

PROJECT PROPOSALS: MATHCAMP 2005

SURREAL NUMBERS

Supervised by: Alfonso and Pedro

Description: “In the beginning, everything was void, and J.H.W.H. Conway began to create numbers. Conway said, ‘Let there be two rules which bring forth all numbers large and small. This shall be the first rule: Every number corresponds to two sets of previously created numbers, such that no member of the left set is greater than or equal to any member of the right set. And the second rule shall be this: One number is less than or equal to another number if and only if no member of the first number’s left set is greater than or equal to the second number, and no member of the second number’s right set is less than or equal to the first number.’ And Conway examined these two rules he had made, and behold! they were very good[...]”

Starting from nothing, we will construct surreal numbers and operations within them using just a set of axioms, assuming nothing and proving everything we need. Actually, *you* will prove it. Didn’t you ever wonder why $1 + 1 = 2$?

Expected project output: A formally rigorous, self-contained, theory of surreal numbers. How far you get depends on you.

Difficulty level: Medium/high.

Mathematical prerequisites: Abstraction and proof techniques.

Math you’ll get to learn along the way: Surreal numbers, of course.

Computer use: No.

Preferred number of project participants: Small group.

Degree of supervision: This will be a Moore-method-like project. We will give you definitions and a series of exercises that should lead you to derive all the interesting properties. It is your task to prove them. We are there to check your results and guide you when you get lost, but you are the ones working.

COUNTEREXAMPLES IN POINT-SET TOPOLOGY

Supervised by: Alfonso

Description: Some time ago Dover published a little book called “Counterexamples in Point Set Topology” which provided a list of every possible pathological property that a topological space may satisfy – or so they claimed. Mathematicians are divided into those who wish such book had never been published (chicken) and those like me (brave).

The usual topology in \mathbb{R}^n is boring. I like spaces where sequences are useless because there are too many non-comparable open sets. I like connected, path connected spaces which are not locally connected. I like accumulation points of sequences not to be the limit of any subsequences. Your life is not spiced enough if you do not use Zorn’s lemma at least once per day!

Expected project output: A big diagram containing all possible combinations of standard properties of topological spaces, with a counterexample of every non-implication.

Difficulty level: High. You need abstract thinking and ingenuity, as well as being ready to be called “heretic”.

Mathematical prerequisites: Point-set topology.

Math you'll get to learn along the way: More point-set topology? How to think far away from the box.

Computer use: No.

Preferred number of project participants: 1 or a few.

Degree of supervision: Not much.

CONSTRUCTING A 17-GON WITH A STRAIGHT-EDGE AND A COMPASS

Supervised by: Alfonso.

Description: You probably know how to build an equilateral triangle, a square, and an hexagon with a compass and a straight-edge. You may have also learned a method to build a pentagon, or even a general method for a regular polygon with n edges. Well, they lied to you. That general method is just an approximation and there is not a way to construct a regular heptagon. I am not saying we do not know a method; I am saying that we can prove that it is impossible to construct. A regular 17-gon, on the other hand, can be constructed. Are you intrigued?

Expected project output: Draw a regular 17gon! If you want to go farther, we will calculate which are the numbers n for which a regular n -gon can be constructed.

Difficulty level: Medium.

Mathematical prerequisites: Basic complex numbers, some number theory.

Math you'll get to learn along the way: Some number theory and abstract algebra.

Computer use: Not necessary.

Preferred number of project participants: 1 or a few.

Degree of supervision: More in the beginning. Then you will spend more time alone trying the actual construction of the 17gon. If you want to solve the bigger question, more again in the end.

$\Pi_1(\text{DORMS})$

Supervised by: Alfonso and Anti.

Description: I stand at the MC office holding one end of a very long rope. You take the other end and run through the corridors and lounges of Scholz and Foster (do not forget the laundry room!), then come back to me and give me the other end. At this point, I pull both ends and try to recover as much rope as possible. If there were no walls (or floors or ceilings), I would be able to recover all of the rope. But since there are walls, the rope will stay describing a *path* through the dorms. Can you describe all the possible paths constructed in this way? In other words, can you describe the fundamental group of the space of corridors and lounges?

Expected project output: A description of the aboved-mentioned fundamental group.

