prediction machines based on the trained parameters. One fun fact here is that it generally costs way less to run inference than to train the model. More in-depth explanation will be available in latter notes where we discuss neural nets.

22.1.1 ML Paradigms

In ML, there are three primary learning paradigms: supervised learning, unsupervised learning, and reinforcement learning. First, what does it mean to "learn" and "train"?

Definition 22.1.3

Machine learning utilizes the process of **training** to learn from data with algorithms. A common example would be dog and cat recognition from picture with **labeled** data.

There are multiple ways in which models are trained. One of the most used method out of the three paradigms is supervised learning.

Definition 22.1.4

When a model is trained with **supervised learning**, the model utilizes labeled dataset, or examples of input-output pairs, to learn.

With supervised learning, there are two primary predictions that models make.

Definition 22.1.5

Classification is a prediction where the label of an input data is being predicted. **Regression** is utilized to predict continuous values.

Classic examples of classification would be categorizing dog/cat pictures and email spam filtering. An example of regression would be the house price over years.

The second most common learning paradigm is unsupervised learning.

Definition 22.1.6

Unsupervised learning is a process of training a model with unlabeled data.

How is it possible for a model to learn without labeled data? One primary example algorithm is k-means clustering, where the model categorizes the data with relevance and similarity. To cluster high dimensional data, we often perform **dimensionality reduction** to reduce the dimension for clustering with the trust in **manifold hypothesis**. I won't go to deep into unsupervised learning and k-means clustering in this note, but you can check out this awesome visualization of k-means clustering.

The third most used learning paradigm is reinforcement learning.

Definition 22.1.7

Reinforcement learning is a process of training models through interactions with environment and reward system.

Personally, reinforcement learning is more complicated compared to the two paradigms introduced above. On the high level, an agent will take an action in an environment, where it returns **rewards** based on the action. The agent will then update its policies with the provided reward. The process is continued until the "adequate" amount is reached.

Training a model can be understood as writing a best fit mathematical function to training datasets that captures the holistic trend. There are few terms that describe the function.

Definition 22.1.8

A model is **overfit** if it learns too much from the data that it does not effectively capture that holistic pattern but small **noises**. On the other hand, an **underfitting** model is formed when it does not learn effectively from the data.

On the high level, our goal is to generalize and capture the general pattern and prevent overfitting or underfitting. We normally would stop training a model when it does not appear to improve, or is degrading, but how do we assess it?

Definition 22.1.9

To prevent overfitting and to evaluate the model during training, **validation** is utilized with **unseen data**. Unlike validation, **test sets** are used at the end of the training to evaluate the finalized model.

We have discussed the fundamental terms and concepts that would repeatedly appear in the following notes. Now let's build our high level intuitions on ML with the introduction to LLMs!

§22.2 Introduction to LLMs

You probably have heard of Large Language Models (LLMs) and even have your favorite ones. You may have also noticed that if you and your friend ask a same question to a LLM, you get different response, or at least outputs that are not identical from word to word. That is because LLMs are generative models. As always, let's get started by getting used to few vocabs.

Definition 22.2.1

Generative models are models that predicts and estimates the values on a probability distribution.

A generative model is also capable of taking in sequential data for a more context-rich prediction. An example of generative models would be an autoregressive model.

Definition 22.2.2

An **autoregressive model** is a generative model that predicts the future values in a sequence given the past input sequence.

For instance, when given the sequence of integers $0, 1, 1, 2, 3, 5, 8$, an autoregressive model is likely to generate 13 by using previously learned pattern and the input. In other words, it is simply a model that predicts the term $P(x_n \mid x_1, \ldots x_{n-1})$.

Definition 22.2.3

A **language model** is a model that utilize human languages and predicts the next tokens given the previous inputs.

An interesting fact about such models is that they sometimes exhibit **emergent behavior** where they actually "learn" and exhibit capabilities that were not explicitly present in the training data.

22.2.1 Tokenization

We know that at the end, language models are huge machines that predict the next numerical values. Then how do such models generate outputs in human language? This is where **tokenization** kicks in.

Definition 22.2.4

Tokenization is a process of breaking down inputs into smaller units called **tokens** which are often represented as numeric values.

To tokenize inputs, a tool called **tokenizer** is used with **Byte Pair Encoding (BPE) algorithm**. Intuitively, what we could do is assign numbers to each alphabets like 1 for a, 2 for b, and so on. However, this would be very inefficient considering the frequency of each letter and scalability. Instead, what we can do with the BPE algorithm is create a byte-efficient list of merged chunks (that are not necessarily a single letter) based on frequency. Here is an example from the OpenAI Tokenizer for GPT-5.x & O1/3.

Consider the following sentence:

Hippopotomonstrosesquippedaliophobia means fear of long words.

When tokenized, this 62 characters sentence turns into 17 tokens as following.

['H', 'ipp', 'opot', 'omon', 'st', 'ros', 'es', 'qu', 'ipped', 'ali', 'ophobia', ' means', ' fear', ' of', ' long', ' words', '.']

The token IDs of the tokens above are the following numbers.

[39, 3012, 101543, 43960, 302, 2199, 268, 351, 8193, 3010, 141868, 4748, 11747, 328, 1701, 6391, 13]

Now, to help the models "understand" the tokens and take the IDs as

input, we generally turn them into vectors through the process of **One-Hot Encoding (OHE)**, though the process may differ in modern Deep Learning for efficiency.

With OHE, the token ID i can be translated to a vector with 1 in index i and 0 in other slots. However, notice that OHE itself is not highly effective. To make the tokenizer more "meaningful" and "effective", we could use the process of embeddings. **Learned embedding vectors** uses somewhat low dimensional vectors (usually a few thousand dimensions) to make the vectors "meaningful". For example, in learned embedding vector, the difference between the vectors that represent king and queen will be similar to the difference between the vectors that represent man and woman. Modern models use **relative positional embeddings** to effectively distinguish the position of the words. (Let's save this topic for latter notes on transformers.)

22.2.2 Language Model Output

As you probably have guessed, language models do not simply output the texts that we see. An autoregressive transformer language model will output a sequence of multi-dimensional vectors with probability distribution for the next token based on the sequence of multi-dimension input vectors.

Definition 22.2.5

Logits are the **scores** of the next possible token represented by individual vectors in the output sequence.

Note that logits are *raw* scores. To convert them to probability distribution, we use something called **softmax**, in which we will discuss further in notes about neural networks.

Definition 22.2.6

Decoding is the process selecting the next token to convert raw and abstract model output into a human language with **detokenization**.

There are different strategies in decoding, but the two main methods are **greedy decoding** and **sampling**. In greedy decoding, you select the token with the highest probability. Meaning, the temperature (we will cover this is neural nets notes) does not influence the selection. Unlike greedy decoding, temperature does influence the result from sampling. In sampling, you randomly select the next token with accounting the individual probability. To put it simply, imagine you have a biased die with thousands of sides. You roll the die with the bias in each side, and the next token is selected by the result.

Model outputs are random and highly organized. Take a look at the following example. In the first case, say the model input is "I love" and the expected model output is "I love potatoes". In the second case, the model input is "Veni, vidi," and the expected output is "Veni, vidi, vici". When the model returns an output, which case do you think the model has the higher probability of returning the expected output? To answer this question, we must understand that models are composed of billions of parameters that are trained with countless data. The model likely learned lots of texts that includes "I love dogs", "I love coffee", and more. However, for a model dedicated for English, it is highly unlikely that it learned another phrase that starts with "Veni, vidi," other than "Veni, vidi, vici", the famous line from Julius Caesar. The model probably assigned lots of logits and converted them to probability distribution with softmax. While the first case have somewhat flat distribution, the second case will likely assign "vici" with high probability like 0.98 or 0.99. Therefore, the second model has the higher probability of returning the expected output.

Now that we have some intuitions on model outputs, let's discuss about

post-training.

§22.3 Improving Large Language Models

Up until now, we have discussed fundamental terms and very high-level overview of pre-training phase. This section discusses very high-level introduction to how a raw autocomplete model transitions to a model that we use in daily life.

22.3.1 Introduction to Post-Training

One of the most important phase of training that makes the chatbots chatbots are post-training. When a base model is developed through pre-training phase, it is simply an autocomplete model. Whereas, an aligned model is capable of generating conversation-like responses. For instance, if the input is "Why are dogs so cute?", the base model will generate responses like "What are cats doing?", which does not feel like a conversation. Whereas, an aligned model will generate responses like "People find dogs cute because humans developed a biological phenomena called baby schema."

The post-training process contains two primary phases: Supervised Fine-Tuning (SFT) and Reinforcement Learning from Human Feedback (RLHF) On the high-level, SFT is the process where human provides prompt and response samples. The model then learns from the samples to provide a chatbot-like experience. After the phase, the model can be reinforced through RLHF. There are multiple ways of doing this, but a common method is making another "reward model" to judge and provide feedback on a model's response. Instead of a human manually judging and rewarding every responses, reward model will predict how a human expert will reward (usually an integer from 0 to 5) a response to more effectively align the model. The specific process is quite out of scope for this note, and will be introduced in the following notes.

Most of the "learning" happens in pre-training. However, alignment is

necessary to improve HHH, or helpfulness, honesty, and harmlessness. And yes, SFT and RLHF are popular, but they are not the only methods. Few other examples contain LIMA, RLAIF, DPO, PPO, and URIAL.

22.3.2 Reasoning

As you probably have noticed, LLMs are predictions models. In other words, they do not "reason" like how us humans reason. Language models will predict the next likely token that will provide a plausible response to a reasoning task. One way to test if the models can actually reason is by using benchmarks. One of the foundational datasets, which is considered outdated now, is GSM8K by OpenAI. This dataset is used to evaluate if a model can reason grade school level math. The modern, highly challenging, benchmark includes FrontierMath. As you have noticed, benchmarking and reasoning is a giant hurdle for many LLMs. Modern engineers classify reasoning as another section in post-training to improve a model's logic.

One of the most effective way to improve LLM reasoning is to use **prompting strategies**.

Definition 22.3.1

One technique is **K-shot prompting** where you provide k examples for the model to follow a template. Zero-shot prompting will be when $k = 0$, where you provide no examples, but instructions. The second prompting strategy is **Chain-of-Thought (CoT) prompting**. This strategy explicitly tells a model to break a complex reasoning tasks into intermediate steps.

Let's take a look at an example on how K-shot prompting and CoT prompting work together.

Instruction:

Consider the following examples to provide a response to the final equation. Provide a step by step explanation for your answer as demon-

strated in the samples.

Example 1:

Q: Henry the dog loves tomatoes. He originally had 5 tomatoes. Henry ate 3 of them, and sneaked into the owner's yard to secretly bring 4 more. How many tomatoes does Henry have now?

A: Henry started with 5, but ate 3. Therefore, he has $5 - 3 = 2$ tomatoes left. Because he brought 4 more from the owner's yard, he now has $2 + 4 = 6$ tomatoes. Henry has 6 tomatoes now.

Example 2: Q: Bob the yak loves burgers. One day, his lizard friend

Sofia told him that she can bring three boxes of burgers than contain 2 burgers each. Bob instantly pleaded Sofia, and he received the boxes. Out of hunger, he ate three burgers. How many burgers does he have now?

A: It is assumed that Bob does not have any burgers before the interaction with Sofia. With Sofia's virtue, Bob now has $2 \cdot 3 = 6$ burgers. Because he ate three burgers from six in total, Bob now has $6 - 3 = 3$ burgers left.

Question:

Charlotte the cat has five close friends. She wishes to split her 120 kg tuna equally with her friends. After the allocation, Charlotte ate 4 kg of tuna. How much tuna in kg does Charlotte have left?

Output 22.3.2

Charlotte initially has 120 kg of tuna. Because she split it with her friends, she has $\frac{120}{5+1} = 20$ kg of tuna after the allocation. Since Charlotte ate 4 kg of tuna from 20 kg, she now has $20 - 4 = 16$ kg of tuna.

As demonstrated, CoT has the potential to increase reasoning of LLMs. However, it is important to note that although performance may significantly ameliorate, numerous potential consequences follow. For instance, because the model includes the intermediate steps, the token generation considerably increases, which will increase the cost. Moreover, when

model starts to hallucinate in the middle of the reasoning, the entire output from the hallucination will no longer be useful. Furthermore, models sometimes "lie". What I mean to lie is that some models tend to provide a "plausible CoT" without actually providing one. This is more related to AI safety, and let's save that for final notes.

Another technique that could improve reasoning and performance is **self-consistency**.

Definition 22.3.3

Self-consistency is a technique where a model generates numerous possible outputs to an input, and aggregate the outputs for the final output.

The primary motivation of the technique is that just as mathematical problems have numerous paths to an answer, we could derive an "accurate" output with different reasoning. However, like any other techniques, there are limitations. One somewhat self-evident caveat is that this requires much greater computation. Because the technique requires multiple reasoning and aggregation for the final, the amount of computation drastically increase. One other issue is that this self-consistency only works on closed-solution prompts. I mean different people and different outputs can have differing opinions. Although this technique is not fully outdated, modern ML engineers use more effective and efficient techniques.

The field of reinforcement learning is one of the most rapidly developed field in ML. For instance, unlike RLHF, the emerging method Reinforcement Learning with Verifiable Rewards (RLVR) allows models to align themselves with minimum human contributions. In the next note, we will discuss about neural network from scratch!

Note 23

Deep Dive into Deep Neural Networks

Neural networks (or neural nets) can be comprehended as a complex function that does not require manual feature engineering. Due to the high level of abstraction and automation in learning, neural nets are highly scalable. This note covers neural nets with mathematical interpretations and its application with Python.

§23.1 Definitions and High Level Overview

As always, let's get started with the fundamental terms and high level overview of what neural nets are. Before the introduction of neural nets, traditional models like linear/logistic regression and decision trees were used. The problem with traditional models is that you are likely required to manually adjust the **features**. With neural nets, the features are automatically engineered, making us to focus on the architecture.

23.1.1 High Level Architecture

To understand the high level architecture, consider the diagram below.

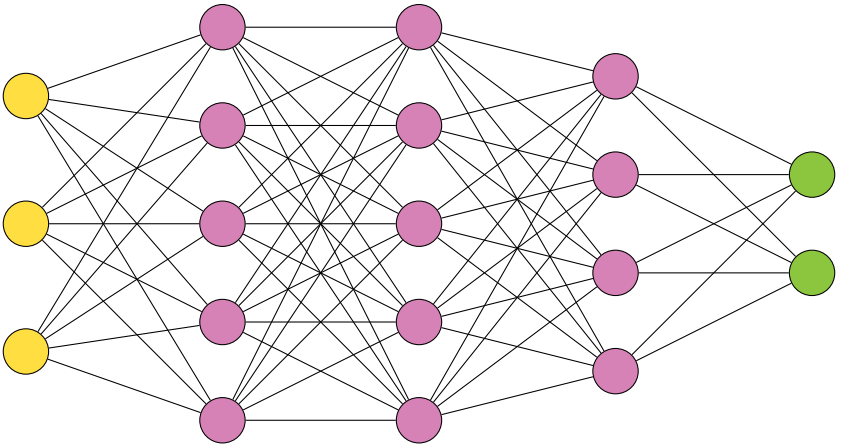


Figure 23.1: A visualization of a vanilla neural network (MLP)

From now, let's refer to the nodes as **neurons**.

