HOW I FELL IN LOVE WITH MATHEMATICS

JASON MEDCOFF

Abstract. This article explores the mathematical interests and motivations that drove me to study mathematics. I will examine some experiences and events that induced a scientific interest in me, followed by my journey so far with mathematical sciences in higher education. I assess my achievements in the field, and make projections of the near future.

Contents

1. Introduction

It is often said by [1] that we study history so as not to repeat its mistakes . I believe it is also true that studying history may give us renewed motivation. It is with this in mind that I will explore my journey in mathematics over the years, so that I may scrutinize my own motivation for my field of study.

I was born the son of two electrical engineers. My father grew up on a farm in Michigan, and after pursuing his degree, entered the software industry. My mother, being the daughter of an engineering professor, met my father while studying at Wayne State University in Detroit. I spent most of my formative years living with my parents and younger sister in Troy, bicycling, walking, and reading.

Early on, my parents instilled in me a sense of curiosity, especially with regard to natural science and technology. I explored some of these interests with simple chemical experiments, circuit kits, and vintage Fischertechnik sets. I read books that explained the inner workings of cars, computers, and internal organs. The time spent tinkering with such technical contraptions led to my interest in the hard sciences and engineering later on.

2. Secondary education

Upon entering high school, I made a point of exploring as many opportunities as I could. I took every science available, from microbiology to chemistry, and continued my mathematical education as prescribed by the school. My epiphany occurred in my final year, when I concurrently enrolled in physics and calculus. The former was taught without any assumed knowledge of the latter, but regardless, I was able to draw the parallels. I had never before seen the world described with such mathematical precision. The way that physical science could describe the universe with such precision amazed me.

It was at this time that I discovered programming. Taking an interest in computers, I picked up the python language and began writing scripts to solve my physics homework. I knew that with my trivial programs, I was only scraping the tip of the iceberg. The feeling of harnessing the power of a microprocessor to do my bidding consistently gave me an endorphin rush. I knew what I had to do.

I had applied to Oakland University and was already accepted. I declared a major in computer science, hoping to deepen my understanding of the nature of programming. I had a desire to explore connections between computing and natural science. I graduated, and before I knew it, I was off to Oakland in pursuit of glory and a degree.

3. College education

I began my university mathematics with a second round of calculus. In fall of 2014, I was enrolled in MTH 154 under the instruction of Kevin Andrews. This was a class that I greatly enjoyed. Professor Andrews was a fantastic lecturer, and his graded material did not assume too much from the student. I passed the course easily with full marks.

The following semester, I took MTH 155 as an evening class, which was a huge mistake. Unfortunately, there is no nice way to put this. I despised going to study integrals until 21:30 at night. Although I didn't like it, I still studied and put in the work necessary for a 4.0.

During this first year, I slowly began to find mathematics surprisingly boring. I suspect this came mostly from the knowledge that I was repeating a subject that I had already learned. This was largely apparent during MTH 154 and the first half of MTH 155. My high school course in calculus had already covered differentiation and integration, with the only new subjects at the university level being sequences and series, the (ε, δ) definition of a limit, and a more thorough treatment of the fundamental theorem. Even with this in mind, I still felt that I was missing something. I understood that we as students were simply expected to understand the processes taught during lecture, and demonstrate that understanding on examinations. This felt like only part of the big picture.

While taking calculus, I was studying in the computer science department's introductory sequence of programming classes. The requisite courses were mostly boring for me. The first was a beginner's treatment of unix and C. I had already experimented with gnu/linux far beyond its scope, and found C to be rather clunky in comparison to my native python. The next course was in object oriented programming, and started my dislike of the java language. It didn't feel real, in the sense that java focuses on piecing together code written by other people to do what you want.

The following year's first semester brought a heavy workload, thanks to courses in electricity and magnetism, circuits, data structures, and discrete mathematics. I picked up the math very easily, thanks to the instruction of the eminent Eddie Cheng. Both the subject material and lecture style caused me to enjoy attending this class. The context of sets and logic suited mathematics in my mind, and so I looked forward to the following semester, where I would learn linear algebra.

On the first day of class, I walked out of the lecture room in the mathematics and science building with a smile on my face. I had experienced the first of many lectures by Serge Kruk, and I immediately knew that I was going to like him. Despite belonging to the faculty of mathematics, he was a self proclaimed computer scientist. He would occasionally bring in a laptop and demonstrate some principle of the course material using GNU octave. Professor Kruk also presented a handful of applications, such as Google's page ranking algorithm, and the mathematics behind training artificial intelligence with a data set. This influential linkage of computer science and mathematics was not lost on me. Once or twice, I talked with Professor Kruk outside of class in a casual setting, and he invited me to enroll in his algorithms class the following semester. I was sold.

