3Blue1Brown Summer of Math Exposition 2021
Version 2

Feel free to animate this content ! Contact me on discord : happyves

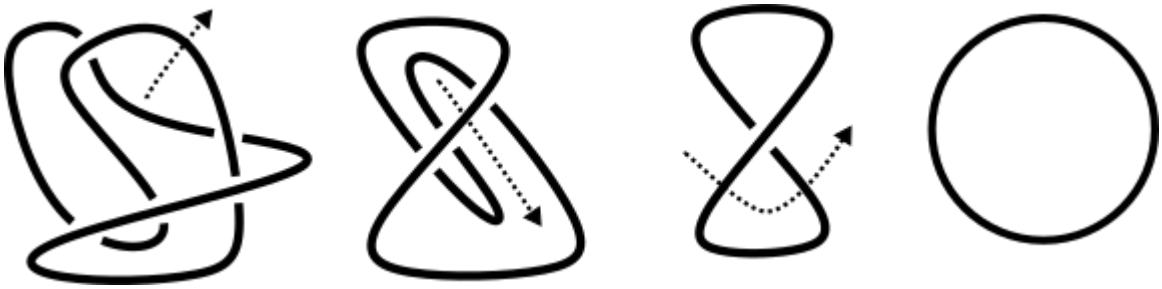
1 Knots !

The mathematician's knot isn't exactly what you'd expect it to be : it's a knotted loop in space such as the ones shown in the figure below.

However, there is a way for getting from the mathematician's knot to the sailor's knot and vice versa : *can you find it ?*

To get from the sailor's knot to a knotted loop, one can just glue the ends of the string together. If we start with a knotted loop and we want to obtain a knot such as the one we get from tying our shoes, we can cut the loop at a point and pull the ends of the string away from each other.

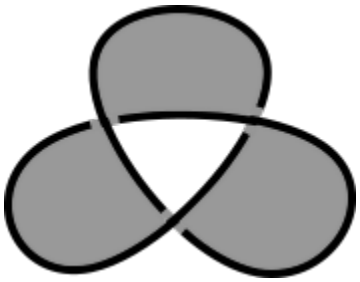
But why do mathematicians care about knots ?
It's because they are the objects of an incredibly hard puzzle, a real conspiracy of nature !
The puzzle is this : given two knots, how do we know if we can move one so as to look like the other ?
To let you see why answering this is difficult, I lied to you. The loop on the right of the figure above **isn't** knotted ! Here's how to untie it so as to look like to a circle (the moves are shown by dotted lines) :



How could you have known that I lied to you ? Is there an efficient way of telling whether a given loop can be untied without cutting and regluing it ?

It's the job of knot theorists to answer that question, among many others. Knot theory is a field of math (and computer science too ?) that is a subfield of a broader one called Topology.
The reason I chose this topic for 3b1b's expo is that undergraduate topology is rather boring and highschool topology doesn't exist. So I'd like for us to work on a question of knot theory one of who's ways to answer involves three amazing topology tricks/methods that require no prerequisite knowledge.

And yes, you heard me, **we** will work on this together ! If you read a question written in *italic* such as the "*can you find it ?* " from before, I'd like you to try to answer it before continuing reading.
The question this expo will try to answer is : **for a given knot, can we find a surface who's boundary is that knot ?**
Here's an example of such a surface :



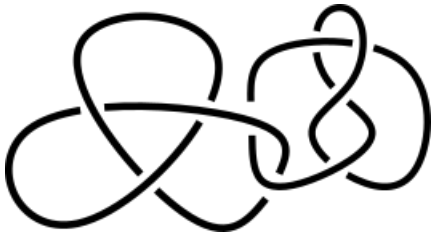
Look closely : *could you make this surface out of paper ?*

The surface is a band/rectangle which we gave 3 half-twists and who's ends we then glued together.
Open question : *can you find a spanning surface for the knot in the middle of our first figure ?*

At this stage, there is a practical question you may have asked yourself : *how did I come up with the knots we've seen so far ?* To answer honestly : I found them in books on knot theory.
But seriously : *how does one produce knots ?*

The figures you've seen aren't technically knots, but knot **diagrams**. This is a fantastic example of how finding representations of complicated objects allows us to understand those objects. What we're really asking is : *how does one produce knot diagrams ?*
Suddenly, answering becomes much easier : we can produce diagrams by drawing a loop in the plane in such a way that we pass through a same point at most twice if we ever pass through it again ; then, at each such a double-points, we choose one of the two possible of crossings (the over-under information) for our diagram. *Are you convinced by this procedure ?*