Definition 23.1.1

Neurons are the fundamental unit in neural networks that process input into output with set of parameters.

From the diagram, the neurons are organized in to different columns. We call the columns **layers**.

Definition 23.1.2

The **input layer** is the layer that receives inputs for the neural network. **Hidden layers** are where the input data is processed with the learned parameters to extract information. The **output layer** is the final layer that represents the final prediction of the neural network.

In the diagram above, the input layer is represented as the layer with yellow neurons, hidden layers as the layers with purple neurons, output layer with the layer with green neurons. One of the advantages of neural network is that there is simply no limit, except computational limit. You can have unlimited amount of neurons in each layers and

unlimited amount of hidden layers. The best neural net will be effectively structured in a way that there are sufficient neurons to learn from while avoiding overfitting and decreasing computational cost.

Definition 23.1.3

Connections are the links that connects the neurons with each other.

Just as our biological neurons are interconnected to each other, artificial neurons are connected to effectively extract the holistic feature from the input. Now it is worth noting that each neuron represents **feature**.

Here is a brief example. Say your neural net will classify handwritten numbers. The number 9 and 0 retains a similar feature. Both the numbers have a loop in the top. Therefore, in this scenario, the initial hidden layers may have similar features when categorizing the numbers.

In the past we used **perceptrons**, or neurons that uses step functions with binary output. However, modern multilayer perceptron (MLP) uses a more flexible neurons with continuous activation functions. Now let's take a look at what individual neurons have with the diagram below.

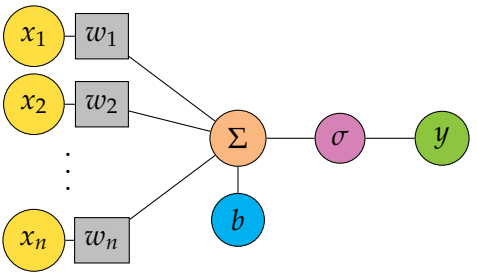


Figure 23.2: Visualization of a single neuron

On the high level, an individual neuron takes the weighted sum of the previous layer with the trained parameter. Bias is then added to make the activations more flexible. Then, an activation function is used to determine the final activation of the neuron.

Definition 23.1.4

Activation of a neuron indicates the significance of a neuron, or presence of a feature from the input data. An **activation function** is a function that generalizes each activation and introduce **non-linearity**.

In the diagram above, the purple color represents the activation function and the green color represents the final output of the neuron. Here, non-linearity is important for generalization with activation function. The activation is represented as a linear weighted sum. However, for generalization, activation functions like ReLU (Sigmoid is considered outdated) are used.

Here, the superscript represents the layer and i identifies the neuron in the layer. The function σ is an activation function. It could be sigmoid, ReLU, tanh, and other activation functions.

23.1.2 Training

Inference is rather straight forward, but how do we "train" a model? Well, that's what this part of the note is all about. There are few terminologies that we need to know, and I think we can discuss them as we go along.

Definition 23.1.5

Forward pass, also known as **forward propagation**, the process of generating output with input data.

Here, it is worth noting that forward pass is different from inference in a sense that the process of inference only applies to a complete trained model while forward pass also applies to models being trained.

Let's discuss the definitions and underlying concepts in individual process after capturing the big picture. Generally speaking, the following diagram captures the training process.

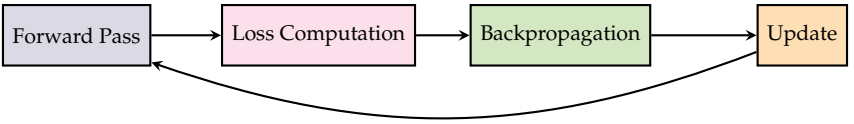


Figure 23.3: Overview of the Training Process

The first thing that we do after the forward pass is to calculate the **loss**.

Definition 23.1.6

Loss functions are algorithms that calculate the **loss**, or the performance of the prediction compared to the expected outcome.

There are many loss functions, but the two dominant functions are **mean squared error (MSE)** and **cross-entropy** (a.k.a. negative log-likelihood.) Generally, MSE is great for regression and cross-entropy excels at classification. Therefore, cross-entropy will commonly be used for LLMs for token predictions. Let's take a look at each functions.

Following the convention, let $\hat{y}_n$ and y_n be the prediction and the true value respectively. With MSE, the loss will be represented as $\frac{1}{n}\sum(\hat{y}_i - y_i)^2$. The intuition behind this formula is we always want the loss to be positive as it is comfortable to think that our goal is to reduce the loss. If the loss is negative, it will be hard to apply this sense. That's why we squared the difference. Now you might ask, why not absolute value? Indeed, there is a function Mean Absolute Error (MAE), but we like to square the values as it amplifies the loss. With cross-entropy, its a bit more complicated, but a simplified version would be $\sum(-y_i \log(\hat{y}_i))$. More on cross-entropy will be discussed soon!

We know we can calculate the loss, but how do we actually improve the model with the calculated loss?

Definition 23.1.7

An **optimizer** is an algorithm that determines the necessary changes for model weights.

The dominant optimizer type is **gradient descent** with its variants like Adam, RMSProp, and SGD. There's quite a bit of content for gradient descent for ML, so let's discuss that in a separate section of the note.

23.1.3 Gradient Descent and Learning Rate

Before we implement the ideas and concepts above with code, I wanted to discuss gradient descent and training more in-depth to actually train our model. The partial derivatives and mathematical definition of gradient descent was discussed in the first note. Here, we will discuss about gradient descent specifically for ML.

Definition 23.1.8

Gradient descent as an optimizer represents how the weights and biases of a model are updated based on the gradients of the loss function.

You would typically start from a random position to move in the opposite direction of the gradient of the loss function. Before I introduce mathematical equation, here are two famous analogies of a ball rolling down the hill and finding a way in a foggy forest. Starting with the first analogy, consider the diagram below.

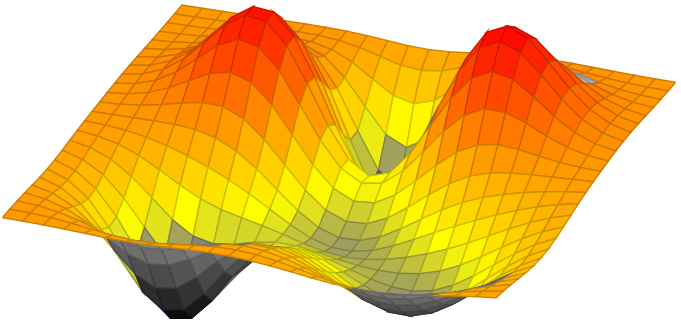


Figure 23.4: Ball Rolling Downhill Analogy

Imagine rolling a ball from the graph above. After some time, it will naturally fall into one of the pits. If we are lucky, we will find the global minimum, if not, we will fall into one of the local minima. To see how we roll a ball, consider two diagrams below.

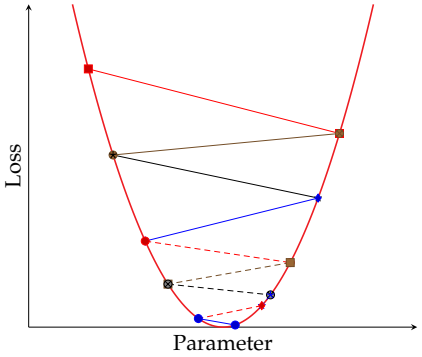


Figure 23.5: Visualization of Case I

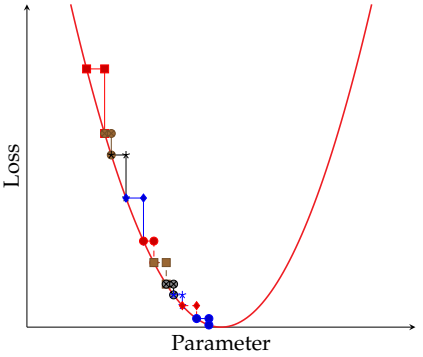


Figure 23.6: Visualization of Case II

Taking a look at Figure 23.5, imagine your random point in the red square. Because the slope of the line tangent to the point in negative, you want to move to the right, or in positive direction. This time, your learning rate is high that the slope of the line tangent to the next landing position is positive. Therefore, you want to move to the left, which is the negative direction. After continuing this process over and over again while adjusting the learning rate, you will likely find the local minimum. In Figure 23.6, our learning rate is low. Despite being low, it will still lead to the local minimum. One thing to note is that you don't want

your learning rate to be low or high as you might need to train forever or overshoot. Learning rate is often denoted as η, and we normally would adjust it with the gradient, which we will see in a bit. Of course in real training, we won't be finding the slope of the tangent line as we do in the 2-dimensional plane. In higher dimension, we would find the gradient. Please refer to the first note for more information!

Another very popular analogy is foggy mountain analogy. Imagine you are in the top of the mountain, and its getting dark. You want to get to the lowest point as fast as possible. However, you can't see anything beyond your local information because the mountain is very foggy. What would you do? We would find the gradient and multiply it with the learning rate to determine our step size. Then, with your local information, we would be able to approach the local minimum of the foggy mountain. Ideally, we should be hoping that we reach the lowest point and villages.

Now, before we continue for building our own neural net without machine learning libraries, let's conclude the concepts by discussing different types of gradient descent and their effects.

Types of Gradient Descent

There are many types of gradient descents like Batch GD, Mini-batch GD, Stochastic GD, and Momentum-Based GD. However, the three major types are Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD), and Mini-batch Gradient Descent. On the high level, consider the following definitions.

Definition 23.1.9

With **Batch Gradient Descent (BGD)**, each update will occur based on the entire training examples, which are often averaged. In contrast, **Stochastic Gradient Descent (SGD)** updates based on a single data point. **Mini-batch Gradient Descent** is the middle ground of BGD and SGD that it updates based on few data points like 32 or 64, usually in a power of 2.

From the sense, we can notice that the biggest difference in use case would be efficiency and accuracy. BGD is generally more accurate in each update than SGD as it is based on the entire data points. However, SGD is generally more efficient as it does not have to perform massive computations for each update to incorporate all the training examples. One interesting thing to note is that because SGD is "stochastic", or more noisy, it is sometimes better at optimizing the updates as it moves more randomly than BGD.

When visualizing, we often consider the following diagram.

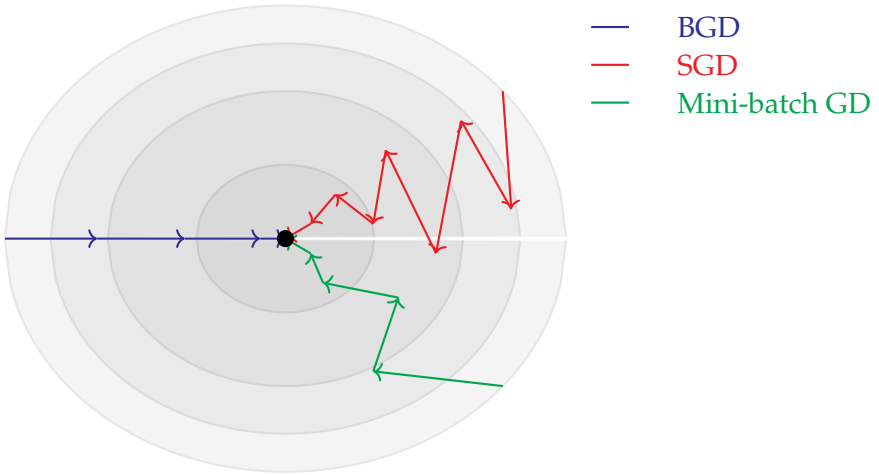


Figure 23.7: Visualization of Different Types of Gradient Descent

As you can see, BGD often heads straightly towards the minimum as the entire training examples are incorporated. Unlike BGD, SGD shows more random and noisy movement and often requires more updates to reach the local minimum. Finally, Mini-batch GD is like the movement in between those two. Its is some what random, but less noisy SGD that it often requires less updates.

Overall, this is what artificial neural nets are about. The pre-trained and inter-connected neurons work together to predict that output as a giant function. This giant function are composed of weights and biases, which

are fine-tuned during the training process. During training, you calculate the loss and tweak the parameters to minimize the loss through gradient descent. You repeat the training process until your model learned well enough, but not too much to prevent overfitting. That's pretty much it in the high level! For our next section, let's discuss more on training and gradient descent before we actually try and build a neural network from scratch without ML libraries like PyTorch and TensorFlow.

Congratulations! We just finished the high level overview of what neural nets are and what vanilla neural nets are composed of. Before we actually try and implement the concepts with Python to build a vanilla neural net without machine learing libraries like PyTorch and TensorFlow, let's take a look at mathematical equations to write them in code.

§23.2 Mathematical Interpretations

If you have not already read the first note on math for ML, please do so as this section of the note does not include explanations for mathematical concepts, but their interpretation here with neural networks.

Starting off, let's start with some conventions with parameters for the following neural net with two inputs, two hidden layers with four neurons respectively, and a single output.

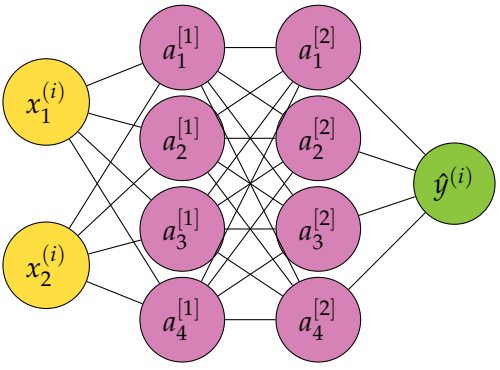


Figure 23.8: A visualization of a neural network with layers of 2, 8, 8, and 1

As you can see in the diagram above, the input of the i^{th} training example is represented with the notation $x_n^{(i)}$ for integer n as 1 through the total number of inputs. Similarly, we usually count the first hidden layer as the first layer, and the activations are represented as $a_n^{[l]}$ were l represents the layer and n again as the integer from 1 through the total number of neurons in the layer. Finally, we use $\hat{y}^{(i)}$ to represent the prediction for the i^{th} training example.