Difficulty level: Depends strongly on your background

Mathematical prerequisites: You will understand this much better if you have taken Anti's "Orbispac" or Ari's "Groups via generators and relations".

Math you'll get to learn along the way: Some group theory, some topology.

Computer use: No.

Preferred number of project participants: 2 or 3.

Degree of supervision: Depending on your background, much at the beginning to give you the initial tools and concepts. Less later, while you run along the corridors.

OPEN PROBLEMS IN COMBINATORIAL GAME THEORY

Supervised by: Alfonso

Description: Here are some real problems which game theorists consider interesting and which still need to be solved. Do not despair: although very difficult, they are not intractable:

- *Gale's Vingt et un.* Cards numbered 1 through 10 lie on the table. A chooses a card. Then B chooses cards until his total of chosen cards exceeds the card chosen by A . Then A chooses cards until her cumulative total exceeds that of B , etc. The first player to get 21 wins. Who is it?

I always thought that this was a very small problem, and that we should be able to check all the cases and solve it. But apparently not.

- *3D Chomp* (\$100 price)
- *Turning turtles* We have a line of turtles, some of them on their back, some of them on their feet. At each move a player must put one turtle on its back and may also turn over any single turtle to the left of it. This second turtle, unlike the first, may be turned either onto its feet or onto its back. The player who turns the last turtle upside-down wins.

This simple game, and various generalizations of it, alternate patterns of chaos and order.

Expected project output: A general solution for the game would be publishable. A partial result would be nice, too.

Difficulty level: High, I suppose.

Mathematical prerequisites: Being familiar with basic game theory would help.

Computer use: Would be useful.

Preferred number of project participants: 1 per game, probably.

Degree of supervision: Depends on you.

GENERALIZING ARCHIMEDES'S METHOD

Supervised by: Anti and Sam (by email)

Description: Use mechanical methods, such as balanced levers, to compute volumes and centers of gravity of geometric objects. Archimedes did the 3-sphere. Can you compute the volume of a 4-sphere this way? What about an n -sphere? Or a solid defined by equations of degree higher than 2?

Expected project output: Diagrams of balanced levers illustrating the volumes of various figures.

Difficulty level: ***

Mathematical prerequisites: Class on the method of Archimedes.

Math you'll get to learn along the way: More method of Archimedes.

Computer use: None required.

Preferred number of project participants: 1-3.

Degree of supervision: Light.

THE MOTHER OF ALL TOPOLOGIES

Supervised by: Alfonso.

Description: Consider the set of all topological spaces up to homeomorphism. (OK, it is not a set; we can ignore that for now or restrict ourselves to topological spaces whose ambient set has cardinality bounded by some fixed cardinal). There are various topologies that we can define in this set, and I will suggest some ideas. But none of them are good enough. I would like such a topology to have certain properties, but the main one is the following.

There is a natural distance, called the Hausdorff-Gromov distance, defined on the “set” of all metric spaces. I would like to define a topology on the “set” of all topological spaces such that the subset of all metrizable topological spaces inherits a topology which is the same as the one generated by the Hausdorff-Gromov distance. In a way, we would be “generalizing” the Hausdorff-Gromov distance to non-metrizable topological spaces. I do not know how difficult (if at all possible) this will be.

Expected project output: Definition of a topology as described above. If the results are nice enough, this could become a publishable paper.

Difficulty level: High.

Mathematical prerequisites: Point-set topology.

Math you’ll get to learn along the way: More point-set topology.

Computer use: I do not think so.

Preferred number of project participants: Up to you.

Degree of supervision: Up to you.

RICOCHE T ROBOT COMPUTER SOLVER AND ANALYSIS

Supervised by: Dan

Description: The game “Ricochet Robot” is an intriguing board game where players compete to move robots from their starting locations to a destination location. The robots can only move in one direction, and can only stop by hitting a wall or another robot.

Most games proceed without any knowledge of what the optimal solution really is — it is often very difficult for a human to determine this quickly. Computers might have the processing power to do so, although the number of possible moves is exponential.

As a first goal, this project will attempt to produce a program that can solve Ricochet Robot quickly — given four robots and a goal, how fast can the target robot move there?

Possible additional goals include: programming a good heuristic algorithm (i.e. one that doesn’t necessarily find the optimal solution, but very quickly finds a good solution), proving that the general problem on an $n \times n$ board is NP-complete, and looking at other theoretical aspects of the problem.