That question sounded biased and it was : this procedure may actually produce the diagram of a "link" instead of a knot. A link is a bunch of loops linked to each other such as the ones in the figure below :



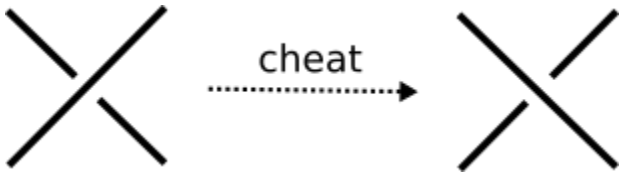
So the procedure might not work perfectly all the time : *how do we know that the procedure produced a link instead of a knot ?*

If we want a knot, the diagram should represent a single loop. So we can start at a point in the diagram and move in one direction, coloring the part of the diagram as we go over it : once we've returned to our point, we'll have colored a loop ; if this loop isn't all of the diagram, we'll know we made a link, one of who's components we colored.

Here's an open question you could spend a few years on : *by choosing the crossings at random and independently in our procedure (for a given loop), how probable is it to get a knot ?*
But now, let's return to our search for surfaces spanning knots.

2 The unknotting algorithm

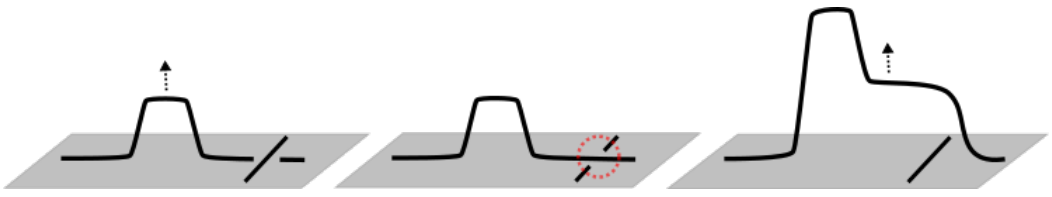
Our story starts quite far away from surfaces spanning knots. It starts with a knot theorist trying to find an efficient way to tell if we can (or not) untie a given loop. After decades of unsuccessful research, our knot theorist goes insane !
"Damn you, knots ! I'll show you a way of untying you ! I'll just cut and reglue you so as to switch two of your crossing strands from over-strands to under-strands"



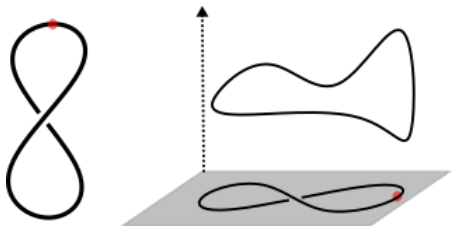
"... but how do I unknot the loop with only those specific "switching crossings" moves ?"
Could you help our poor knot theorist ?
To be clear with what we're asking : *starting from a knot, can we find an algorithm (if it even exists) that works by only switching crossings (as is shown in the previous figure) and gets us a loop that can be unknotted/untied to look like a circle ?*

The fact that such an algorithm exists may come as a surprise, since I stressed that it is hard enough already to simply tell if a loop can be unknotted/untied to look like a circle. So how could the algorithm circumvent this ?
Let's visit the knot theorist (who's been locked into a cell of an asylum by now) and ask for advice.
"I'm plagued by a feverish dream each night : in it, I see the loop of string lying in front of me on my desk ; then, I pinch it at a point and I just lift it up from the desk and ... I obtain an unknotted loop hanging from my hand !", is all we get to hear. *What can you do with this "advice" ?*

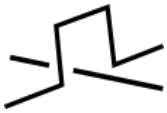
If we tried to lift the loop by pinching it at a point and pulling it up, the strand we are pulling up may be hindered in moving up by another strand passing over it. Fortunately, we are allowed to cheat : we can switch the problematic crossing so that the strand we're pulling up is now above the one that previously hindered its movement !
We can do this over and over again if necessary and expect the resulting loop to be unknotted.
Here's a figure of what we're doing.



The next figure is a (boring) example of the procedure. We pinch the knot at the red dot and lift it as we move along the knot. If we move to the right, we wont have to switch the crossing, but if we move to the left, we'll have to. On the right of the figure, we show the lifted version of the knot.