Following the convention, we could write a single neuron as the following. Note that you might be familiar with σ instead of g, but here let's use g for activation function as σ usually represents sigmoid.

Theorem 23.2.1: Formula for a Neuron

For $z_i^{[l]}$ defined as the following, the activation of a neuron can be represented as the following.

$$z_i^{[l]} = \sum_j w_{ij}^{[l]} a_j^{[l-1]} + b_i^{[l]}$$

$$a_i^{[l]} = g\left(z_i^{[l]}\right) = g\left(\sum_j w_{ij}^{[l]} a_j^{[l-1]} + b_i^{[l]}\right)$$

One reason why linear algebra is so important in machine learning is because we could represent the entire layer with simple matrix operations. Consider the following example for the second layer.

$$\begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} & w_{13}^{[2]} & w_{14}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} & w_{23}^{[2]} & w_{24}^{[2]} \\ w_{31}^{[2]} & w_{32}^{[2]} & w_{33}^{[2]} & w_{34}^{[2]} \\ w_{41}^{[2]} & w_{42}^{[2]} & w_{43}^{[2]} & w_{44}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \\ b_4^{[2]} \end{bmatrix} = \begin{bmatrix} a_1^{[2]} \\ a_2^{[2]} \\ a_3^{[2]} \\ a_4^{[2]} \end{bmatrix}$$

Similarly, we could write different notations for updates.

Theorem 23.2.2: Adjusting Parameters with Gradient Descent

Let l_i, b_i, J, and η represent weight, bias, cost function, and learning rate respectively. The new weight and bias can be represented with the following formula.

$$w_2 = w_1 - \eta \frac{\partial J}{\partial w_1}$$
$$b_2 = b_1 - \eta \frac{\partial J}{\partial b_1}$$

As you can see above, we normally use the cost function J than the loss function L as the cost function captures the entire loss while the loss function only represent the loss per example. Now that we discussed the basic notations and mathematical formulas for the concepts that we discussed, let's take a look at math in backpropagation.

23.2.1 Deriving Equations and Backpropagation

Personally it took me a while to fully understand backprop to the state where I would be able to build a vanilla neural net from scratch. Here, I would like to share more in-depth understanding of backpropagation across numerous layers.

To be honest, it took me few days to understand it since I decided to study mainly because I didn't understand the delta rule. After reading this note, I highly recommend please try and write your own network without assistance from LLMs. I though I understood backprop, which in reality was only the last layer. That understanding is probably all you need to build the first thing that we will create in the next section. However, I was stuck when trying to do backprop across numerous layers. To discuss this, I thought it would be helpful for us to actually rebuild or derive main equations involved in a neural network and backprop, so let's start with that. If you haven't already, please read the first note if you are unfamiliar with derivatives of multivariable functions as I won't be going over them here.

Before we derive generalized equations, let's start with a simple neural net that has two hidden layers, 2 inputs, 3 neurons in each hidden layer, and two outputs. That way, we can track the indices for matrices.

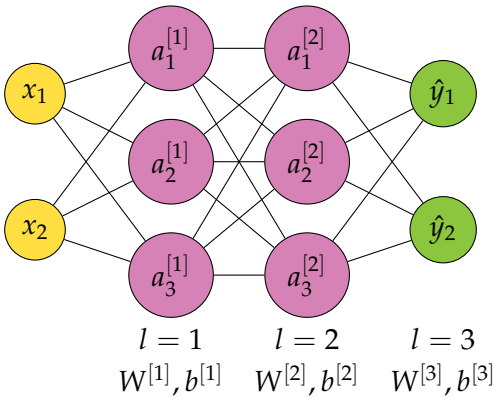


Figure 23.9: A visualization of a neural network with layers of 2, 3, 3, and 2

For simplicity, let's use MSE for our loss function, but we will be generalizing it later. Since $L_1 = \frac{1}{2}(\hat{y}_1 - y_1)^2$ and $L_2 = \frac{1}{2}(\hat{y}_2 - y_2)^2$, the following equation is obtained.

$$L = \sum_{k=1}^{2} \frac{1}{2}(\hat{y}_k - y_k)^2$$

Moreover, the preactivation output of the second hidden layer can be written as following.

$$\begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \\ z_3^{[2]} \end{bmatrix} = \begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} & w_{13}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} & w_{23}^{[2]} \\ w_{31}^{[2]} & w_{32}^{[2]} & w_{33}^{[2]} \end{bmatrix} \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \end{bmatrix}$$

From the matrix multiplication above, we can write a single output as

the following.

$$\begin{aligned} z_2^{[2]} &= w_{21}^{[2]} a_1^{[1]} + w_{22}^{[2]} a_2^{[1]} + w_{23}^{[2]} a_3^{[1]} + b_2^{[2]} \\ &= \sum_{j=1}^{3} w_{2j}^{[2]} a_j^{[1]} + b_2^{[2]} \end{aligned}$$

Now keep in mind the indices! The 2 in w_{2j} and b_2 is the same 2 as z_2. Generalizing for our cases, we can write the preactivation outputs as the following.

$$z_i^{[2]} = \sum_{j=1}^{3} w_{ij}^{[2]} a_j^{[1]} + b_i^{[2]}$$

Nice, now that we have the forward pass for our examples, let's generalize all equations that we have now. For MSE, we could write it as the following.

$$L = \sum_k \frac{1}{2} (\hat{y}_k - y_k)^2$$

For individual preactivation output, we could write it as the following equation where n represents the number of neurons in layer $l-1$.

$$z_i^{[l]} = \sum_{j=1}^{n} w_{ij}^{[l]} a_j^{[l-1]} + b_i^{[l]}$$

Now with the equations in mind, let's get into a more confusing but fun part with loss and gradient. This part is important! We are not computing the loss of individual neurons. We will be computing *gradients* instead. Loss will typically be calculated once with the final outputs.

Let's start with the last layer. Remember! This is the last layer. To make it more special, let's use l_f to denote the final layer.

$$\frac{\partial L}{\partial w_{ij}^{[l_f]}} = \frac{\partial L}{\partial a_i^{[l_f]}} \frac{\partial a_i^{[l_f]}}{\partial z_i^{[l_f]}} \frac{\partial z_i^{[l_f]}}{\partial w_{ij}^{[l_f]}}$$

In this chain rule here, we don't add other gradients because it is the last layer. The weights for the output layer does not go through other

paths, but straight to the loss. That's why we don't add gradients. This would be different for different layers. Using the definition, MSE, and activation function g, the following equation is obtained.

$$\frac{\partial L}{\partial w_{ij}^{[l_f]}} = \left(a_i^{[l_f]} - y_i\right) g'\left(z_i^{[l_f]}\right) a_j^{[l_f-1]} = \delta_i^{[l_f]} a_j^{[l_f-1]}$$

Notice that $\delta_i^{[l_f]}$ will be different depending on the loss function that we use. To address that and to make our lives easier for hidden layers, we define $\delta_i^{[l]}$ as the following.

$$\delta_i^{[l]} = \frac{\partial L}{\partial z_i^{[l]}} = \frac{\partial L}{\partial a_i^{[l]}} \frac{\partial a_i^{[l]}}{\partial z_i^{[l]}}$$

Again, because the path from z_i to a_i does not require additional paths, we do not add other gradients. Now here comes the **delta rule**. Let's start deriving them together. Consider the following equation.

$$\begin{aligned}\frac{\partial L}{\partial a_i^{[l]}} &= \sum_j \left(\frac{\partial L}{\partial z_j^{[l+1]}} \frac{\partial z_j^{[l+1]}}{\partial a_i^{[l]}}\right) \\ &= \sum_j \delta_j^{[l+1]} w_{ji}^{[l]} = \sum_j w_{ji}^{[l]} \delta_j^{[l+1]}\end{aligned}$$

Substituting back to the definition of $\delta_i^{[l]}$, the following equation is obtained by the **delta rule**.

$$\delta_i^{[l]} = g'\left(z_i^{[l]}\right) \sum_j w_{ji}^{[l]} \delta_j^{[l+1]}$$

However, writing all that for each gradient is quite tedious. Therefore, we get our general formula shown below, which will be used to build our vanilla MLP.

$$\frac{\partial L}{\partial w_{ij}^{[l]}} = \delta_i^{[l]} a_j^{[l-1]}$$

That was for the weights. Good news is that bias is more simple! Con-

sider the following equation.

$$\frac{\partial L}{\partial b_i^{[l]}} = \frac{\partial L}{\partial a_i^{[l]}} \frac{\partial a_i^{[l]}}{\partial z_i^{[l]}} \frac{\partial z_i^{[l]}}{\partial b_i^{[l]}} = \delta_i^{[l]} \frac{\partial z_i^{[l]}}{\partial b_i^{[l]}} = \delta_i^{[l]}$$

Now with that formulas and learning rate η, we can update the weights and biases as shown in Theorem 23.2.2. There are more advanced concepts like Jacobian matrix for ease in representing, but this is pretty much it with backprop. Calculating the gradients and updating based on the loss.

§23.3 Building Vanilla NN From Scratch

For our toy examples, we will create two things. The first one is technically not a neural network as it does not contain conventional neurons and structures. The second one is an actual neural net that is very simple. In this section, I will share the entire code first, then provide explanations.

23.3.1 Simple Stochastic Gradient Descent-Based Learning Engine

What we will make now is a simple stochastic gradient descent-based learning engine. This isn't technically a neural network, as you will see why in the code, but I decided to include it here as making similar examples really helped me understand backprop with code.

Our task with this learning engine is simple. We are going to predict e^x for limited range with concatenated Maclaurin Series. If you recall from your calculus courses, the following equation holds by Taylor series.

$$e^x = \sum_{n=0}^{\infty} \frac{1}{n!} x^n = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

Here, for limited x, we will be approximating e^x with the following

polynomial where b is the bias and w_i are the weights.

$$e^x \approx b + w_1 x + w_2 x^2 + w_3 x^3 + w_4 x^4 + w_5 x^5 + w_6 x^6 + w_7 x^7 + w_8 x^8$$

We will get back to this on why our x should be limited, but let's see our structure first.

We will be "training" the engine with 200 examples from -1.00 to 0.99 inclusive. Then, we will be testing it with 50 validations from 1.00 to 1.49 inclusive. This backprop machine is stochastic as it will adhere to the following structure.

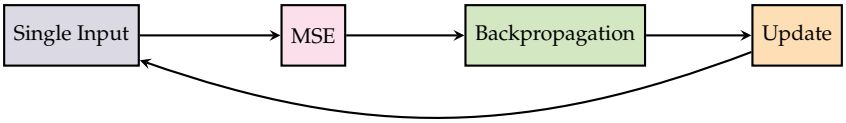


Figure 23.10: Overview of the Simple Learning Engine

As you can see, we will be using MSE for our loss function, and we do not need activation functions like sigmoid and tanh because we don't need one in our simple example. Okay now here is the breakdown of the entire notebook available here.

Detailed Explanation

Now let's take a look at individual cells.

```
1 import math
2 import random
3 import matplotlib.pyplot as plt
4 %matplotlib inline
```

Here, I tried to keep it as minimal as possible. Instead of NumPy, I decided to use the math module for all math stuffs like `math.exp()`. We need `random` as it would make our lives easier than to manually set the eight weights and the bias. Matplotlib of course since we want to see

how it learns for limited range of input.

```
train_x = [i/100 for i in range(-100, 100)]
train_y = [math.exp(i/100) for i in range(-100,
    100)]
test_x = [i/100 for i in range(100, 150)]
test_y = [math.exp(i/100) for i in range(100, 150)]
```

Setting up the dataset for our engine is also quite simple. Instead of writing every single number, I just decided to loop through them and store them in an array, and that is `train_x`. The values for `train_y` is also quite straightforward as it is e^x for corresponding x. The same goes for the test set. I tried to keep it 80% and 20% for training and validation, but that is not necessary and it varies by models. However, I decided to use that here because that's pretty much what most people do when they first train. Now let's get into the fun stuffs.

```
class SGDBLE:
    def __init__(self):
        random.seed(3141592) # So that we get the
            same result
        self.ws = [random.uniform(-1, 1) for _ in
            range(8)]
        self.b = random.random()
```

Now I decided to store the core functions in the class SGDBLE as we will be using `self` to store attributes and objects. Here we set the specific seed so that you will also get the same result when you run the code in somewhere. Next we set weights and biases and stored them in `self` with our set random values.

```
def forward(self, x):
    self.ypred = sum(w * x**k for w, k in zip(self.
```

```
            ws,range(1, 9))) + self.b
    return self.ypred

def loss(self, y, ypred):
    self.loss_val = 0.5 * (y - ypred)**2
    return self.loss_val
```

This is where we set some core functions. Forward is forward pass and it is somewhat straight forward as that is how we defined it. For the loss function, we are using MSE, i.e. $\frac{1}{2}(y - \hat{y})^2$.

```
def grads(self, x, y, ypred):
    base_der = ypred - y # i.e. dL / dypred
    self.grads_w = [base_der * x**i for i in range
        (1, 9)]
    self.grad_b = base_der

def update(self, lr):
    for i in range(8):
        self.ws[i] -= lr * self.grads_w[i]
    self.b -= lr * self.grad_b
```

This is the core part of our engine. Let's briefly look at the math and write the loop out for some to see what's actually going on. Consider the following equation by the chain rule.

$$\frac{dL}{dw_i} = \frac{dL}{dy} \cdot \frac{dy}{dw_i}$$

Notice that $\frac{dL}{dy} = \frac{d}{dy}(\frac{1}{2}(y - \hat{y})^2) = y - \hat{y}$. This is `base_der`. Continuing with the chain rule, we can write the weights and the bias as the

following.

$$\frac{dL}{dw_1} = \frac{dL}{dy} \cdot \frac{dy}{dw_1} = \text{base_der} \cdot x$$
$$\frac{dL}{dw_2} = \frac{dL}{dy} \cdot \frac{dy}{dw_2} = \text{base_der} \cdot x^2$$
$$\frac{dL}{db} = \frac{dL}{dy} \cdot \frac{dy}{db} = \text{base_der} \cdot 1$$

Now the update function. Recall $w_2 = w_1 - \eta \frac{dL}{dw_1}$ and $b_2 = b_1 - \eta \frac{dL}{db_1}$. That is exactly what we did, but for weights we have to loop through each element.

```
def train(self, epochs, lr):
    losses = []
    for epoch in range(epochs):
        loss_total = 0
        for i in range(len(train_x)):
            ypred = self.forward(train_x[i])
            loss_total += self.loss(train_y[i],
                ypred)
            self.grads(train_x[i], train_y[i],
                ypred)
            self.update(lr)
        loss_avg = loss_total / len(train_x)
        losses.append(loss_avg)

        if (epoch+1) % 100 == 0:
            print(f"Epoch: {epoch+1}/{epochs}, Loss
                : {loss_avg}")

    return losses
```