Expected project output: A Ricochet Robot computer solver, and possibly also some results about the difficulty of variations on the game.

Difficulty level: Medium-to-low.

Mathematical prerequisites: If we go the NP-completeness route (which I’d like to), a solid basis in complexity theory. The first two weeks of Theoretical CS are sufficient.

Math you’ll get to learn along the way: Data structures in algorithms, a solid foundation for how to prove NP-completeness (hopefully — it still might not be NP-complete...).

Computer use: Heavy.

Preferred number of project participants: 2-3

Degree of supervision: Low. I plan to check how you’re doing, find out what your algorithms are, and argue with you about them, but that’s about it. I might help out more if we go the NP-complete proof route.

THE UNIQUE GAMES CONJECTURE

Supervised by: Dan

Description: This is actually a pretty big open problem in computational complexity.

A *unique game* consists of a graph G (represented, say, by a list of vertices and edges), a set of colors C , and, for each edge $\{i, j\}$ with $i < j$ a permutation $\pi_{ij} : C \rightarrow C$ (i.e. a rule specifying what the color of vertex j must be given the color of vertex i).

One can decide in polynomial time if a coloring of the edges exists that satisfies π ; you should be able to see this for yourself.

Here is an unproved conjecture: for each δ with $0 < \delta < 1$, there is a fixed finite color class C such that it is NP-hard to distinguish if there is a coloring that fulfills the requirements on a $1 - \delta$ fraction of the edges, or if every coloring fulfills at most a δ fraction of the edges.

This conjecture is a pretty big open problem in CS. We'll try to prove it.

Expected project output: Hopefully, a proof of the conjecture or a disproof (e.g. showing that the problem is in a complexity class contained in NP).

Difficulty level: Hard.

Mathematical prerequisites: Theoretical CS. A bit of graph theory wouldn't hurt.

Math you'll get to learn along the way: Lots of complexity classes. Possibly some stuff about randomized computation.

Computer use: None, or little (probably).

Preferred number of project participants: 3-4.

Degree of supervision: Moderate-to-heavy; I'll point you to papers and discuss what we read.

SOLVING SAT

Supervised by: Dan

Description: I don't know very much about this area, but there are many algorithms out there that can approximate SAT or, at least, come up with a solution efficiently in *most* instances.

We'll examine some of these algorithms and see if we can come up with one of our own. We'll try to implement one or more of the algorithms.

Expected project output: A computer program that will, to some degree, solve SAT or approximate SAT efficiently.

Difficulty level: Easy/moderate.

Mathematical prerequisites: Theoretical CS.

Math you'll get to learn along the way: Algorithms and programming.

Computer use: Medium.

Preferred number of project participants: 2-3

Degree of supervision: Low-to-medium. I expect you to seek out algorithms online on your own, and then explain them a bit to me so I can give suggestions and see where you are.

PSEUDORANDOM GENERATORS

Supervised by: Dan

Description: A pseudorandom generator is a Turing machine that, given a random input, outputs a sequence of numbers that "look random." So, what does "look random" mean? Well, it might mean that a computer program that doesn't have much time to run can't

predict the next output given some of the earlier output. Or it might mean that it can't "distinguish" the output from truly random input.

We'll try to understand pseudorandom generators and the theory behind them. We might tie their existence into the P vs. NP question. And perhaps we'll try to come up with one of our own.

Expected project output: A knowledge/presentation on pseudorandom generators. Possibly a programmed pseudorandom generators, or even an entirely new pseudorandom generator.

Difficulty level: Can vary.

Mathematical prerequisites: Number theory. Theoretical CS is very helpful but not absolutely required.

Math you'll get to learn along the way: How pseudorandom generators work.

Computer use: Probably very low, but it depends where we go with it.

Preferred number of project participants: 3ish.

Degree of supervision: Medium. I'll start it off and provide some references, then let you go at it a bit.

A KNOTTY PROBLEM

Supervised by: Ari

Description: One long-standing goal of knot theorists is to list all knots up to n crossings, where n has grown larger as our mathematical and computational capabilities have increased. (It's around 16 now, and working towards 17.) The first step is to list all n -crossing knot diagrams; this can be done fairly easily using Dowker or Conway notation. The second and more much more difficult step is telling which knot diagrams represent the same knot! The Reidemeister moves can be used to get between diagrams representing equivalent knots, but there is no upper bound on the number of moves which may be necessary to get between two diagrams. Fortunately, there are knot invariants which can help us tell knots apart.