It would be amazing to see this procedure animated for a more complicated knot. For this pdf, you'll have to rely on your imagination only : imagine lifting the knot strand by strand, changing problematic crossings if needed.
It sounds feasible to produce an animation for a 3D grapher that lifts the segments of a polygonal knot (a loop made of straight edges) one by one. One could start from a polygonal knot who's crossings are handled as in the following figure and add a temporary color change or some other effect when switching crossings.
Make sure to check my discord to see if anyone has animated this expo.



Let's get back to our knots : *can you now give a precise description of an algorithm the unknots a given knot by changing its crossings ?*

Here's the full algorithm :
We choose a point on our knot as well as a direction to walk in (which correspond to the point we start lifting and the direction in which we successively lift the knot) : if we encounter a crossing we haven't passed through before and which we would have to pass along the under-crossing strand, we switch that crossing, so that we now traverse the over-crossing strand.
It's important to mention that we only switch the crossings we haven't passed through before : we know we'll pass through them twice and in any case, the second passage will be along the under-strand, but that's precisely what we want to happen, so no more switching should be performed.
The lifted loop will have only two points at each height level, so it's a (potentially twisted) circle.

Let's explain our procedure to the knot theorist : "Thank you ! I'm cured !".
And here ends the first chapter of our story.
Now, it's time to get serious about finding spanning surfaces !

3 Spanning surfaces

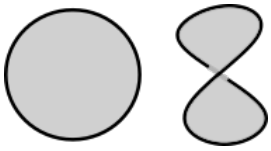
The way we'll build a surface that spans a given knot isn't the fastest, nor the most intuitive. If you want to learn about the fastest way, you can search for the "Seifert algorithm", and if you're looking for the most intuitive, I'd refer you to the "braid" representation of a knot (Alexanders theorem ; it's a gem) and tell you to place the braid strands on parallel discs and to use half-twisted bands to get the crossings.
Our way is perhaps the most beautiful.

3.1 The big idea

I wont lie to you : coming up with the big idea of our construction requires to know about some cool topology tricks. So don't feel bad for not having thought of these ideas on your own.

The main trick is described in the next subsection : it allows us to modify a surface bounding a knot in such a way that the new surface created bounds the same knot, but with one of its crossings changed. *Can you find the big idea with this trick and the previous section in mind ?*

Finding a surface that spans an unknotted circle is quite easy : any old disk will do.
If we move the boundary of that disk around, the disk moves with it, folding and twisting if necessary.
Here's a figure for a first move on a disk :

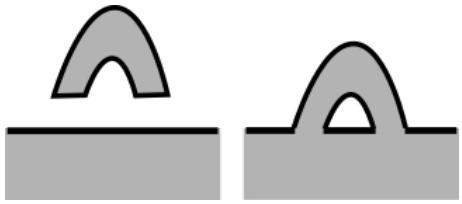


By applying the unknotting algorithm to a given knot, we get an unknotted loop which we now know to be spanned by a disk (a possibly quite twisted and folded one).
So with the trick we'll describe in the next subsection, we can modify that disk successively, switching the crossings of the boundary loop we had to switch in the unknotting algorithm to their initial state. After these changes, we'll end up with a surface that bounds the knot we initially started with.

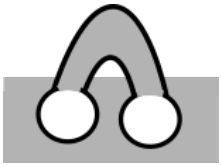
Now that we have the big idea, let's look at the tricks I promised you.

3.2 Switching crossing by gluing bands

There are many ways to modify a surface from the topological viewpoint. The first such modification we'll need consists of gluing bands/rectangles to the boundary of a surface along its opposite sides.

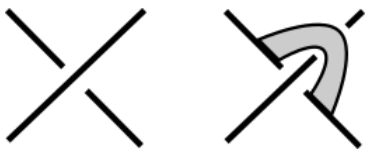


As you can see in the figure, gluing bands may result in adding a loop to the boundary (the new inner circular boundary loop in the figure). But it can also decrease the number of loops that make up the boundary of the surface, as you can see here :



Can you find a way to glue one or more bands to a surface that affects the boundary knot of that surface in a switch of crossing ?