This is the training loop. Of course we don't want to stop at a single epoch especially in stochastic setting. You can ignore the `losses` in the beginning as we only use them to graph. For each epoch, we want to set the total loss to 0 as they will accumulate. Then, for all x values in our

training set, we will predict, calculate the loss, backprop, an update as shown. Note that we have to set `epoch` and `lr`, which is the learning rate. Now we will take the average of the total loss and print it every 100 epoch to see how well our engine is learning.

```
def test(self):
    loss_total = 0
    for i in range(0, len(test_x), 2):
        ypred1 = self.forward(test_x[i])
        loss_total += self.loss(test_y[i], ypred1)

        if i + 1 < len(test_x):
            ypred2 = self.forward(test_x[i + 1])
            loss_total += self.loss(test_y[i + 1],
                ypred2)

            print(f"Test {i+1}
                                        Test {i+2}")
            print(f"y:     {test_y[i]}      y:
                {test_y[i+1]}")
            print(f"ypred: {ypred1}     ypred: {
                ypred2}")
            print()
        else:
            print(f"Test {i+1}")
            print(f"y:     {test_y[i]}")
            print(f"ypred: {ypred1}")

    loss_total /= len(test_x)

def print_params(self):
    coeff = [1 / math.factorial(i+1) for i in range
        (8)]
    for i in range(8):
        print(f"w{i+1}: {self.ws[i]}, Expected: {
            coeff[i]}")
    print(f"b: {self.b}, Expected: 1")
```

```
27
28 model = SGDBLE()
```

The test loop is also straight forward. It predicts, calculates the loss, and print them as shown. Also, because we know the coefficients for the Maclaurin Series, I decided to see if the parameters match with the actual coefficients. Finally, the last line is to be able use the model.

```
1  def graph_loss(losses):
2      plt.plot(losses)
3      plt.xlabel("Epoch")
4      plt.ylabel("Loss")
5      plt.show()
6
7  def graph_pred():
8      xs = [i / 10 for i in range(-100, 200)]
9      true = [math.exp(x) for x in xs]
10     pred = [model.forward(x) for x in xs]
11
12     plt.plot(xs, true, label="e^x")
13     plt.plot(xs, pred, label="Prediction")
14     plt.yscale("log")
15     plt.legend()
16     plt.show()
```

This part is plotting with Matplotlib. Please refer to the second note if you are confused! One thing though `yscale` is necessary as exponential function will get too large to show.

```
1 losses = model.train(1000, 0.005)
2 graph_loss(losses)
3 model.print_params()
4 graph_pred()
```

This part is the actual training part. We set the epoch to 1000 and learning rate to 0.005. This learning rate might be too small for the start, but I realized that this gives slightly better results than 0.001 in this setting. Then we graph and show loss, parameters, and predictions.

```
model.test()
```

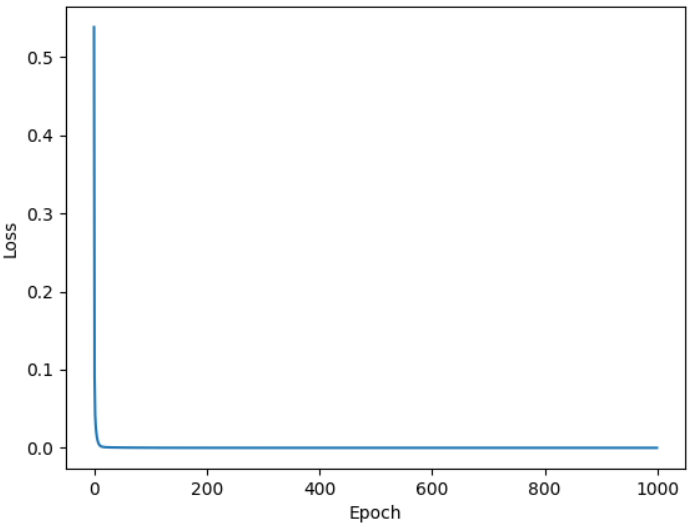
This part is for testing. Now that we have seen the entire code and explanation for it, let's now analyze our results.

Analyzing the Results

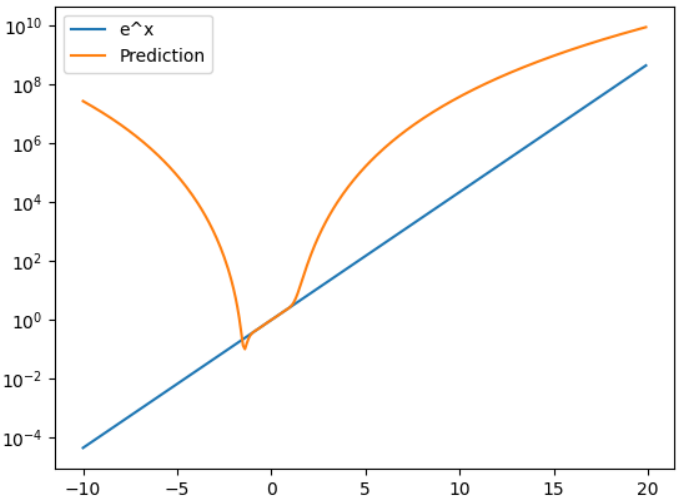
First, let's take a look at the first result.

```
losses = model.train(1000, 0.005)
graph_loss(losses)
model.print_params()
graph_pred()
```

```
Epoch: 100/1000, Loss: 0.00013234985876534958
Epoch: 200/1000, Loss: 3.427320321762344e-05
Epoch: 300/1000, Loss: 2.4492707827970274e-05
Epoch: 400/1000, Loss: 2.2669862503550263e-05
Epoch: 500/1000, Loss: 2.172205317266906e-05
Epoch: 600/1000, Loss: 2.0984248749352728e-05
Epoch: 700/1000, Loss: 2.036538265870468e-05
Epoch: 800/1000, Loss: 1.9834658817269138e-05
Epoch: 900/1000, Loss: 1.937270927247297e-05
Epoch: 1000/1000, Loss: 1.8965386261105253e-05
```




```
w1: 0.9444454598562942, Expected: 1.0
w2: 0.49986306356857685, Expected: 0.5
w3: 0.620431907367758, Expected: 0.16666666666666666
w4: 0.21618569290559692, Expected:
    0.041666666666666664
w5: -0.9321256839917001, Expected:
    0.008333333333333333
w6: -0.48709437200615985, Expected:
    0.001388888888888889
w7: 0.5594822789317027, Expected:
    0.0001984126984126984
w8: 0.33002989302732855, Expected: 2.48015873015873e
    -05
b: 0.9983077779493585, Expected: 1
```



This is what we got after the first training. The loss looks pretty good actually. As you can see for each 100^{th} epoch, the loss is decreasing well. From the graph, you can see the sudden drop in losses; however, we can't necessarily say that this is a grokking phenomena as it might just be model understanding the pattern from underfitting.

Looking at the weights, some are good and some are bad. We will get back to this later, but this is also one of the reason why we cannot scale this engine. Similarly, we can see that our polynomial fits very well for the small range, but not well as the input increases.

```
model.test()
```

```
Test 1                      Test 2
y:     2.718281828459045    y:     2.7456010150169163
ypred: 2.7495260176087557   ypred: 2.7867851769675642

Test 3                      Test 4
y:     2.7731947639642978   y:     2.801065834699079
ypred: 2.8257757734623574   ypred: 2.8666322590606814
```

```
Test 5                      Test 6
y:     2.82921701435156     y:     2.857651118063164
ypred: 2.909497162739583    ypred: 2.954521430346852

Test 7                      Test 8
y:     2.8863709892679585   y:     2.915379499976997
ypred: 3.001864774074873    ypred: 3.0516960317129964

Test 9                      Test 10
y:     2.944679551065524    y:     2.9742740725630656
ypred: 3.1041935358457007   ypred: 3.159545493165112

Test 11                     Test 12
y:     3.0041660239464334   y:     3.034358394435676
ypred: 3.2179503740678106   ypred: 3.2796173127071624

Test 13                     Test 14
y:     3.0648542032930024   y:     3.095656500124711
ypred: 3.3447665176737535   ypred: 3.4136296934778425

Test 15                     Test 16
y:     3.1267683651861553   y:     3.158192909689767
ypred: 3.4864504730090555   ypred: 3.5634848611499024
...
Test 45                     Test 46
y:     4.220695816996552    y:     4.263114515168817
ypred: 9.347554553339133    ypred: 9.752958544358306

Test 47                     Test 48
y:     4.305959528345206    y:     4.349235141062741
ypred: 10.179646340417854   ypred: 10.628580569357089

Test 49                     Test 50
y:     4.392945680918757    y:     4.437095519003664
ypred: 11.100758361572902   ypred: 11.597212287422602
```

For the training set, I only listed some of them as you can refer to the previous pages. Notice that for first few tests, the predictions are not bad.

However, for Test 5, the engine does not seem be effective and seems lost from Test 10. We could actually run our training loop multiple times to see what happens, and I will leave this to the readers. However, the engine does not seem to learn well.

Here is the question. Is the engine overfitting right now? If you said no, well... that is correct! Technically, we can see it as overfitting, but what I think is more accurate is to say that this is the limitation of the scalability of this engine and our method. When predicting, we only used nine terms. From the parameters, some were very close to the expectation and others were very bad. This is because we are using a polynomial with finite term to predict, whereas the real Maclaurin series uses infinite sum. Therefore, it is techinically an overfitting to certain point in a way that our engine tries to fit our polynomial in limited range as shown in the graph, but this too could improve if we have the computational ability to use infinite sum. Therefore, it may be more accurate to say that it is the limitation of the engine and scalability that causes high loss in testing.

We just completed our first model! Although it is very limited in terms of scalability and usability, I hope this example was very helpful in using backpropagation in code. For our next example, we will build an actual simple neural net to predict a nonlinear function.

23.3.2 Vanilla Neural Network

Now that we have created a simple backprop engine, let's increase the number of hidden layers to predict $x_1 x_2$ for inputs x_1 and x_2. I thought it would be great to introduce nonlinearity to our problem, and it works! Let's started with detailed explanations of the entire notebook available here.

Detailed Explanation

Now let's take a look at each block of the code to see their role in our vanilla neural net.

```
import math
import random
import matplotlib.pyplot as plt
%matplotlib inline
```

Here, I imported essential modules and libraries to do math, get a randomized value, and plot the loss.

```
# Predicting x1 * x2
def tanh(x):
    return (math.exp(x) - math.exp(-x)) / (math.exp
        (x) + math.exp(-x))

def tanh_der(x):
    return 1 - tanh(x)**2

def loss(y, ypred):
    return 0.5 * (y - ypred)**2
```

For this problem, I decided to use tanh for our activation function as sigmoid squishes the values to 0 and 1, which is not ideal for regression problem like ours and ReLU was not effective when I tried. Since $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ and $\tanh'(x) = 1 - \tanh^2(x)$, we did the same. For our loss function, I decided to use MSE for simplicity again.

```
# Data
data_x = [[i/10, j/10] for i in range(-10, 11) for
    j in range(-10, 11)]
data_y = [(x1 * x2) for x1, x2 in data_x]

data = list(zip(data_x, data_y))
random.seed(3141592)
random.shuffle(data)
```

```
8
9  split = int(0.8 * len(data))
10 train_data = data[:split]
11 test_data = data[split:]
12
13 train_x = [x for x, y in train_data]
14 train_y = [y for x, y in train_data]
15 test_x = [x for x, y in test_data]
16 test_y = [y for x, y in test_data]
```

This part is for making our datasets. I decided to divide i and j by 10 because if I don't, math overflow error will be resulted due to tanh function.

This time, I decided to shuffle data in a way that we will get the same result when you try running this, and split the data into 80% for training and 20% for validating.

```
1  class MLP:
2      def __init__(self, layers):
3          self.ws = []
4          self.bs = []
5
6          random.seed(3141592)
7          for i in range(len(layers) - 1):
8              layer_in = layers[i]
9              layer_out = layers[i + 1]
10             layer_w = [[random.uniform(-1, 1) for _
                   in range(layer_in)]
11                            for _ in range(layer_out)]
12             layer_b = [random.uniform(-1, 1) for _
                  in range(layer_out)]
13
14             self.ws.append(layer_w)
15             self.bs.append(layer_b)
```

Again for our example, I decided to store our core functions in a class since we want to reuse and store our attributes. When this class is first called, we will create an empty array and for `ws` and `bs` and store them in `self`. Then, for the randomized value, we will loop through each layers to store a randomized value between -1 and 1 for weights and biases. Notice that unlike the bias which only loop through the neurons in the output layer, we have to loop through the previous layer too for weights since we have different weights for each previous neuron for each output neuron. We will then append, or store, our layer to `ws` and `bs`.

```
def forward(self, x):
    self.act = [x]
    self.zs = []

    # Layers
    for i in range(len(self.ws)):
        prev = self.act[-1]
        pre_layer = []
        act_layer = []

        # Neurons
        for j in range(len(self.ws[i])):
            pre = self.bs[i][j]
            for k in range(len(self.ws[i][j])):
                pre += self.ws[i][j][k] * prev[k]

            pre_layer.append(pre)

            if i != len(self.ws) - 1:
                act_layer.append(tanh(pre))
            else:
                act_layer.append(pre)

        self.zs.append(pre_layer)
        self.act.append(act_layer)
```

```
26
27      return self.act[-1][0]
```

This is the forward function. Here, I decided to create both `self.act` and `self.zs` since we need both of them for backprop function. As the name suggests, `self.act` is the value after tanh is applied to `self.zs`. Moreover, I stored the input in `self.act` in the beginning to start using them in the first hidden layer.

Continuing, we create arrays for each layer for preactivation and activation. Then, we will continue by looping through the neurons within each layer. Our preactivation will start from the bias, and by following the definition, we will add the products of weights and the previous activation. For the if-else condition, I decided to do this because we don't want tanh for our final output layer. Then, we will store the layers in `self.act` and `self.zs` for latter use.

```
1  def backprop(self, y):
2      self.grad_w = [[[0.0 for _ in range(len(self.ws
           [i][j]))]
3                      for j in range(len(self.ws[i]))
                           ]
4                     for i in range(len(self.ws))]
5      self.grad_b = [[0.0 for _ in range(len(self.bs[
           i]))]
6                     for i in range(len(self.bs))]
7      ypred = self.act[-1][0]
8      delta = [0] * len(self.ws)
9      delta[-1] = [ypred - y]
10
11     # range(start, stop, step)
12     for k in range(len(self.ws) - 2, -1, -1):
13         delta[k] = []
14         for i in range(len(self.ws[k])):
15             dlda = 0.0
16             for j in range(len(self.ws[k + 1])):
```

```
                dlda += self.ws[k + 1][j][i] *
                    delta[k + 1][j]
            delta[k].append(dlda * tanh_der(self.zs
                [k][i]))

    for i in range(len(self.ws)):
        prev = self.act[i]
        for j in range(len(self.ws[i])):
            self.grad_b[i][j] = delta[i][j]
            for k in range(len(prev)):
                self.grad_w[i][j][k] = delta[i][j]
                    * prev[k]
```

Ah the backprop. I think this is the hardest part from our code. Assuming that you have read the previous section, let's interpret the code with the corresponding math equations.