Design and analysis of algorithms was what I consider to be my first real computer science course. This class sated the hunger that had developed for difficult theory and academic readings. We studied the classical algorithmic problems in CS: searching, sorting, and graph traversal. Serious context was finally given to the previous year's discrete mathematics course. We read papers by McIlroy, Edmonds, and Dijkstra. I was beginning to see the truth. Computer science was not really best described as a science. It also had little to do with computers.

Linear algebra and algorithms are two of the most influential classes I have ever taken. Near the end of algorithms, I decided I was ready for more. I approached Professor Kruk, asking about research opportunities during the summer. We discussed two possibilities, but eventually settled on one.

After algorithms, I took a programming languages class under Professor Hua Ming. The class was meant as both an introduction to functional programming and a basic treatment of compiler design. The sheer simplicity of scheme, the language the class was taught with, felt very close to mathematics. Recursion was simple to write and understand. It was close to writing a proof using mathematical induction. It also helped me to understand lisp languages.

I began my work with Serge Kruk in May of 2017. The chosen implementation language was lisp, hot on the heels of programming in scheme.

4. Research

One constraint of interest in CP (constraint programming) is the `at_least` predicate. The general formulation is `at_least(m, n, k, l)`, read as "at least m of $x_1, x_2, \ldots, x_n$ equals k for x_i, $i \in \{0, 1, \ldots, l\}$". The first phase of the project focused on deriving an IP (integer program) from a formulation of the `at_least` predicate. The resulting code (referred to as the generator) returns an augmented matrix representing the IP.

The next phase was my main contribution to the project. I wrote a piece of code (referred to as the projector) that would take the generator's IP formulation and project its polytope onto $x_1, x_2, \ldots, x_n$. The method used is Fourier-Motzkin elimination, so named because it was discovered independently by Joseph Fourier and Theodore Motzkin. My version of the algorithm is described as follows.

Theorem 1. *Consider a system of the form $\boldsymbol{Ax} \leq \boldsymbol{b}$, with the system containing m inequalities and n variables. Decide on a variable to eliminate from the system x_i. For each inequality, divide by the absolute value of the coefficient on x_i if it is nonzero.*

Construct three sets ***pos****,* ***neg****, and* ***naught****. Place in* ***pos*** *those inequalities that have a positive coefficient on x_i. Place in* ***neg*** *those inequalities that have a negative coefficient on x_i. Place those inequalities with a zero coefficient on x_i in* ***naught****.*

For every pair of inequalities P from ***pos*** *and N from* ***neg****, place the inequality $P + N$ in* ***naught****. At the end of this process,* ***naught*** *contains a system equivalent to the original, projected onto x_i.*

Repeat this process as needed until the desired variables have been eliminated.

This process of Fourier elimination is computationally slow. Every elimination step will generally produce redundant inequalities belonging to one of three classes: trivial inequalities of the form $\alpha \geq 0$, inequalities that are multiples of others in the system, or inequalities that are loose bounds on the polytope. These additional inequalities greatly slow successive elimination steps, as the sets `pos` and `neg` will grow exponentially.

The detection of these redundant inequalities can be made simple by way of [2], but the pairwise addition of inequalities is still slow. This suggests the implementation of parallelism. We can divide the sets of inequalities among several threads of a computer, and due to the order of inequality combination being independent, this significantly speeds up the algorithm.

Following the entire process of FM elimination, we have a system equivalent to the original, that has been projected onto every variable except for $x_1, \ldots, x_n$ as specified by the `at_least` predicate.

5. Today and beyond

I greatly enjoyed the taste of applied mathematics offered by this project. Since the beginning of summer, I declared a mathematics minor so that I may continue to pursue study in the field. My current goals are further research in applied mathematics and theoretical computer science. I'm currently looking forward to my current semester, where I will study proof writing with Professor Tony Shaska and theory of computation with Professor Hua Ming.

In the following semester, I will further my study of multicore computing with a computer science course by the same name. I will also study groups and rings in abstract algebra I and take a special topics course with Professor Kruk. After that, I will graduate.

Just as I have in the past, I look forward to my studies in mathematics this semester. I hope my time studying proofs will be highly stimulating.

References

[1] Santayana, George. The Life of Reason.
[2] Imbert, Jean-Louis. About Redundant Inequalities Generated by Fourier's Algorithm.