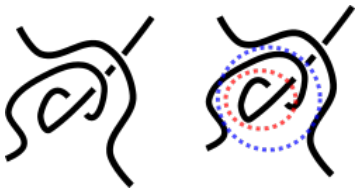
For now, we'll just ignore the rest of the surface and focus on the boundary and the bands.
Let's add a band that lets the under-strand pass above the over-strand :



Unfortunately, we now have a new loop on the boundary : it encircles the strand that used to be the over-strand. *Can we get rid of that loop with more bands ?*

How about we connect it to the the strand that used to be the over-strand :

Now, if we move the new boundary a bit (untwist in the red circle, then in the blue one and straighten the strands), we can see that the effect of adding the two bands is that of switching a crossing. Just what we wanted :

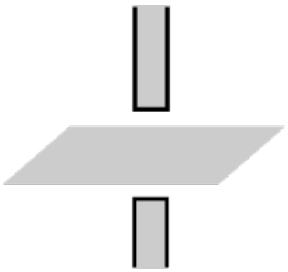


There is a serious problem we ignored in this section : the rest of the surface.
It's entirely possible that the over- and under-strand of the crossing are separated by the surface. In such a case, we can't add the bands as we did just now, since they would have to intersect the rest of the surface.
Is all hope lost ? Or is there a way we can fix this problem by modifying the surface even further ?
At this stage, the further modifications shouldn't modify the boundary of the surface, since we already have a way of getting the boundary we want. Fortunately, there is a topological tool that meets our demands : tubes.
With them, we'll get a first way of fixing the problem of bands that intersect the rest of the surface.

3.3 The funky way

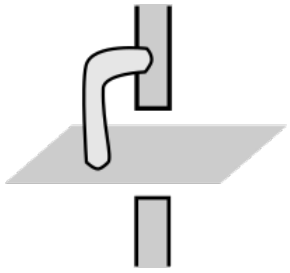
If you cut out two discs on a surface, you get two new boundary circles that you can glue a hollow cylinder/tube to. So the total effect of this "surgery" doesn't affect the boundary ; it only adds a handle to the surface.

Let's work on fixing that band-intersection problem.
We want to keep the band, so we'd like to connect the two bands in the following figure without intersecting the plane :



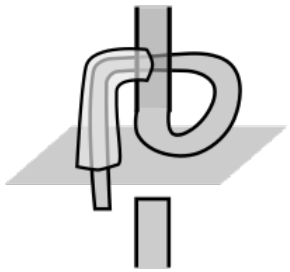
Can we use tubes to achieve this ? How ?

Putting both ends of the tube on the plane or on the same band doesn't help and putting the ends one of each band would require the tube to intersect the plane. So this is a configuration we want to investigate further :

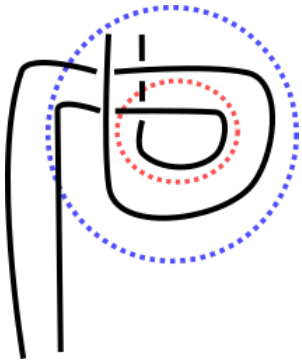


Can we connect the bands now ?

Yes, we can ! We can pass the band through the tube we attached to it.



What a funky trick ! (In the figure above, the band on the middle left should be beneath the gray plane representing the rest of the surface. I wouldn't know how to fix it because my Inkscape tricks suck...) Rigorously speaking, we still need to check that the boundary hasn't been knotted in this process :

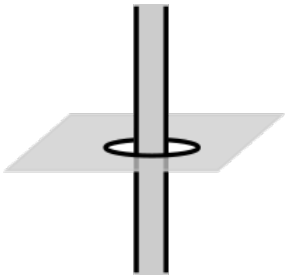


This can be seen by untwisting in the red circle, then in the blue one and finally by straightening the strands.

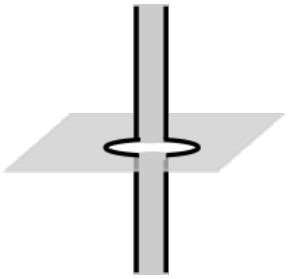
As is often the case in math, there are multiple ways to get the same result. Let's explore two more of those ways.

3.4 The cheater's way

We could just cut out a disc in the surface, let the band pass through the new hole and hope no one notices that we obtained an additional a boundary loop.



There are two ways we can proceed with to finish the job : add more bands or cut and re-glue the initial band.
If we cut and re-glue the initial band, we can see what we get from re-gluing it to the new boundary loop we created by cutting out a disc. We get a perfectly admissible solution :

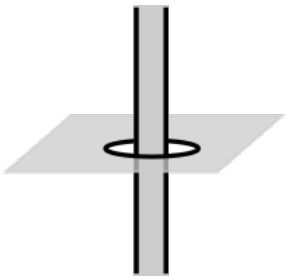


Notice that the new boundary hasn't been knotted in this process : it's only been enlarged.