Recall although we do not calcuate the loss per individual parameter, we do have to calculate the gradients. To start, I decided to put the initial values as 0. To store them, we can do the same just as we randomized the initial weights and biases because ultimately, each gradient will have a matching parameter. Continuing, `ypred` is the prediction, and that is the first element from the last activation layer. This will vary when we have numerous outputs, but since this is a simple regression problem and we only have one output, this is good. `delta` here is the delta term for our hidden layers. Let's start by setting them as 0 at first and `ypred - y` for the last layer.

Here comes the fun part. In Python, the `range` function can be used as `range(start, stop, step)`. Here, we start from the last hidden layer and we subtract 2 due to indices. Then, we will stop when we reach -1 by going backward by 1. We will loop through them for our delta terms. Recall the formula that we derived below.

$$\delta_i^{[l]} = \frac{\partial L}{\partial a_i^{[l]}} \frac{\partial a_i^{[l]}}{\partial z_i^{[l]}} = g'\left(z_i^{[l]}\right) \sum_j w_{ji}^{[l]} \delta_j^{[l+1]}$$

As the formula suggests, we will add the weights and the corresponding following delta term with the for loop for $\frac{\partial L}{\partial a_i^{[l]}}$. Then, we will mulitply it by the derivative of the actiavtion function, which is tanh for our example. Finally, we can store our delta terms that are calculuated. Next, we can now use them to calculate the gradients.

$$\frac{\partial L}{\partial w_{ij}^{[l]}} = \delta_i^{[l]} a_j^{[l-1]}$$

$$\frac{\partial L}{\partial b_i^{[l]}} = \delta_i^{[l]} \frac{\partial z_i^{[l]}}{\partial b_i^{[l]}} = \delta_i^{[l]}$$

As the formula suggests, for each parameters, will calculate the gradients again with for the loop. This is it for the backprop!

```
def update(self, lr):
    for i in range(len(self.ws)):
        for j in range(len(self.ws[i])):
            for k in range(len(self.ws[i][j])):
                self.ws[i][j][k] -= lr * self.
                    grad_w[i][j][k]
            self.bs[i][j] -= lr * self.grad_b[i][j]

def train(self, train_x, train_y, epochs, lr):
    losses = []
    for epoch in range(epochs):
        total_loss = 0

        for x, y in zip(train_x, train_y):
            ypred = self.forward(x)
            total_loss += loss(y, ypred)
            self.backprop(y)
            self.update(lr)

        average_loss = total_loss / len(train_x)
        losses.append(average_loss)

```

```
22          if (epoch + 1) % 100 == 0:
23              print(f"Epoch: {epoch+1}/{epochs}, Loss
                    : {average_loss}")
24
25      return losses
```

Now we need to update. This part is similar with our previous example, but for each parameter, we will update it by subracting the product of the corresponding gradient and the learning rate. The training part is again very similar to our previous example. We will begin our loss with 0, the we will predict, accumulate the loss, backprop, update, and print every hundred epoch.

```
1       def show_params(self):
2           for i, (w, b) in enumerate(zip(self.ws,
                self.bs)):
3               layer = "OUTPUT" if i == len(self.ws) -
                    1 else f"LAYER {i+1}"
4               print(layer)
5               for j, neuron_w in enumerate(w):
6                   weights_str = ", ".join(f"{x:.4f}"
                        for x in neuron_w)
7                   print(f"  Neuron {j+1}:")
8                   print(f"    Weights: [{weights_str
                        }]")
9                   print(f"    Bias = {b[j]:.4f}")
10              print()
11
12  mlp = MLP([2, 8, 8, 1])
```

This part is for printing the parameters, and as you can see, it is pretty straight forward. I ended up not using it because I don't think it is that effective at this point, but why not just have it. Now we create an instance for the MLP class with two hidden layers with eight neurons each.

```
losses = mlp.train(train_x, train_y, epochs=2000,
    lr=0.01)
print()

for x, y in zip(train_x[:10], train_y[:10]):
    print(x, y)
    print(f"     ypred: {mlp.forward(x)}")

plt.plot(losses)
plt.xlabel("epoch")
plt.ylabel("loss")
plt.show()

#mlp.show_params()
```

We will train with the learning rate of 0.01 and 2000 epochs. We then print the first ten p9redictions and plot the loss.

```
test_loss = 0
for x, y in zip(test_x, test_y):
    ypred = mlp.forward(x)
    test_loss += loss(y, ypred)

for x, y in zip(test_x[:10], test_y[:10]):
    ypred = mlp.forward(x)
    print(x, y)
    print(f"    ypred: {ypred}")

print("Test Loss:", test_loss / len(test_x))
```

This part is for printing the loss and prediciton for testing. That was it for the explanation! Now let's take a look at the results.

Analyzing the Results

Now that we have looked at the source code, let's analyze the result to see how our vanilla neural net performs.

```
losses = mlp.train(train_x, train_y, epochs=2000,
    lr=0.01)
print()

for x, y in zip(train_x[:10], train_y[:10]):
    print(x, y)
    print(f"     ypred: {mlp.forward(x)}")

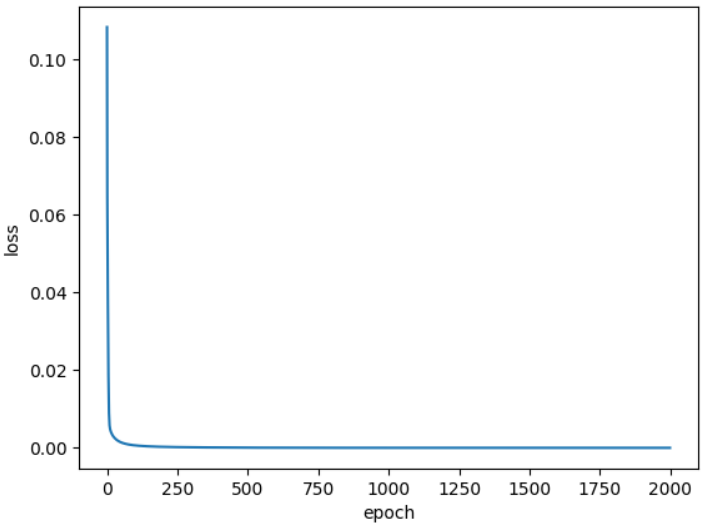
plt.plot(losses)
plt.xlabel("epoch")
plt.ylabel("loss")
plt.show()

#mlp.show_params()
```

```
Epoch: 100/2000, Loss: 0.0006795237745380497
Epoch: 200/2000, Loss: 0.0002780199905385479
Epoch: 300/2000, Loss: 0.00015780133554953304
Epoch: 400/2000, Loss: 9.996884725589866e-05
Epoch: 500/2000, Loss: 6.544552166502369e-05
Epoch: 600/2000, Loss: 4.373246439583505e-05
Epoch: 700/2000, Loss: 3.032175895325618e-05
Epoch: 800/2000, Loss: 2.2295750007135315e-05
Epoch: 900/2000, Loss: 1.7559747007674843e-05
Epoch: 1000/2000, Loss: 1.4723606896641237e-05
Epoch: 1100/2000, Loss: 1.2951440585334302e-05
Epoch: 1200/2000, Loss: 1.1773900112740833e-05
Epoch: 1300/2000, Loss: 1.0936643742787753e-05
Epoch: 1400/2000, Loss: 1.0302935457567813e-05
Epoch: 1500/2000, Loss: 9.79811628836463e-06
Epoch: 1600/2000, Loss: 9.379957580934361e-06
Epoch: 1700/2000, Loss: 9.023360655331247e-06
Epoch: 1800/2000, Loss: 8.712547584218995e-06
```

```
Epoch: 1900/2000, Loss: 8.437042172730164e-06
Epoch: 2000/2000, Loss: 8.189547717119376e-06

[1.0, -1.0] -1.0
     ypred: -0.9914314084230468
[0.8, -1.0] -0.8
     ypred: -0.8026975789984329
[0.8, -0.9] -0.7200000000000001
     ypred: -0.7229659139930072
[0.1, -0.8] -0.08000000000000002
     ypred: -0.07901899113208088
[0.3, -0.3] -0.09
     ypred: -0.08931568886644314
[-0.7, -0.1] 0.06999999999999999
     ypred: 0.07003157242570901
[-0.6, -0.5] 0.3
     ypred: 0.29863205953269367
[0.7, 0.8] 0.5599999999999999
     ypred: 0.5613061391881997
[-0.5, -0.4] 0.2
     ypred: 0.19629911113434118
[0.9, 0.8] 0.7200000000000001
     ypred: 0.7218073084017129
```



I would say this is pretty good! The loss is constantly decreasing and the prediction seems to be pretty good. Looking at the graph, the model seemed to have little idea at first, but instantly learned to see the holistic view as suggested by the sudden drop.

```
test_loss = 0
for x, y in zip(test_x, test_y):
    ypred = mlp.forward(x)
    test_loss += loss(y, ypred)

for x, y in zip(test_x[:10], test_y[:10]):
    ypred = mlp.forward(x)
    print(x, y)
    print(f"    ypred: {ypred}")

print("Test Loss:", test_loss / len(test_x))
```

```
[-0.9, -0.3] 0.27
    ypred: 0.27224246623233583
[0.9, 1.0] 0.9
    ypred: 0.889162612694173
```

```
[0.1, 0.5] 0.05
    ypred: 0.050824340568330406
[1.0, 1.0] 1.0
    ypred: 0.9737260427905209
[0.6, -0.8] -0.48
    ypred: -0.4811987339447982
[0.6, -0.3] -0.18
    ypred: -0.17587974783858806
[0.6, 0.4] 0.24
    ypred: 0.23577905969192914
[0.5, -0.8] -0.4
    ypred: -0.4007379739663608
[0.5, 0.2] 0.1
    ypred: 0.0970149401772119
[0.9, -0.2] -0.18000000000000002
    ypred: -0.18071247280108016
Test Loss: 1.1432408362631248e-05
```

Our test also seems pretty good. The loss seems small because our validation set uses small numbers, but we could see that the prediction somewhat matches the real values!

Now looking back at our model, this is again good for the limited range as suggested by the result above. However, there is a reason for that. Recall that we shuffled the dataset and seletcted 80% for training and 20% for testing there. Therefore, for the limited range that the model is trained in, the model did pretty good job. However, when I first tried making dataset by splitting the training and validation as first 80% and latter 20%, our model did poorly. This is because the model learned the left side well, but it does poorly in predicting the right side since it has no idea what's going on there with absolutely no training example. However, the good news is that our neural net is scalable unlike the previous example! By increasing the range for datasets, and could see whether the model underfits or overfits.

We now covered two models from scratch! Now that we know what happens in deep neural nets, let's build our own language models!

Note 24

Building Language Models

From the third note, we discussed foundations and intuitions on language models. In our previous note, we discussed about in-depth about neural networks. Before going into different topics including AI safety, I wanted to build language models because there are a lot of concepts to discuss there. First, using our knowledge, let's build a bigram. Then, let's build a more sophisticated language model that actually sounds like a chatbot with the introduction of transformers.

§24.1 Bigram Language Model

§24.2 Attention and Transformers

Finally! Its time to discuss about transformers. If you have not already, please complete the playlist by 3Blue1Brown's highlighted in the last note and read Attention Is All You Need. More great resources can be found in the last note. *Attention Is All You Need* is considered one of the most important paper in computer science and behind its meme-like title, it makes current LLMs possible.

Although the transformer architecture is used in many places, I want to introduce it LLMs as I think it is most intuitive and classic. The problem with prior language models is that they can't remember what they were talking about seconds ago. Because the next token is predicted with limited context window, it is not possible to "remember" the entire context like us. To improve this, eight researchers from Google introduced transformers. Although the basic ideas existed before the paper, it made the transformers well known.

The content itself, when you think about it, is quite simple and very intuitive. What should we do to enhance the use of information that we have? The answer is quite simple and circular. Literally use more prior contexts with efficient methods. Think of transformer blocks, or layers, as a function that takes in and spits out vectors of the same shape. The input vector goes in to a magical concept called multi-head attention, add and norm, and feed forward (which is basically a fancier word for MLPs that we discussed so far).

§24.3 Building an LLM with PyTorch

Note 25

AI and The Future

§25.1 AI Safety

Note 26

Further Resources

Throughout learning, I was blessed with countless resources dedicated for machine learning. Therefore, I decided to list recommended readings and other resources I used when studying and writing the notes. A huge shout out to creators for their contributions in machine learning community!