Your task is to follow in the footsteps of early knot theorists, and make your own table of knots!

Expected project output: A knot table. The value of n will depend on how much time you spend on it. I might also ask you to prove that certain quantities are knot invariants.

Difficulty level: Medium

Mathematical prerequisites: None

Math you'll get to learn along the way: Reidemeister moves, Dowker and/or Conway notation, some knot invariants (tricolorability, unknotting number, Alexander polynomial, Jones polynomial, possibly others)

Computer use: none

Preferred number of project participants: 1-4 (multiple independent groups okay)

Degree of supervision: After the initial learning of notation, most of the actual tabulating can be done unsupervised. I will probably give mini-lectures on some invariants during TAU.

LEBESGUE INTEGRATION

Supervised by: Ellen and Shilpa

Description: Measure theory and Lebesgue integration are essential for modern analysis and probability. Inspiration for the use of the Lebesgue integral are things like a desire to be able to compute

$$\int_0^1 f(x)dx$$

where

$$f(x) = \begin{cases} 1 & \text{if } x \in \mathbb{Q} \\ 0 & \text{if } x \notin \mathbb{Q} \end{cases}$$

or an interest in computing the probability that the above function $f(x)$ is 1 when picking a random $x \in [0, 1]$.

We hope to have this be a reading course, where the students read on their own and then discuss what they've learned with each other. Our role will mainly be to moderate discussion and answer any questions that you cannot answer on your own.

Expected project output: An understanding of measure theory and Lebesgue integration

Difficulty level: Depends on how much you want to learn and in what depth.

Mathematical prerequisites: Understanding of the Riemann integral. (Mathcamp calculus is enough.)

Math you'll get to learn along the way: Measure theory of course!

Computer use: Probably nonexistent.

Preferred number of project participants: ≥ 2

Degree of supervision: As much as you need (but only as much as we can give).

TOPICS IN POLYTOPES

Supervised by: Ellen

Description: If you are taking my polytopes class (or if you've taken Sam and Matt's class and want to see *alternate* proofs of some theorems) the following topics may be of interest:

- (1) Think of any 3 dimensional polytope (anything! even non-symmetric!). Let f_0 be the number of vertices, f_1 the number of edges, and f_2 the number of facets. Now calculate $f_2 - f_1 + f_0$. Do you get 2? The fact that you always get 2 is the 3-dimensional version of the Euler-Poincare formula, which you can prove fairly easily using indicator functions.
- (2) If you blaze through the Euler-Poincare formula, you can also prove the Dehn-Sommerville equations, which give inequalities describing how many faces a polytope can have in each dimension.
- (3) The Euler-Poincare formula and Dehn-Sommerville equations describe which f -vectors *can* correspond to polytopes. In \mathbb{R}^3 , the Euler-Poincare formula and Dehn-Sommerville equations describe which f -vectors *do* correspond to polytopes. You will prove the above by constructions of such polytopes. (Surprisingly, the Euler-Poincare formula and Dehn-Sommerville equations do *not* describe which f vectors do correspond to polytopes in \mathbb{R}^4 !)

Expected project output: Proofs of some or all of the above theorems.

Difficulty level: Medium-advanced

Mathematical prerequisites: An understanding of most of the terms stated above (polytope, face).

Math you'll get to learn along the way: Those exciting theorems you'll be proving!

Computer use: none

Preferred number of project participants: 1-?

Degree of supervision: Probably quite a bit, but no more than you want.

KNIGHT'S TOURS ON CHESSBOARDS

Supervised by: Mark

Description: Consider a knight moving on an $m \times n$ chessboard using legal moves (two squares in one grid direction and, at the same time, one square in a perpendicular direction). Can the knight get to all the squares without visiting any square more than once? If so, we have an "open" knight's tour of the board. If it can be done so that at the end of the tour the knight can move back to its starting square, we have something better yet, a "closed" knight's tour. The purpose of the project is to investigate for which values of m and n each kind of knight's tour is possible, and to exhibit tours (they make nice pictures) to illustrate the results.