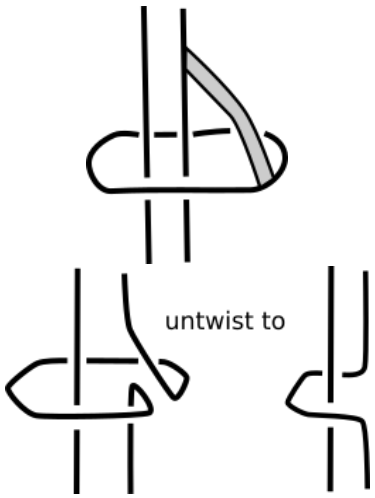
There is a way to relate this approach to the previous one. If we compress the tube in the last figure from the previous subsection to a circle along its height, we get the last figure from this subsection !

3.5 The funky cheater's way

Starting from the figure below, can you get rid of the extra boundary loop by attaching band(s) to it and still get the same boundary as for a band that doesn't intersect the rest of the boundary ?

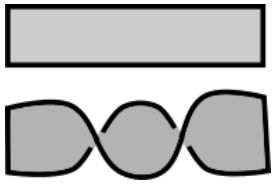


In the next figures, we represent the modified surface in the top figure, ignoring the surface of the main band as well as that around the cut-out disc, for better visibility. On the bottom one, we display the effect on the boundary. We depict what happens if we connect the the main band to the rest of the surface with another band :



The effect on the boundary is the addition of a full twist ! This is bad because the trick we used in subsection 3.2 was performed with untwisted bands. *Can we fix this ?*

We can add an additional twist in the other direction. Any straight band can be moved so as to have these two opposite twists : you can imagine giving it a half twist in the middle while holding the ends fixed to get the figure below. Here's a picture for a pair of half-twists, "cancelling" each other out :



For a full-twist, we add another pair of opposite half-twist to the ends the lower band of the above figure.

3.6 A conclusion, a remark

Let's summarise our construction. We started from a certain knot. We applied the unknotting algorithm to it by switching certain crossings. For this new loop, we found a folded and twisted disc that has this loop as its boundary. Next, we found a way to modify that disc by gluing bands to it so that the boundary of the final surface was the knot we started with. Finally, we fixed the problematic cases in which the added bands could have intersected the rest of the surface (in 3 alternative ways, no less).

I hope that you understood everything and that you're impressed with the topological trickery as much as I am. Perhaps you found your own personal tricks along the ride by trying to answer the questions in *italic*. The understanding of the tricks would benefit immensely from the animating of my figure : if you'd like to animate them, go ahead.

I'd like to make a little remark : our construction of a spanning surface is constructive.
This remark doesn't seem to make sense, doesn't it ?
The confusion comes from the meanings of "constructive" and "non-constructive" in math.
To see what "non-constructive" means, let's look at an example.
Did you know that if you summed up n numbers so that their sum is S, there must be at least one number that is greater or equal to $\frac{S}{n}$? This sounds like a mysterious property, but it's easy to obtain once you consider the "contrapositive".

In math, saying "If A is true, then B is true" is the same as saying "If B is false, then A is false" : if you assume one, then the other must be valid as well (*why ?*). This is called "contraposition".
The contrapositive of our property is : if all n numbers are strictly smaller than $\frac{S}{n}$, then their sum isn't S. Suddenly, the property seems almost obvious : the sum of these n numbers must be strictly smaller (thus, different) than $n \times \frac{S}{n} = S$!
Where's the link to non-constructive math ?
Well, if I picked a list of n numbers and told you nothing but their sum S, could you tell the exact value or rank in the list of the number that's bigger than $\frac{S}{n}$?
With a lot of luck, you could guess it. But the proof of the property we gave didn't contain a systematic technique to find the number(s) that's bigger than $\frac{S}{n}$: it's a non-constructive proof.

Unless you're name is David Hilbert and you're dead, you wont be satisfied by non-constructive proofs. Thankfully our construction of a spanning surface is constructive !
Indeed, our process requires us to find a twisted and folded disk spanning the loop we obtain through the unknotting algorithm. One might think that this requires us to find the moves that change the loop to look like an unknotted circle, which knot theory is still incapable of doing (efficiently). But it doesn't : we knew that our loop was unknotted because we could lift it up. Once lifted, only pairs of points of the loop have the same height : we can connect them by segments, and the collection of these segments draw out our spanning disk. We can then imagine letting go of the loop so that it falls back down to its initial state : the disk then just folds and twists with it, getting us the disk we wanted.
The other tricks are all constructive, so the whole process is too.