Papers

- **All Time Favorites**
  - Attention Is All You Need
  - A Mathematical Framework for Transformer Circuits
- **LLMs**
  - Chain-of-Thought Prompting Elicits Reasoning in Large Language Models
  - Chain-of-Thought Reasoning Without Prompting
  - Language Models are Few-Shot Learners
  - Scaling Laws for Neural Language Models
  - A Neural Probabilistic Language Model
  - Large Language Models: A Survey
- **Diffusion Models**

  - Hierarchical Text-Conditional Image Generation with CLIP Latents
  - Score-Based Generative Modeling through Stochastic Differential Equations
  - Denoising Diffusion Probabilistic Models
  - Learning Transferable Visual Models From Natural Language Supervision

Websites

- **Articles and Blogs**
  - Colah's Blog (Christopher Olah)
  - When AI builds itself (Anthropic)
  - Deep Double Descent (OpenAI)
  - Visualizing K-Means Clustering (Naftali Harris)
  - Jason Osajima (Jason Osajima)
  - ML Foundations: Understanding the Math Behind Backpropagation (Jordan Coblin)
- **Miscellaneous**
  - MLU-EXPLAIN (Machine Learning University)
  - FrontierMath
  - Machine Learning Mastery
  - xKiwiLabs
  - Distill
  - PyTorch Tutorial
- **GitHub**
  - ML-From-Scratch (Erik Linder-Norén)

Books and Notes

- **Books**
  - Goodfellow's Book
  - Dive into Deep Learning
  - Neural Networks and Deep Learning (Michael Nielsen)
  - Mathematics for Machine Learning

  - Deep Learning with PyTorch
  - Foundations of Computer Vision (Antonio Torralba, Phillip Isola, William Freeman)
- **Notes**
  - CS229 (Stanford University)
  - Sparse Autoencoder (Andrew Ng)

Courses and Videos

- **Courses**
  - Practical Deep Learning
  - CME 295 (Stanford University)
- **Videos**
  - Neural Networks (3Blue1Brown)
  - Neural Networks: Zero to Hero (Andrej Karpathy)

Tools and Datasets

- **Tools**
  - Weights & Biases
  - OpenRouter
- **Datasets**
- **Platforms**
  - Cirleback
  - Fireflies

Part V

Competitive Programming

Contents

Introducing the Notes **489**

27 C++ Basics **491**
27.1 Foundations 491
27.1.1 Variables 494
27.2 Conditionals and Loops 496
27.2.1 Conditionals 497
27.2.2 Loops 502
27.3 Functions and References 507
27.3.1 Functions 508
27.3.2 References 513
27.4 Arrays and Vectors 517
27.4.1 Arrays 517
27.4.2 Vectors 519
27.5 Practice Problems and Solutions 522

28 Foundations **525**
28.1 Time Complexity 525
28.2 Simulation and Complete Search 528
28.2.1 Simulation 528
28.2.2 Complete Search 535
28.3 Practice Problems and Solutions 535

Introducing the Notes

TBW

Note 27

C++ Basics

Before learning algorithms and solving problems, we need to choose a language to use. There are many programming languages that we can choose from for competitive programming such as C++, Python, and Java. However, many competitors use C++ due to its speed and popularity [Laa19]. I was debating whether to learn C++ or just stick with Python that I am already familiar with; however, I decided to learn C++ because it never hurts to learn a new language and it is also a recommended language [USA26]. With that in mind, let's get started with basic syntax from C++!

§27.1 Foundations

Let's start with the Hello World example. However to do so, we need a text editor or an integrated development environment (IDE) and a compiler. There are many different options for text editor or IDE. There are online editors that lets you write and compile there. However, I highly recommend using local editors and compilers for maintainability. Among many local editors, some popular choices are Visual Studio Code, CLion, and Vim. Please don't get mad for not including your favorite

editor as it could get very philosophical. I personally use Neovim due to its low learning curve and ease in set up. Choosing an editor is up to you!

For compilers, two popular options are GNU Compiler Collection (GCC) and Clang. I personally use Clang, but this too is up to you! If you have an editor and a compiler of your choice, let's get started with printing "Hello, World!" on our screen.

Code 27.1.1: CS0201.26050-01

```
#include <iostream>

int main() {
    std::cout << "Hello, World!\n";
    return 0;
}
```

I know this looks a lot, but let's first compile and see it before breaking down each line. To compile this, we can run

```
$ clang++ program.cpp -o program
```

and execute it with the following commend.

```
$ ./program
```

Then, you should see the following output in your terminal.

Output 27.1.2

```
Hello, World!
```

Now let's break down the program line by line. First, `#include <iostream>` is for including the standard input and output streams library. We need this to print our output on the screen.

Next, `int main()` defines our `main` function for execution. Next, `std::cout << "Hello, World!\n";` is to `cout`, or character out, from the standard library specified with the namespace `std`. We also

have `\n` to add a line break at the end to make a pretty output. Finally, don't forget your semicolon! We need to tell our compiler that our statement has ended with the semicolon. Meaning, unlike Python, indentation is only for writing a legible code.

Lastly, `return 0;` is used to tell that the program has successfully terminated. Nowadays, it isn't strictly necessary as reaching the end of the `main` function automatically returns 0. However, it's still a good habit to keep them at the end of the `main` function.

Below is an example with many outputs.

Code 27.1.3: CS0201.26050-02

```
#include <iostream>
using namespace std;

int main() {
    cout << "I love banana. " << "Banana likes me" <<
       endl;
    cout << "I ate five bananas today\n";

    return 0;
}
```

Compiling and executing returns the following.

Output 27.1.4

```
I love banana. Banana likes me
I ate five bananas today
```

Notice that we can add `using namespace std;` to avoid writing `std::` every time we need `cout`. We can do this for small projects and speed in competitive programming. However, this is highly discouraged in header files for large projects as it causes collisions. One more thing to note is that both `\n` and `endl` returns a new line. However, they are **not** the same thing! We will return to this later when we discuss about optimization.

27.1.1 Variables

We know how to print a value. How about storing them? Let's first start with definitions.

Definition 27.1.5

A region of a storage that a hardware can store a value is called an **object**. **Variables** are objects with a name. The **data type** is the type of value that can be stored in the object [Pom24].

There are many data types and we need to specify the data type before declaring a variable. Below are some examples of basic data types.

Types	Stores...
int	Integers
long long	Larger 64-bit integers
float	4-bit decimals
double	8-bit decimals
char	Single character
string	Series of characters
bool	True or false value

With the data type in mind, we can declare a variable as the following.

```
dataType variableName = value;
```

Below is a quick example of declaring a variable and printing the value.

Code 27.1.6: CS0201.26050-03

```
#include <iostream>

int main() {
    std::string name = "Enoch";
    int numBananas = 12 + 34;

    std::cout << "Hi! I am " << name << ", and I ate "
              << numBananas << " bananas today.\n";
```

```
9
10      return 0;
11  }
```

Notice that it is important to use quotation when declaring a string, and we can use simple operations when declaring a variable. This time, let's take a look at another example where we can take in the value of a variable and declare variables of the same type.

Code 27.1.7: CS0201.26050-04

```
1  #include <iostream>
2
3  int main() {
4      int a, b;
5
6      std::cout << "Type two numbers to add: ";
7      std::cin >> a >> b;
8      std::cout << "Your sum is: " << a + b << std::endl;
9
10     return 0;
11 }
```

When you compile and run it, we can get a simple calculator that can add two numbers with adequate size (32-bit).

Output 27.1.8

```
Type two numbers to add: 12345 67890
Your sum is: 80235
```

If you don't want the value of a to change, we can use `const`.

Code 27.1.9: CS0201.26050-05

```
1  #include <iostream>
2
3  int main() {
4      const int a = 5;
5      int b;
6
```

```
    std::cout << "Type a number: ";
    std::cin >> b;
    std::cout << b << " + 5 = " << a + b << std::endl;

    return 0;
}
```

In this variation, a is fixed as 5 and cannot be changed.

Output 27.1.10

```
Type a number: 123456789
123456789 + 5 = 123456794
```

One more thing to keep in mind about inputs is that some competitions and older USACO takes file as an input. In that case, we can do the same by using this template.

```
#include <cstdio>
#include <iostream>

int main() {
    freopen("inputFileName.in", "r", stdin);
    freopen("outputFileName.out", "w", stdout);

    // Your code
}
```

Now that we can take inputs and print outputs with various data types, let's discuss about loops and conditions.

§27.2 Conditionals and Loops

Before getting started with loops, let's briefly go over increments. For our counter i, we can **increment** or **decrement** the value, or increase or decrease, by 1 with `i++` and `i--`.

For instance if we do the following,

```
i = 5
i++
```

then it is equivalent of the following statements.

```
i = 5
i = i + 1
i = 6
```

Similarly, `i--` is identical to saying `i = i - 1`. There is also `++i` and `--i`, which first increments or decrements the value before defining, but it is just better to adjust index and use `i++` and `i--` to avoid confusion unless there is a specific reason to use them.

27.2.1 Conditionals

One more thing to note before we move on is logical operations. Unlike how we would usually write in math, we cannot use $=$ and $\neq$ for equal to and not equal to. We also won't be able to use comparisons $\geq$ and $\leq$. Instead, we would be using == for equal to, != for not equal to, <= for less than or equal to, and >= for greater than or equal to. This is crucial when we have to compare two values in different statements. With these in mind, let's get started with if statements!

If-Else Statements

What if you want to display different statements depending on the user input? Fortunately, we could achieve that through if-else statements without manual human inspections. If, else, if-else statements have the following forms.

```
if (condition) {
    body
} else if (condition) {
    body
} else {
    body
```

```
}
```

Let's take a look at a quick example.

Code 27.2.1: CS0201.26051-01

```
#include <iostream>

int main() {
    int a;

    std::cout << "Enter a number: ";
    std::cin >> a;

    if (a > 10) {
        std::cout << "Your number is greater than ten!\n
            ";
    } else if (a == 10) {
        std::cout << "Your number is ten!\n";
    } else {
        std::cout << "Your number is smaller than ten!\n
            ";
    }

    return 0;
}
```

In this short example, we compare an input from the user with 10. As you can see, we must use `a == 10`, not `a = 10`, as it would be declaring a as ten. We can see that our program works as expected!

Output 27.2.2

```
$ ./program
Enter a number: -1
Your number is smaller than ten!

$ ./program
Enter a number: 10
Your number is ten!
```

```
$ ./program
Enter a number: 11
Your number is greater than ten!
```

Instead of comparing numbers, we could also use statements for comparing strings.

Code 27.2.3: CS0201.26051-02

```
#include <iostream>

int main() {
    std::string password = "password";
    std::string input;

    std::cout << "Enter your password: ";
    std::cin >> input;

    bool correct = (input == password);

    if (correct) {
        std::cout << "Correct!\n";
    } else {
        std::cout << "Incorrect!\n";
    }

    return 0;
}
```

As you can see from the example above we can take in the boolean value. An alternative way would be directly putting `input == password` in the condition for if without defining a new variable.

Output 27.2.4

```
$ ./program
Enter your password: password
Correct!
```

```
$ ./program
Enter your password: Password
Incorrect!
```

If you have fixed options, and you don't want to type every syntax for if-else statements, we can use **switch** statements.

Switch Statements

Switch statements have the following structure.

```
switch (expression) {
    case case1:
        block
    case case2:
        block
        break;
    default:
        block
}
```

Here, `break` and `default` are optional, but nice to have. Consider the following example for more intuitive explanation.

Code 27.2.5: CS0201.26051-03

```
#include <iostream>

int main() {
    char input;

    std::cout << "Choices for Breakfast:\n"
              << "  A. Two Bananas\n"
              << "  B. Three Bananas\n"
              << "  C. Four Bananas\n"
              << "What would you like for breakfast
                  today?\n";
    std::cin >> input;

```

```
13      switch (input) {
14          case 'A':
15              std::cout << "You chose two bananas!\n";
16              break;
17          case 'B':
18              std::cout << "You chose three bananas!\n";
19              break;
20          case 'C':
21              std::cout << "You chose four bananas!\n";
22              break;
23          default:
24              std::cout << "Please choose from the options
                    \n";
25      }
26
27      return 0;
28  }
```

Here, the program would go through all the options to see if there are any matching cases. If not, it will execute `default` as the fallback. If you also want to exit from the switch statement after finding the right case, we can do so with the `break` keyword.

Output 27.2.6

```
$ ./program
Choices for Breakfast:
  A. Two Bananas
  B. Three Bananas
  C. Four Bananas
What would you like for breakfast today?
B
You chose three bananas!

$ ./program
Choices for Breakfast:
  A. Two Bananas
  B. Three Bananas
```

```
  C. Four Bananas
What would you like for breakfast today?
D
Please choose from the options
```

With the basic ideas of conditions in mind, let's discuss different loops for C++.

27.2.2 Loops

There are many different fundamental loops that we can use in C++. First, let's start with for loops.

For Loops

For loops retain the following structure.

```
for (start; condition; update) {
    body
}
```

The introduction of loops really made our lives easier. One of the reason why we write program is to avoid repetitive tasks and work more efficiently. Say we want to print "I ate six bananas" five times on the screen. Without loops, we may have to use `cout` five times, which is ridiculous if the number increase to hundreds and thousands. Instead, we can use a for loop.

Code 27.2.7: CS0201.26051-04

```
#include <iostream>

int main() {
    for (int i = 0; i < 5; i++) {
        std::cout << "I ate six bananas\n";
    }

    return 0;
}
```

When we compile and run, it is telling the computer to print "I ate six bananas" if $i < 5$ starting from $i = 0$ where it increments by 1 after each pass. As expected, we will get the following result.

Output 27.2.8

```
I ate six bananas
I ate six bananas
I ate six bananas
I ate six bananas
I ate six bananas
```

A more practical use of for loops would be this.

Code 27.2.9: CS0201.26051-05

```
#include <iostream>

int main() {
    for (int i = 2; i <= 6; i++) {
        std::cout << "I ate " << i << " bananas\n";
    }

    return 0;
}
```

Instead of manually changing the numbers, the loop will display the following.

Output 27.2.10

```
I ate 2 bananas
I ate 3 bananas
I ate 4 bananas
I ate 5 bananas
I ate 6 bananas
```

With this, it is natural to consider for loops inside a for loop. This is called **nested loop** made with for loops. Consider the following example that generates all possible combinations of my breakfast assuming there exists three distinct bananas and two distinct apple where I must eat

exactly one banana and exactly one apple.

Code 27.2.11: CS0201.26051-06

```
#include <iostream>

int main() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
            std::cout << "Banana " << i+1 << " and "
                      << "Apple " << j+1 << std::endl;
        }
    }

    return 0;
}
```

This will run the inner loop when $i = 0$, $i = 1$, and $i = 3$. We also have to start from $i = 1$ and $j = 1$ or add the displayed number by 1, or else we will get Banana 0 and Apple 0. Compiling and running, we can notice that we get $3 \cdot 2 = 6$ distinct combinations.

Output 27.2.12

```
Banana 1 and Apple 1
Banana 1 and Apple 2
Banana 2 and Apple 1
Banana 2 and Apple 2
Banana 3 and Apple 1
Banana 3 and Apple 2
```

Moving on from for loops, let's discuss while loops.

While Loops

While loops have the following form.

```
while (condition) {
    body
}
```

There are many things that we can do with while loops. Below are two simple examples.

Code 27.2.13: CS0201.26051-07

```
#include <iostream>

int main() {
    int i = 10;

    while (i > 0) {
        std::cout << i << " ";
        i--;
    }
    std::cout << std::endl;

    return 0;
}
```

The code block inside the while loop will iterate until i reaches 1.

Output 27.2.14

```
10 9 8 7 6 5 4 3 2 1
```

Unlike the first example, the second example uses boolean for the condition.

Code 27.2.15: CS0201.26051-08

```
#include <iostream>

int main() {
    int birthdayYear = 9999;
    int input;
    bool wrong = true;

    while (wrong) {
        std::cout << "Guess which year I was born in!\n"
            ;
        std::cin >> input;
```

```
11
12          if (input == birthdayYear) {
13              wrong = false;
14              std::cout << "Correct!\n";
15          } else {
16              std::cout << ":(\n";
17          }
18      }
19
20      return 0;
21  }
```

This program will ask the user the same question with feedback until the boolean value `wrong` is false, or the answer is correct. Notice that it is important to put `std::cout << ":(\n";` in `else` to prevent it showing when the answer is correct.