Expected project output: Poster

Difficulty level: **; although the problem is easy to state, getting good results requires some persistence and ingenuity.

Mathematical prerequisites: Proof by induction (although I suppose this could be learned along the way)

Math you'll get to learn along the way: Probably none, except for the specific results of the project.

Computer use: None expected; it's probably harder to write code for this problem than to do it by hand.

Preferred number of project participants: 1 or 2

Degree of supervision: Minimal, I hope; I'd like to hear from participant(s) at least every couple of days, but work can be done at their convenience, and unless they're seriously stuck I don't have to be there while they do it.

MATHEMATICAL FLIP BOOKS

Supervised by: M@

Description: Computer animation allows us to visualize things we might not be able to otherwise by using time as the "extra" dimension. For this project we'll be exploring the most low-tech version of animation there is: flip books. We'll start off by brainstorming on ideas for what to illustrate. Then you'll get together in small groups and write code to draw one frame at a time. Then laying up, copying, cutting, and collating. There will be a fair amount of grunt work involved but the final result can be quite satisfying.

Expected project output: Flipbooks of course! (See samples in the office lounge that I made a while ago.) You will get your very own book to show off at the project fair and then take home.

Difficulty level: Mathematically, it depends on what you want to draw pictures of!

Mathematical prerequisites: None, but at least one or two people in each group should have some coding experience.

Things you'll have to learn along the way: How to draw mathematical pictures with a computer. Maybe a little math concerning projections and cross sections and such.

Computer use: Required

Preferred number of project participants: Up to 15-20, which I will break up into groups of 4-5 at the first meeting.

Degree of supervision: Meetings once or twice a week.

PROPER CARE AND FEEDING OF PROCESS CALCULI

Supervised by: Ryan

Description: Formally, a process calculus is a language of expressions with a set of rules for reducing some expressions into others which captures in some way the idea of multiple processes running alongside each other at the same time and interacting with each other. Informally, a process calculus is a way of talking about systems that are made up of lots of little things all happening at once, communicating back and forth and creating, destroying, or changing each other—much like real life! With the power of process calculi, we can begin to analyze not only systems like computer programs and networks, but also security procedures, personnel infrastructures, politics, ecosystems, and even biological cells and organs. Come learn about some of the more distinguished process calculi that have already been developed, and then design and analyze your own!

Expected project output: A process calculus (or two) of your very own!

Difficulty level: Moderately easy with some programming experience, fairly challenging without.

Mathematical prerequisites: Familiarity with mathematical logic is a big help, but not a strict requirement. Familiarity with algebra, category theory, or the lambda calculus are all small helps. Don't be scared off if you are not the hardcore computer science type—your untainted perspective will be an asset!

Math you'll get to learn along the way: Who knows?

Computer use: Not required or even particularly encouraged, although those who would be interested in coding simple implementations of their calculi are welcome to do so.

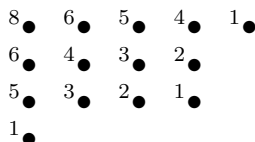
Preferred number of project participants: Four to six, but not too many more than that.

Degree of supervision: Medium, with less supervision once you actually start developing your own calculi (I want to be surprised by your ideas!)

t -CORE PARTITIONS

Supervised by: Jackie and Holly

Description: For each node in the Ferrers-Young diagram of a partition, you can assign a hook number by counting the number of nodes directly below or to the right of the node plus itself. For example, for the partition $5 + 4 + 4 + 1$ of 15, the hook numbers are as pictured:



For a positive integer t , we say that a partition is t -core if none of the hook numbers from its Ferrers-Young diagram is a multiple of t . For example, the partition $5 + 4 + 4 + 1$ is t -core for $t = 7$ and $t \geq 9$.

In an undergraduate project, Jackie proved that if t and s are relatively prime, then there are only finitely many partitions (out of all the infinitely many partitions of all numbers) that are both t - and s -core.

Recently, Jorn Olsson and Dennis Stanton proved some further results involving simultaneous s - and t -core partitions. For one result in their paper, they prove that there is a largest s - and $s + 1$ -core partition. By largest, we mean that the Ferrers-Young diagram of every s - and $s + 1$ -core partition fits inside the Ferrers-Young diagram of the largest s - and $s + 1$ -core partition.