A final comment on the unknotting problem. There is a theorem you can find under the name "the Fáry-Milnor theorem" (and more precisely in section 2.3 of Bär's "Differential Geometry") which deserves an expo (perhaps in SoME2) and which gives a basic condition for telling if a loop is actually knotted. This condition can be determined computationally by computing the curvature of the loop ; so you can draw a polygonal loop, smoothen it (with splines ?) and compute an integral, et voilà, you might have saved yourself a lot of time and trouble. Unfortunately, the proof of the theorem (the one I read) is not really constructive ! It requires knowing a way to determine the so-called "bridge number" of a knot, which one can hope to do with approximation methods. Also, the condition given by the theorem is only sufficient, not necessary : if a loop doesn't verify it, you're as clueless as without the theorem.

4 References and recommended reading

I came up with this method of getting spanning surfaces while working on my Bachelors thesis. I don't know if this method is truly original, but I would be surprised if I was the first to construct spanning surfaces this way. It has the advantage of featuring the three following tricks, which I find to be beautiful. Here's where I learnt them from :

- The unknotting algorithm : p.400 of "Introduction to Topology", Adams and Franzosa, Pearson, 2009.
- Switch crossings with bands : p.118 of "Surface knots in 4-space", Kamada, Springer, 2017.
- Pass bands through surface : p.10 of "Slice knots : knot theory in the 4th dimension", Teichner, lecture notes, 2011.

If you want to learn more about knot theory, I recommend the following books :

- "Introduction to Topology", Adams and Franzosa, Pearson, 2009. Chapter 12 is about knots : it is short but complete, requires no prerequisites in topology and hints at applications of knot theory to molecular biology. Ideal for people that want to quickly snoop around knot theory.

— "Knots and links", Cromwell, Cambridge university press, 2004. It has a large selection of topics and develops the prerequisites needed (such as homology) to treat them. Ideal for people with no background in topology that want a detailed overview of knot theory.
— "Quandles" Elhamdadi and Nelson, AMS, 2015. It treats the algebraic part of knot theory, which is left out by Cromwell, and develops all the prerequisites needed. It makes for a great complement to Cromwell's book.
— "Knots and links", Rolfsen, AMS, 1976 (reprint 2003). It covers more advanced topics from knot theory and topology. Many proofs are left as exercise without solutions. Ideal for people that have a background in topology or that are taking classes in (geometric or algebraic) topology : you'll see interesting applications of the topics developed in the traditionally terse/boring introductory courses in topology. For example, it explains how to use the spanning surfaces we constructed to get a covering of the fundamental group of the knot complement (homotopy).

5 Some ideas for math expositions

Here you can find some ideas for topics for expositions or math resources in general. They're underrated classics : I haven't anything to add to them, I just want to point them out as overlooked topics. Of course, this part isn't part of my entry (in the context of SoME1).

I'd like to add something to the "math didactic" recommendations of 3b1b : time and feedback.
Producing good educational resources takes an awful amount of time : consulting many sources, trying to come up with original solutions/proofs/approaches/exercises/motivation, writing all of the details and making figures ; all of this for a few pages or minutes of content.
I believe that a major reason for which bad resources are as bad as they are is that the authors ran out of patience and rushed the production of their work. So please don't rush your work !
Next : always ask for feedback. Feedback has a fancier name : the scientific method. The only way to know if something that seems clear to you has been understood by someone else is by asking them : they may then even tell you precisely what part they don't get and why.
Asking for and getting feedback won't hurt your pride !
Please give me feedback for this document on discord !

5.1 Linear programming with a pen, a ruler and some paper

Consider the following problem : the pi creature owns a food-stand in a busy street of a city ; it sells crêpes and waffles and wants to make a lot of money. When planning it's business, it only considers the price of the dough per unit volume it'll have to buy : let's call these prices c_c for crêpes and c_w for waffles. Pi creature has a budget of b, so that if we denote the quantities of dough q_c for crêpes and q_w for waffles, the quantities it'll buy verify $c_c q_c + c_w q_w \leq b$. Since Pi creature is quite smart, it stores the dough (which is quite liquid) in plastic bags which it puts in a cupboard of volume v : this means that all configurations of quantities have only $q_c + q_w \leq v$ as a storage constraint. Finally, Pi creature sells its delicacies for prices p_c for crêpes and p_w for waffles per unit volume of dough. It always manages to sell all of its products, so that it makes $p_c q_c + p_w q_w$ in revenue from it's buisness.
What quantities q_c and q_w should it buy under these constraints so that it's profit is as big as possible ?
Can Pi creature business be viable ? Does it spend more buying the dough than it makes selling crêpes and waffles ?