Output 27.2.16

```
Guess which year I was born in!
1234
:(
Guess which year I was born in!
5678
:(
Guess which year I was born in!
9999
Correct!
```

Lastly, below is an introduction to another loop.

Do/While Loops

Do/While loop is a variation of while loop that always run the first time [W3S26]. Unlike the traditional while loop that always check the condition first before running, do/while loop run the first time, then check if next run is eligible even if the first pass does not meet the condition. Below is the general structure of do/while loops.

```
do {
    block
} while (condition);
```

Consider the following example.

Code 27.2.17: CS0201.26051-09

```
#include <iostream>

int main() {
    bool correct = false;

    do {
        std::cout << "Correct!\n";
    } while (correct);

    return 0;
}
```

With the traditional while loop, no outputs are expected. However, do/while loop will give a different result.

Output 27.2.18

```
Correct!
```

Notice that even if the condition is false for the first pass, the block still runs anyhow.

This is it for the basics of conditionals and loops! Before discussing fundamental algorithms, let's discuss more essential topics on functions and references.

§27.3 Functions and References

Up until now, we had everything in our `main` function. However, you may have noticed that dumping everything in there is not efficient nor scalable. This is where functions kick in.

27.3.1 Functions

As always, let's start with definition.

Definition 27.3.1

A reusable code block dedicated for a certain task is called a **function** [Gee26].

Functions are specially useful as we can use it multiple times just by calling them instead of copy and pasting every time we need them. Functions in C++ generally have the following format.

```
outputDataType functionName(parameters) {
    code block
    return outputValue;
}
```

Looking at a bare `main` function, we can see that we have `int` since the `main` function returns an integer value like 0. Since no arguments are taken in a bare `main` function, we have empty parenthesis. There are some functions that do not return any value. In that case, we would be using `void`. Moreover to use a function, we need to call it. Below is a quick example.

Code 27.3.2: CS0201.26052-01

```
#include <iostream>

void helloWorld() {
    std::cout << "Hello, World!\n";
}

int main() {
    helloWorld();
    helloWorld();
    helloWorld();
    return 0;
}
```

In the example above, we have our `helloWorld` function that prints the text "Hello, World!" to the console. The type is `void` since it does not return any values. By calling the function three times in `main`, we don't have to manually write long texts every time!

Output 27.3.3

```
Hello, World!
Hello, World!
Hello, World!
```

Below is a more practical use of simple functions for basic mathematical operations.

Code 27.3.4: CS0201.26052-02

```
#include <iostream>

int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}

int multiply(int a, int b) {
    return a * b;
}

double divide(double a, double b) {
    return a / b;
}

int mod(int a, int b) {
    return a % b;
}

int main() {
    std::cout << "1 + 2 = "     << add(1, 2)       << std
```

```
        ::endl;
25      std::cout << "3 - 4 = "    << subtract(3, 4) << std
            ::endl;
26      std::cout << "5 * 6 = "    << multiply(5, 6) << std
            ::endl;
27      std::cout << "7 / 8 = "    << divide(7, 8)   << std
            ::endl;
28      std::cout << "10 mod 9 = " << mod(10, 9)     << std
            ::endl;
29
30      std::cout << "((1 + 2) * 4) mod 10 = "
31                << mod(multiply(add(1, 2), 4), 10) << std
                      ::endl;
32
33      return 0;
34  }
```

Each function will take in the values a and b to perform the dedicated task. After declaring the functions, we can call them in the `main` function to display them. Many times it is better to make the functions declare a value rather than print directly to the console for reusability. As shown in the last example, we can call a function as an input of different function.

Output 27.3.5

```
1 + 2 = 3
3 - 4 = -1
5 * 6 = 30
7 / 8 = 0.875
10 mod 9 = 1
((1 + 2) * 4) mod 10 = 2
```

Another very important technique that allows us to break down hard problems into simpler problems is recursion.

Recursion

As always, let's get started with definition.

Definition 27.3.6

A technique of calling a function in the function itself is **recursion**.

The technique is rather straight forward when we look at some examples.

Exercise 27.3.7

Write a code that returns the sum of integers from 1 to n inclusive were n is any input.

Solution.

There are few ways of solving this problem. First, we can use a simple for loop. Consider the following program.

Code 27.3.8: CS0201.26052-03

```
#include <iostream>

int main() {
    int n;
    int sum = 0;

    std::cout << "Enter n: ";
    std::cin >> n;

    for (int i = 1; i <= n; i++) {
        sum += i;
    }
    std::cout << "Sum: " << sum << "\n";

    return 0;
}
```

In the code above, we use a simple loop without compound assignment operator to find the sum from 1 to any positive input n.

Output 27.3.9

```
Enter n: 3
Sum: 6
```

Below is another output.

Output 27.3.10

```
Enter n: 10
Sum: 55
```

Although this method works for simple example like this, we can also use recursion.

Code 27.3.11: CS0201.26052-04

```
#include <iostream>

int sum(int n) {
    if (n > 0) {
        return n + sum(n-1);
    } else {
        return 0;
    }
}

int main() {
    int n;

    std::cout << "Enter n: ";
    std::cin >> n;

    std::cout << "Sum: " << sum(n) << "\n";

    return 0;
}
```

In the code above, we have a function `sum` that returns `n + sum(n-1)`. We also need `return 0` as our base case. If $n = 5$, then it will return `5 + sum(4)` which continues as `5 + 4 + sum(3)`

until we have $5+4+3+2+1+0$. With recursion, we broke down a hard problem of adding a range of numbers to a simple problem of adding two numbers.

Output 27.3.12

```
Enter n: 3
Sum: 6
```

Below is another output.

Output 27.3.13

```
Enter n: 10
Sum: 55
```

As shown above, both methods work.

For a simple problem like the exercise above, the could be a more simple solution than recursion. However, the technique will come very handy when the problem gets harder. Now with the ideas of functions in mind, we can continue with another important topic of references.

27.3.2 References

One of the primary goals of writing codes for competitive programming is efficiency. Indeed most, if not all, codes written for competitions are not maintained as the main purpose is to solve problems efficiently [USA26]. Moreover, it is important that your program is fast enough to execute under the time limit. One way to achieve this is by preventing making of copies, and this is where reference comes in.

References and pointers serves different purposes, although they look similar. Indeed, references are using pointers behind the scenes. However there are much more topics to discuss for pointers, and let's focus on references for now for the purpose of this note.

Definition 27.3.14

A **reference** is an alias of an existing object that allows indirect access to the object.

Let's take a look at a quick example.

Code 27.3.15: CS0201.26052-05

```
#include <iostream>

int main() {
    int a = 5;
    int& ref = a; // reference to a

    std::cout << "ref: " << ref << "\n";
    std::cout << "a:   " << a   << "\n";
    ref++;
    std::cout << "After adding 1 to ref\n";
    std::cout << "ref: " << ref << "\n";
    std::cout << "a:   " << a   << "\n";

    return 0;
}
```

When declaring a reference, we use the ampersand &. Technically `int& ref` and `int &ref` does the same thing, but let's use the first one to be consistent. Here, `ref` is our reference to `a`. Unlike declaring `int ref = a;`, which creates a copy of `a`, reference become an alias of `a` that we can use to edit `a`. Below is the output from running the code.

Output 27.3.16

```
ref: 5
a:   5
After adding 1 to ref
ref: 6
a:   6
```

As you can see above, incrementing `ref` also increments `a` as they refer

to the same object. One of the most common application of reference is passing arguments in functions [Gee26]. Consider the example below.

Code 27.3.17: CS0201.26052-06

```
#include <iostream>

void increment1(int a) {
    a++;
    std::cout << a << "\n";
}

void increment2(int& a) {
    a++;
    std::cout << a << "\n";
}

int main() {
    int x = 5;
    increment1(x);
    x = 5;
    increment2(x);

    return 0;
}
```

In the example above, the difference in performance between the two functions are trivial. However, the first function creates a local copy of x while the second one doesn't. Passing arguments by reference can be especially helpful when we are dealing with large objects. Indeed, copying large objects won't be as efficient as just indirectly adjusting the original value.

Output 27.3.18

```
5
6
```

Another use case of reference in arguments is that we can directly change the value. Let's tweak our code above to further elaborate on the point

above.

Code 27.3.19: CS0201.26052-07

```
#include <iostream>

void increment1(int a) {
    a++;
}

void increment2(int& a) {
    a++;
}

int main() {
    int x = 5;
    increment1(x);
    std::cout << x << "\n";

    x = 5;
    increment2(x);
    std::cout << x << "\n";

    return 0;
}
```

This time, we have `cout` in the `main` function. As mentioned previously, the first function creates a local copy and adjust the copy. Meaning, the original values does not change. However, the second function changes the original value. Therefore, the values 5 and 6 must be the outputs.

Output 27.3.20

```
5
6
```

References are frequently used for efficiency and cleaner code. Before getting into exciting data structures and algorithms, let's discuss arrays and vectors.

§27.4 Arrays and Vectors

One of the most important and essential topic in computer science in general could be arrays and vectors and they provide a way to effectively store and manipulate data. Let's get started with arrays!

27.4.1 Arrays

A simple array of n elements can be thought of as an n-tuple.

Definition 27.4.1

An **array** is a collection of elements of the same data type.

When we declare an array, we need to specify the type, name, and the number of elements [W3S26].

```
dataType arrayName[number];
```

Consider the following example.

Code 27.4.2: CS0201.26053-01

```
#include <iostream>

int main() {
    int arr[5] = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        std::cout << arr[i] << " ";
    }
    std::cout << "\n";

    return 0;
}
```

In the example above, we have an array `arr` with five elements. When we actually print them to the console with a for loop, we can see that we have to start from `i = 0` since the index of the first element is 0. To

access an element, we can have the name of the array, bracket, and the index number.

Output 27.4.3

```
1 2 3 4 5
```

We can also edit an element by specifying it.

Code 27.4.4: CS0201.26053-02

```
#include <iostream>

int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    arr[2] = -3;
    std::cout << arr[2] << "\n";

    return 0;
}
```

The line `arr[2] = -3` will override the previous assignment.

Output 27.4.5

```
-3
```

Continuing, an array need not be 1-dimensional. There exists multidimensional arrays that are made of lower dimensional arrays [Mic26]. Below is an example of a 2×3 matrix.

Code 27.4.6: CS0201.26053-03

```
#include <iostream>

int main() {
    int arr[2][3];

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            arr[i][j] = (i + 1)*10 + (j + 1);
        }
```

```
    }

    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            std::cout << arr[i][j] << " ";
        }
        std::cout << "\n";
    }

    return 0;
}
```

In the example above, we declared a multidimensional array with two rows and three columns. Then, we used loops to assign a value in each index. Using the same way to put an input, we printed the values to the console. The index for the matrix should be familiar.

Output 27.4.7

```
11 12 13
21 22 23
```

From the examples above, you may have noticed that array is not really flexible in terms of number of elements. Sometimes we might want to add or remove elements from an array, but it won't allow that since the size is fixed. This is were vectors come to place.

27.4.2 Vectors

A simple way of thinking vectors is resizable array.

Definition 27.4.8

A **vector** is a dynamic container that contains elements of same the data type.

Vectors are provided by the Standard Template Library and we need to include directive to access them [Gee26]. Because vectors are dynamic, we only need the data type and name of the vector when declaring.

Output 27.4.9

```
std::vector<dataType> vectorName;
```

Below is a quick example of declaring a vector and two ways of printing the elements in the console.

Code 27.4.10: CS0201.26053-0

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {1, 2, 3, 4, 5};

    for (int i = 0; i < 5; i++) {
        std::cout << vec[i] << " ";
    }
    std::cout << "\n";

    for (int element : vec) {
        std::cout << element << " ";
    }
    std::cout << "\n";

    return 0;
}
```

From the example above, we declared a vector `vec` and printed the elements to the console in two different ways. Both methods are fine and the `:` symbol can be interpreted as "in."

Output 27.4.11

```
1 2 3 4 5
1 2 3 4 5
```

With the idea in mind, let's discuss few key functions that distinguishes a vector and an array.

The first function is `.push_back()` function. This function allows us to

add an element as the last element. If you wish to delete the last element, we can use the `.pop_back()` function. Lastly, `.size()` will allow us to see the size of a vector. Consider the following example.

Code 27.4.12: CS0201.26053-05

```
#include <iostream>
#include <vector>

int main() {
    std::vector<int> vec = {1, 2, 3, 4, 5};

    vec.push_back(6);
    for (int x : vec) {
        std::cout << x << " ";
    }
    std::cout << "\n";
    std::cout << vec.size() << "\n";

    vec.pop_back();
    for (int x : vec) {
        std::cout << x << " ";
    }
    std::cout << "\n";
    std::cout << vec.size() << "\n";

    return 0;
}
```

In this example, we declare a vector, add 6 to the end, list the elements, and find the size. Then, we will revert it to the original vector by removing the last element that was added.

Output 27.4.13

```
1 2 3 4 5 6
6
1 2 3 4 5
5
```

This is it for the C++ basics! There are so much more topics to discuss in C++, but for now, let's solve some practice problems before discussing more fundamental concepts in DSA.

§27.5 Practice Problems and Solutions

Below is the list of links to practice problems for this note and personal solutions.

1. **Source:** 2018 USACO Third Bronze
   Problem: Problem 1. Teleportation
   Solution: Enoch's Solution (Git)

2. **Source:** 2016 USACO Second Bronze
   Problem: Problem 1. Promotion Counting
   Solution: Enoch's Solution (Git)

3. **Source:** 2020 USACO First Bronze
   Problem: Problem 1. Do You Know Your ABCs?
   Solution: Enoch's Solution (Git)

4. **Source:** 2020 USACO Second Bronze
   Problem: Problem 1. Word Processor
   Solution: Enoch's Solution (Git)

5. **Source:** 2019 USACO US Open Bronze
   Problem: Problem 1. Bucket Brigade
   Solution: Enoch's Solution (Git)

6. **Source:** Codeforces
   Problem: Problem 4A. Watermelon
   Solution: Enoch's Solution (Git)

7. **Source:** Codeforces
   Problem: Problem 231A. Team
   Solution: Enoch's Solution (Git)

8. **Source:** Codeforces
   Problem: Problem 546A. Soldier and Bananas
   Solution: Enoch's Solution (Git)

Note 28

Foundations

Before getting into different techniques and methods, let's discuss fundamental topics including time complexity and basic DSA.

§28.1 Time Complexity

One of the primary goal of competitive programming and an effective program is efficiency. Indeed most, if not all, programming competitions have time limits, which varies for each language. Some problems can be solved with brute force approach while others need an elegant solution to fit in the given time frame.

Definition 28.1.1

The **Time Complexity** of a program denotes the number of operations an algorithm executes [Gee26].

Please note that the definition above is very basic, and not formal. A more formal definition involves theoretical computer science, and maybe I can include more on the formal definition and P = NP in other notes.

The approximate number of the operations for the worst-case is repre-

sented using the **big O notation**. The lower the complexity, the efficient the program is.

The six major complexities are Constant $\mathcal{O}(1)$, Linear $\mathcal{O}(n)$, Logarithmic $\mathcal{O}(n \log n)$, Quadratic $\mathcal{O}(n^2)$, Exponential $\mathcal{O}(2^n)$, and Factorial $\mathcal{O}(n!)$ [Ola22]. We will evaluate the time complexities of algorithms as we walk through them, but here are some simple examples using input/output and loops.

A simple hello world example have constant time complexity since it only prints once. One the other hand, the following snippet

```
for (int i = 0; i < 5*n; i++) {
    // constant time code
}
```

is $\mathcal{O}(n)$ [Laa19]. When we start nesting them, it gets horrible. For instance, the following snippet is $\mathcal{O}(nm)$.