The goal of this project is to generalize the result of Stanton and Olsson by either generalizing their proof or finding a different proof. We would like to prove that, if s and t are relatively prime, then there is a largest s - and t -core partition.

Expected project output: If we can prove the conjecture, a paper!

Difficulty level: The problem will be easy to understand, but proving the conjecture will probably not be easy.

Mathematical prerequisites: Basics of partitions

Math you'll get to learn along the way: Lots about partitions and maybe some connections to representation theory

Computer use: None

Preferred number of project participants: A few

Degree of supervision: Moderate to High

GÖDEL'S INCOMPLETENESS THEOREM

Supervised by: Pedro.

Difficulty Level: Advanced.

Mathematical Prerequisites: Propositional and predicate calculus.

Description: Many people talk about Gödel's Incompleteness Theorem as having profound philosophical implications. Discover what makes sense and what is sensationalist nonsense, and in the process conquer this most famous and monumental pinnacle of XX century mathematics on your own, Moore method style. This project may interest you if you've enjoyed Theoretical Computer Science.

Expected Output: A full proof of Gödel's Incompleteness Theorem from scratch.

Things you'll have to learn along the way: Recursive functions.

Computer use: None.

Preferred number of project participants: Any. Since the project is Moore Method style, everyone will work individually.

Degree of supervision: High.

TEXAS HOLD'EM

Supervised by: Pedro.

Difficulty Level: Advanced.

Prerequisites: Proficiency with discrete probabilities, really good programming skills.

Description: This is a project for a small group of people. The goal is to write a sophisticated program, making use of probability theory, that plays Texas Hold'em poker competently online (so we'll take into account how online play is different from face-to-face play), or that at the very least simulates the behavior of a good Texas Hold'em online poker player. There is an empirical component to this project. Ideally, we would have at least two or three different implementations (e.g., in one of them the program builds psychological profiles of its opponents, in another it doesn't) of the program, so that we can compare their performance. This is a programming intensive project!

Expected Output: A full implementation of a program that plays Texas Hold'em competently.

Things you'll have to learn along the way: You will have to learn about the systems that some poker experts use in their play.

Computer use: A lot.

Preferred number of project participants: More than three.

Degree of supervision: Low.

ANALYTIC NUMBER THEORY THROUGH PROBLEMS

Supervised by: Holly, Jackie, and Pedro

Difficulty Level: Basic.

Prerequisites: Calculus and number theory.

Description: This is a good project for 1-2 people. We could accommodate several couples, or several individuals. The idea is to learn as much analytic number theory as possible by doing problems. It is ideal for people who are enthusiastic about both number theory and calculus. This project will help you practice your calculus skills, and in the process learn a lot of really fun number theory. The way this project will work is as follows: we will have an initial meeting to determine what the right starting point for you is; then we will assign you a list of problems to work on, and you will submit solutions as soon as possible. This project requires constant supervision, and frequent meetings with the project advisors to work well.

Expected Output: A full set of solved problems, and an exposition of a topic of your choice.

Things you'll have to learn along the way: Varies according to the topic of your choice.

Computer use: None.

Preferred number of project participants: 1-2.

Degree of supervision: High.

NUMERICAL ORDINARY DIFFERENTIAL EQUATIONS

Supervised by: Shilpa

Description: This would be a project where you would be reading, learning and coding simultaneously. We will begin with Forward Euler's method and Backward Euler's method. With these two methods, you will explore implicit and explicit methods, and convergence and stability. Convergence is looking at how quickly will the numerical solution converge to the actual solution. Stability is a more complicated idea that will be studied by looking at examples of stiff and non-stiff ODE's. (We will talk about what this means, basically some ODE's have characteristics that cause the solution to blow up when approximated by some numerical methods (unstable methods) and not when the solution is approximated by other numerical methods (stable methods). If interest is high and time is available, we will try to explore other methods, including the Trapezoidal Rule and Runge Kutta methods.

Expected project output: The goal is to understand the numerical theory and verify it through code and examples.

Difficulty level: Medium

Mathematical prerequisites: Calculus (Mathcamp calculus does count, but you would need to work a little bit (with me) to learn about Taylor series.), Linear Algebra, and programming.

Math you'll get to learn along the way: Numerics :), an understanding of how calculus is used regularly!