This is a classic linear programming problem.
The reason I'm including it here is that it can be solved with nothing but analytic/Cartesian geometry.
Once the connection between equations and places in the plane has been established, one can construct

the segments of constant-cost-levels and see the optimal solution(s). This way of solving the problem is included in almost all introductory textbooks in linear programming.
I'm just pointing out that this is a problem which started out as being purely numerical and who's solution can be obtained by visualising it. All one needs to know is Cartesian geometry and all one needs to solve it for concrete values is some paper, a pen and a ruler (one with a right angle to draw the perpendiculars). So why isn't this taught in highschool ? I remember that we were introduced to vectors and Cartesian geometry with no motivation at all. All the applications we were taught didn't require coordinate systems or where completely useless.
Educators need to know about this almost instant and amazing application that is 2d linear programming !

A last few remarks : one can make the problem more interesting and the solution less intuitive by adding a third constraint. For example, Pi creature could require a certain amount of time per unit volume of dough to produce crêpes or waffles, but only have a limited amount of time to sell them.
For a more real-life-application example, Pi creature could be a farmer choosing between two crops with a budget constraint, an land area constraint and a water constraint.

5.2 Affine cryptography as an introduction to algebra

I was taught about cryptography in highschool and in my first semester undergrad algebra course. More specifically, I was taught about the industry and educational standard, the RSA.
There is however a much simpler cipher that educators often ignore : affine cryptopgraphy.

In affine cryptography, you work with a prime number p of characters and you encrypt your message by slicing it into vectors of n characters x and performing the affine transformation $Ax + b$ where $A \in GL(n, Z/pZ)$.
The purpose of this cipher in an algebra course is that with a little basic number theory (no Euler indicators and Chinese remainders needed here), you can motivate the use of the notion of a field in linear algebra. I'm part of the crowd that finds it dumb to drop the abstract definition of a field in a course that only works on $\mathbb{R}$ and $\mathbb{C}$. Affine cryptography is a remedy to this type of linear algebra course more people should know about.

It can even be tied to more advanced linear algebra by encrypting the nth vector of the message with the nth power of A, motivating the search for normal forms for fast encryption. Also, can you give an explicit description of the keys that are $GL(n, Z/pZ)$?

5.3 Kepler implies Newton

I was taught about Newtons law of gravitation in highschool and ever since, I wondered how Newton came up with it. Sure, you can guess that the force should be inversely proportional to the distance between the attracting objects, but why the square distance ? Was Newton prepared to test all possible rational fractions for inverse proportionality, such as $\frac{1}{r^5+r+42}$, and try to recover Keplers laws with the associated force ?
After all, Kepler got his laws from insane computations.
I couldn't find how Newton discovered his law in any mechanics textbook. Most texts just drop the law and derive Keplers laws from them.

It is however possible to derive Newtons law from Keplers laws ! And for some reason, most mechanics texts don't show this !

You can read about it on my blog : https ://mathsyej875771187.files.wordpress.com/2018/07/gravitation.pdf
The article's in french, but copy-pasting it bit by bit into google translate should do the job for you.

5.4 Get rich quick with statistics and linear algebra

Two "get rich quick" schemes that finally work !
Suppose you want a constant stream of revenue by investing in the stock market, so you want to minimise the risk of your gains as much as possible.
With this goal, it would be a bad idea to invest in both coffee and coffee spoons instead of investing in coffee and tea. Let's explain : if a devastating study on coffee is published, proving that coffee is bad for your health, so that most people switch to drinking tea, you'll get no money with the first investment bundle, but the same amount of money with the second.
And now, substitute milk to tea in our portfolio. Will it increase risk, or diminish it ? After all, some people drink their coffee with milk (yuk !), but others prefer hot chocolate (which is made with milk, if you didn't know) as a substitute for coffee :
To make an informed choice, we could look at the compared yearly gains made by coffee and milk over the past decades : let's denote the values of these gains by (c_i, m_j) and the frequency of occurrence of such a result by $f_{i,j}$.
We then try to find a line that best approximates the relation between the two gains to see how they influence each other.