```
for (int i = 0; i < n; i++) {
    for (int j = 0; i < m; j++) {
        // constant time code
    }
}
```

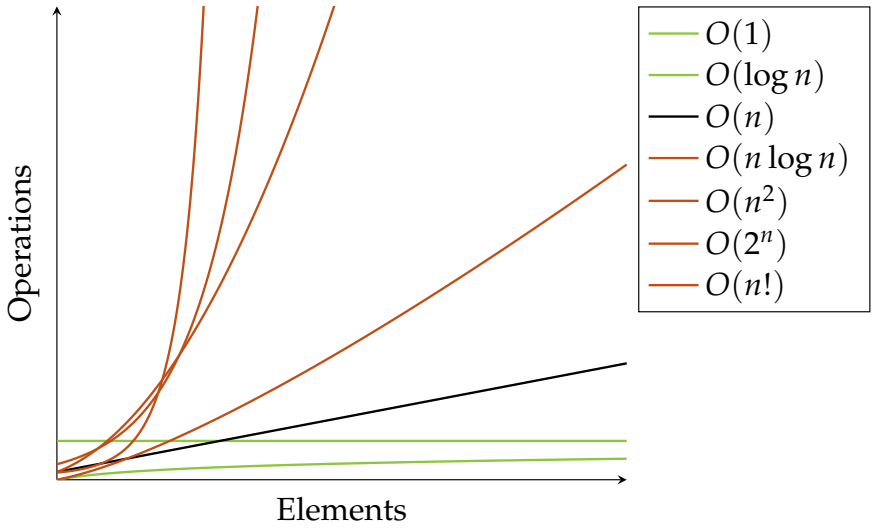
Naturally, we can see that we get quadratic time if we nest code as the following.

```
for (int i = 0; i < n; i++) {
    for (int j = 0; i < n; j++) {
        // constant time code
    }
}
for (int i = 0; i < 10000*n; i++) {
    // constant time code
}
```

The snippet above is $\mathcal{O}(nn) = \mathcal{O}(n^2)$. Note that the time complexity is

not linear since the big O is only interested in the worst case.

One more thing to note is the order of growth. Consider the graph below [Ola22].



From the graph above, we can see that some functions grow faster than the other. For instance, the constant time has the lowest growth rate since it is not growing at all. Next, we can see that $\log n$ grows slower than n, which grows slower than $n \log n$ [Gee26]. Generally speaking, anything below (n) is good while we really want to avoid anything beyond the linear time.

Now big O is important since it is used to compare two algorithms that perform the same task. With **asymptotic analysis**, we can evaluate and compare algorithms without the inconsistency of inputs and machines [Gee26]. This was a short introduction to computing time complexities and big O notation. In the next section, we will be discussing fundamental data structures.

§28.2 Simulation and Complete Search

The first topics that I wish to discuss are simulation and brute forcing. Surprisingly, many problems can be solved just by simulating the environment described in the problem and brute forcing all cases [Yao20]. Meaning, we do not necessarily require a complex algorithms to solve the problems and it suffices to know that basic of the language that we are using, which we already did in our previous note.

Because there isn't a definite concept, definition, and theorems for these concept, let's see what they are through example problems.

28.2.1 Simulation

For our first simulation problem, let's solve 2018 USACO First Bronze Problem 1.

Exercise 28.2.1: 2018 USACO First Bronze Problem 1

(Problem Statement)
Say there are three buckets with capacity c_1, c_2, and c_3 respectively. The buckets are initially filled with m_1, m_2, and m_3 amount of milk. A farmer wants to pour milk from bucket 1 to 2, 2 to 3, 3 to 1, 1 to 2, and so on for hundred times where each pour ends if the bucket being poured is empty or the bucket receiving the milk is full. How much milk is left in each bucket after the 100^{th} pour?

Solution.

To solve this problem, we can literally simulate the process of pouring and receiving milk. Notice that in each pour, we stop if and only if a bucket is empty or a bucket is full. Then, we iterate the same process. Due to this nature, we can define a function that takes in the amount of milk in two buckets and the capacity of the bucket receiving to update the amount of milk in each bucket after the pour.

```
void mix(int& a, int& b, int& x) {
    if (a + b <= x) {
        b = a + b;
        a = 0;
    } else {
        a = a + b - x;
        b = x;
    }
}
```

Here, a and b are the amount of milk in two buckets and x is the capacity of the bucket with b amount of milk. If the bucket receiving has the capacity to receive all milk, i.e. $a + b \leq x$, then our new b will be $a + b$ while the bucket with originally a amount of milk is empty. Notice that we want to update b first before fixing a to zero.

The other case is when we cannot pour all milk when mixing, i.e. $a + b > x$. In that case, our new a will be $a + b - x$ and $b = x$. Also notice that we want to update a first since we don't want $a = a$. Applying this function 100 times, we can successfully simulate the environment in the problem.

Code 28.2.2: CS0202.26057-01

```
#include <cstdio>
#include <iostream>
using namespace std;

void mix(int& a, int& b, int& x) {
    if (a + b <= x) {
        b = a + b;
        a = 0;
    } else {
        a = a + b - x;
        b = x;
    }
}

int main() {
    freopen("mixmilk.in", "r", stdin);
    freopen("mixmilk.out", "w", stdout);

    int c1, c2, c3, m1, m2, m3;
    cin >> c1 >> m1 >> c2 >> m2 >> c3 >> m3;

    for (int i = 0; i < 33; i++) {
        mix(m1, m2, c2);
        mix(m2, m3, c3);
        mix(m3, m1, c1);
    }
    mix(m1, m2, c2);

    cout << m1 << "\n" << m2 << "\n" << m3 << "\n";

    return 0;
}
```

This solves the problem!

As you can see from the example above, we simulate the environment described to solve the problem. Below is another example for simulation.

Exercise 28.2.3: 2017 USACO Open Bronze Problem 1

(Problem Statement)
A farmer and a cow is in a number line with position $x, y > 0$ respectively. The farmer will move in zig zag from x and doubling the distance from x in each move. For instance, the farmer moves from x to $x+1$ to $x-2$ to $x+4$ and so on. What is the total distance that the farmer moves until reaching the cow?

Solution.

To solve this problem, we can simulate the situation where the farmer is moving. However before doing that, let's shift the position of the initial position of the farmer and the cow by x to make our lives easier.

```
int z = y - x;
```

With this, the position of the farmer is 0 and the cow is z. Note that this is possible because the distance between them remains unchanged. From here, we can separate the cases into two case: $z < 0$ and $z > 0$. z can be negative since it is possible that $y < x$.

For integer $n \geq 0$, we can see that the farmer won't reach 2^{2n} for some n if $z > 0$ and the farmer will not reach 2^{2n+1} for some n if $z < 0$. Using this, we can know that the distance that the farmer will move if $z > 0$ can be represented as the following where n is the minimum value that the farmer will stop moving.

$$\begin{aligned} &1 + (1+2) + (2+4) + \cdots + \left(2^{2n-2} + 2^{2n-1}\right) + \left(2^{2n-1} + z\right) \\ &= 2\left(1 + 2 + 4 + \cdots + 2^{2n-1}\right) + z = 2\left(2^{2n} - 1\right) + z \end{aligned}$$

Similarly using geometric sequence, we can apply the same idea for $z < 0$.

$$1 + (1+2) + (2+4) + (4+8) \cdots + \left(2^{2n-1} + 2^{2n}\right) + \left(2^{2n} + z\right)$$

$$= 2\left(1+2+4+\cdots+2^{2n}\right)+z = 2\left(2^{2n+1}-1\right)+z$$

Applying it with C++, we can use the while loop to find when to stop and conditions to determine which value to output.

Code 28.2.4: CS0202.26057-02

```
#include <cstdio>
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    freopen("lostcow.in", "r", stdin);
    freopen("lostcow.out", "w", stdout);

    int x, y;
    cin >> x >> y;
    int z = y - x;

    int n = 0;
    if (z > 0) {
        while (pow(2,2*n) < z) {
            n++;
        }
        cout << 2*(pow(2,2*n) - 1) + z << "\n";
    } else if (z < 0) {
        while (pow(2,2*n+1) < -z) {
            n++;
        }
        cout << 2*(pow(2,2*n + 1) - 1) - z << "\n";
    }

    return 0;
}
```

This solves the problem!

Before wrapping up our discussion on simulation, let's solve one last

problem.

Exercise 28.2.5: 2016 USACO Third Bronze Problem 2

(Problem Statement)
A farmer has a circular barn with n rooms numbered 1 through n clockwise. Each room is connected with internal doors with the adjacent rooms and have a door to enter the room from the outside. The farmer wishes to place r_i cows in the i^{th} room and the farmer will open a single door to a room where cows will enter the room and find their places in clockwise and in organized manner. If the farmer opened a door such that the collective number of distance that all cows moved, i.e. the number of moves between the rooms, is minimum, then what is that minimum distance?

Solution.

After understanding the problem, we can simulate the process where the farmer opened the first room. Then, the total distance can be represented as $(n-1)r_n + (n-2)r_{n-1} + \cdots + 2r_3 + r_2$ since all cows must move to certain location somehow.

First, we can store the inputs in a vector.

Code 28.2.6: CS0202.26057-03

```
int n;
cin >> n;
vector<int> config(n);
for (int i = 0; i < n; i++) {
    cin >> config[i];
}
```

Now we want to simulate the process of cows entering the room and finding the spot. Moreover, we want to simulate all possible cases to actually find the maximum value.

```
vector<int> SUM(n);
for (int i = 0; i < n; i++) {
    SUM[i] = 0;
    for (int j = 1; j < n; j++) {
        if (j + i < n) {
            SUM[i] += j * config[j+i];
        } else {
            SUM[i] += j * config[j+i-n];
        }
    }
}
```

Lastly, we can complete our solution by printing the minimum value from our `SUM` vector.

```
vector<int> SUM(n);
for (int i = 0; i < n; i++) {
    SUM[i] = 0;
    for (int j = 1; j < n; j++) {
        if (j + i < n) {
            SUM[i] += j * config[j+i];
        } else {
            SUM[i] += j * config[j+i-n];
        }
    }
}
```

As you can see, the nested for loops will go through each case where `i` represents the door that the farmer opens. Then, with the nested for loop, we add the count using the math formula shown above to compute the total distance moved by the cow. With the distances stored, we can now find the minimum value.

```
int min = SUM[0];
for (int i = 0; i < n; i++) {
    if (min >= SUM[i]) {
        min = SUM[i];
    }
}
cout << min << "\n";
```

This solves the problem!

Continuing, let's discuss complete search, or brute forcing all cases.

28.2.2 Complete Search

§28.3 Practice Problems and Solutions

Below is the list of links to practice problems for this note and personal solutions.

1. **Source:** 2019 USACO Second Bronze
   Problem: Problem 1. Shell Game
   Solution: Enoch's Solution (Git)

Part VI

Appendix

Contents

Bibliography **541**

Bibliography

[Ant98] Howard Anton. *Calculus: A New Horizon*. Wiley, 1998. ISBN: 9780471243496.

[BC07] Richard Bronson and Gabriel B. Costa. *Linear Algebra*. Elsevier, 2007. ISBN: 9780080510262.

[LNS07] Isaiah Lankham, Bruno Nachtergaele, and Anne Schilling. *The Fundamental Theorem of Algebra*. 2007. URL: https://www.math.ucdavis.edu/~anne/WQ2007/mat67-Ld-FTA.pdf.

[Laa19] Antti Laaksonen. *Competitive Programmer's Handbook*. 2019. URL: https://github.com/pllk/cphb.

[Erd20] J.M. Erdman. *Exercises And Problems In Linear Algebra*. World Scientific Publishing Company, 2020. ISBN: 9789811220425.

[Yao20] Darren Yao. *An Introduction to the USA Computing Olympiad*. 2020. URL: https://darrenyao.com/usacobook/cpp.pdf.

[Ola22] Joel Olawanle. *Big O Cheat Sheet – Time Complexity Chart*. 2022. URL: https://www.freecodecamp.org/news/big-o-cheat-sheet-time-complexity-chart/.

[Pam23] Paris Pamfilos. *Desargues' theorem and perspectivities*. 2023. URL: https://users.math.uoc.gr/~pamfilos/eGallery/problems/Desargues.pdf.

[Pom24] Alex Pomeranz. *LearnCpp*. 2024. URL: https://www.learncpp.com/.

[AS26] Dr. Leonardo De Assis and Dr. Keshav Sutrave. *Multivariable Differential Calculus (XM521)*. 2026.

[Gee26] GeeksforGeeks. *C++ Programming Language*. 2026. URL: https://www.geeksforgeeks.org/cpp/c-plus-plus/.

[Kan26] Margarita Kanarsky. *Linear Algebra (XM511)*. 2026.

[Mic26] Microsoft Learn. *C++ documentation*. 2026. URL: https://learn.microsoft.com/en-us/cpp/cpp/.

[USA26] USACO Guide. *USACO Guide*. 2026. URL: https://usaco.guide/.

[W3S26] W3Schools. *C++ Tutorial*. 2026. URL: https://www.w3schools.com/cpp/.

[WC26] Eric W. Weisstein and David W. Cantrell. *Power Mean*. 2026. URL: https://mathworld.wolfram.com/PowerMean.html.

[Win26] WinnersKMO. *WinnersKMO*. 2026. URL: https://www.youtube.com/@WINNERSMATH.