Computer use: We will use matlab (if available?), octave (if graphics work?) or mathematica.

Preferred number of project participants: 2-5

Degree of supervision: As needed (and as time permits). I will expect you to be able to code without a lot of guidance.

TOPOLOGICAL FIBER ARTS

Supervised by: Anti, Shilpa

Description: In topology, we study all sorts of wacky objects like toruses, Klein bottles, and projective planes; we can prove cool stuff about them, but sometimes it's hard to actually visualize them. We've all had bagels for breakfast at the MC Cafe, so we have some notion of what a torus looks like, but have you ever picked up and played with a Klein bottle or a projective plane? In this project we'll make physical models of these objects and use them to improve our topological intuition, illustrating properties we already know and observing new ones. No experience knitting or crocheting is required. We'll study the fundamental group using actual loops of string on our surfaces, and the Euler characteristic by drawing triangles on them.

Expected project output: models of topological objects: one or more of mobius strips, Klein bottles, hyperbolic planes, knitted and/or crocheted.

Difficulty level: easy

Mathematical prerequisites: a bit of basic topology

Math you'll get to learn along the way: more topology!

Computer use: none

Preferred number of project participants: 2-3

Degree of supervision: low-medium

THE STABLE MARRIAGE PROBLEM

Supervised by: Alfonso, fMatt and Pedro

Description: n men and n women are trying to pair up for marriage. Each of them lists the potential candidates in order of preference. Of course, it is impossible to make everybody happy, because every woman's first choice is Theo. But can we at least find a stable arrangement? More specifically, if Juan prefers María over his wife, and María prefers Juan over her husband, then María and Juan will run away together. Can we avoid this situation? If you are able to find more than one stable arrangement, which one is preferable?

Once you have figured this one out, you can attack the stable gay marriage problem (where anyone can be paired up with anyone else). What about the unstable gay marriage problem? Then you can write a program that solves all of these. While you are at it, could you please improve our current system of assigning academic advisors? Or of assigning Springer books to campers at the end of camp? Please?

Expected project output: A description of an algorithm to solve the stable marriage problem and its generalizations (maybe a description of all of the solutions?) and a computer program that implements it.

Difficulty level: Easy.

Mathematical prerequisites: Programming.

Computer use: Yes!

Preferred number of project participants: 1 or a few.

Degree of supervision: Little.

EIGENCAMPERS

Supervised by: Ellen, Mira, Shilpa

Description: A linear algebra technique called “principal component analysis” allows one to take a number of points in n -dimensional space and find their principal directions of variation. For instance, if the points all lie on (or close to) some 2-dimensional plane inside the big space, the data will have two principal components (directions). These two directions provide a new set of “natural”—perhaps meaningful—coordinates for the data points.

It is interesting to apply this technique to digital pictures of human faces, where you can literally *see* the results. Each picture can be thought of as a vector in very high-dimensional “pixel space” (its coordinates are the pixel values). The principal components of a set of such pictures (often called “eigenfaces”) should, in theory, represent the main parameters along which the pictures vary. We might expect them to have some physical interpretations: gender? glasses? shapes of facial features? It would be impressive if a simple mathematical technique could recover such significant facts about human faces simply from pixel values.

In practice, “eigenfaces” usually work only moderately well, but it’s always fun to try them out. And at Mathcamp, we have an obvious data set: the yearbook pictures. These will need to be processed (also using some linear algebra) so that the faces are all the same size and properly aligned. Then you can obtain pictures of the “mean camper” and “eigencampers”, play around with them, and put them in the yearbook.

Expected project output: Cool pictures (hopefully going in the yearbook)

Difficulty level: Intermediate. The project involves learning an important mathematical technique and then implementing it in a fun context. The math is not so hard if you have the prerequisites, and once you learn it, everything else is pretty straightforward.

Mathematical prerequisites: Basic linear algebra – more or less the Mathcamp linear algebra course.

Math you’ll have to learn along the way: . A bit more linear algebra.

Computer use: Essential! However, advanced programming is not required. I’m not sure yet if we’ll do this using Mathematica or Matlab.

Preferred number of project participants: 2–3.

Degree of supervision: A lot in the beginning: I’ll give you a crash course on the math and help you set up all the computer stuff. After that, you shouldn’t need me much, but I’m around if you do need me.