This is a great way to motivate linear regression. To justify the notion of distance we'll use, one can introduce the notion of variance by its property of being the notion of distance that is minimised by the mean.
Next, the distance is a quadratic form in the parameters of the line. Note that in the case in which the line passes through the origin, there is no need to have strong optimisation techniques at disposal to solve our problem : we just have to bring a polynomial of degree 2 in canonical form.
From this first part, we get the notion of covarinace and it's relation to the notion of risk.
If we then choose how to invest by setting the proportions/weight w_i for an investment X_i (a random variable describing the gains) and we wish to minimise the total risk of the portfolio given by $Var(\sum w_i X_i) = Cov(\sum w_i X_i, \sum w_i X_i)$, we can use the bilinearity of covariance to get a quadratic form to minimise (on a simplex).
Unfortunately, the last step in solving the problem requires some knowledge in optimisation.
The case of a portfolio of size 2 (for example coffee and milk) is still treatable with basic math though : we're minimising a degree 2 polynomial !

I hope you can see why this would make for a great expo : it requires very little prerequisites, it solves a real life and useful problem and it covers 3 math fields in one topic (statistics, (linear) algebra and optimisation).

The second scheme is quite cool as well.
Suppose you're on a financial market : you can buy and sell loans, stocks, or speculative objects (such as land : buy it cheap and hope the neighbourhood becomes popular so as to sell it at a high price). We'll model such a financial service by its cash-flow : service i is represented by the payments (< 0) or gains (> 0) $a_{i,j}$ due in period j.

Here's how to make money out of one's knowledge/information and one's availability. Economists call his "arbitrage". We can try to buy (> 0) or sell (< 0) a quantity x_i of each service i so that we gain money immediately, $\sum_i x_i a_{i,0} > 0$, and so that payments and gains balance each other out in the future periods,

$\sum_i x_i a_{i,j} = 0$ for $j > 0$.
Finding out if this is possible involves solving a system of linear (in)equations, so it can be done with basic linear algebra.
Technically, this is an integer programming problem in it's full and final form. But this easier version of it is already an amazing application that motivates solving big systems of linear equations.
As it is for all expo ideas in this section, this application is missing in most (linear algebra) texts : let's change that.

5.5 The logistic equation

The logistic equation $y' = Cy(N - y)$ makes for a great introduction to differential equations for multiple reasons.
First, it's useful : it can for example describe (quite poorly) the fraction of people in a population infected with a contagious disease (the infection rate is great if there are many people transmitting the disease and if there are many people to be infected).
Next, it's nice to solve : we're looking for derivatives in the equation so that we can solve it by integrating, so a first move could be $\frac{y'}{y(N-y)} = C$; this looks like a derivative of a logarithm and we can get that form by using the decomposition into rational fractions ($\frac{N}{y(N-y)} = \frac{1}{y} + \frac{1}{N-y}$).
Finally, it involves a qualitative study : when dividing by $y(N - y)$, we should justify that a solution to the equation (if there are any) doesn't attain 0 or N (when it starts strictly between these values). Most treatments of the logistic equation skip this step for reasons I don't understand.
As long as $y \in]0, N[$ it's derivative is positive, so it should increase and not attain 0.
The reason for which it shouldn't reach N is harder to understand and doesn't fit our example of a disease spread model. But one can argue as follows : in a continuous deterministic model, there can't be two locally distinct solutions and if a solution does reach N, we expect it to stay at that value since $y' = 0$ from there on. But there is one more way to reach N without violating the differential equation which this last remark hints at : by constantly staying at N. So the solution can't reach N.
There is an argument to get some intuition on why there can be at most one solution passing through the same point at the same time. We find it intuitive that a solution to such a deterministic equation shouldn't start to bifurcate at some point of time, but why shouldn't two solutions with different starting conditions be able to meet at some point in time ?
If they met at time T then the solutions $z(t) = y(T-t)$ (the quantity now evolves back in time) would still satisfy a deterministic differential equation (here, $z' = -Cz(N - z)$) and have the same initial conditions $z(0) = y(T)$ but would bifurcate to different $z(T)$. This contradicts our previous intuition.

More formally :
A constant solution $z(x) = N$ solves the equation ; yet, two solutions differ by an additive constant ; if y would reach N, y would have to be the constant solution since the additive constant would be found to be 0 by evaluating in at the time y hits N ; this can't be the case if we start with $y \in]0, N[$. This argument can be adapted to get the reason for which y doesn't reach 0.

Why do so many texts refuse to include this argument ?