

The Hardest Math I've Ever Really Used

Picture credits. Mona: Leonardo; Map 1: en.wikipedia.org/wiki/Greenhouse_gas; Smokestacks: gbuapcd.org/complaint.htm; Penguin: brentpabst.com/bp/2007/12/15/BrentGoesPenguin.aspx; Map 2: flightpedia.org; Segway: co2calculator.wordpress.com/2008/10;

Dror Bar-Natan, Kansas State April 7 2009, <http://www.math.toronto.edu/~drorbn/Talks/KSU-090407>

Abstract. What's the hardest math I've ever used in real life? Me, myself, directly — not by using a cellphone or a GPS device that somebody else designed. And in "real life" - not while studying or teaching mathematics?

I use addition and subtraction daily, adding up bills or calculating change. I use percentages often, though mostly it is just "add 15 percents". I seldom use multiplication and division: when I buy in bulk, or when I need to know how many tiles I need to replace my kitchen floor. I've used powers twice in my life, doing calculations related to mortgages. I've used a tiny bit of 2×2 linear algebra for a tiny bit of non-math-related computer graphics I've played with. And for a long time, that was all. In my talk I will tell you how recently a math topic discovered only in the 1800s made a brief and modest appearance in my non-mathematical life. There are many books devoted to that topic and a lot of active research. Yet for all I know, nobody ever needed the actual gory formulas for such a simple reason before.

I could be a Mathematician...

Non-Commutative Gaussian Elimination and Rubik's Cube

The Problem. Let $G = \langle g_1, \dots, g_n \rangle$ be a subgroup of S_n , with $n = O(100)$. Before you die, understand G .

1. Compute $|G|$.
2. Given $\sigma \in S_n$, decide if $\sigma \in G$.
3. Write a $\sigma \in G$ in terms of g_1, \dots, g_n .
4. Produce *random* elements of G .

The Commutative Analog. Let $V = \text{span}\{v_1, \dots, v_n\}$ be a subspace of \mathbb{R}^n . Before you die, understand V .

Solution: Gaussian Elimination. Prepare an empty table.

1	2	3	4	...	n-1	n
---	---	---	---	-----	-----	---

Space for a vector $u_i \in V$, of the form $u_i = (0, 0, 0, \dots, 1, \dots, 0)$: $i :=$ "the pivot"

Feed v_1, \dots, v_n , in order. To feed a non-zero v , find its pivotal position i .

1. If box i is empty, put v there.
2. If box i is occupied, find a combination v' of v and u_i that eliminates the pivot, and feed v' .

Non-Commutative Gaussian Elimination
Prepare a mostly-empty table.

1	2	3	4	...	n-1	n
---	---	---	---	-----	-----	---

Space for a $\sigma_{i,j} \in S_n$ of the form $(1, 2, \dots, i-2, i-1, j, s, s, \dots, s)$
So $\sigma_{i,j}$ fixes $1, \dots, i-1$, sends "the pivot" i to j and goes wild afterwards, and $\sigma_{i,j}^{-1}$ "does sticker".

Feed g_1, \dots, g_n in order. To feed a non-identity σ , find its pivotal position i and let $j := \sigma(i)$.

1. If box (i, j) is empty, put σ there.
2. If box (i, j) contains $\sigma_{i,j}$, feed $\sigma' := \sigma_{i,j}^{-1}\sigma$.

The Twist. When done, for every occupied (i, j) and (k, l) , feed $\sigma_{i,j}\sigma_{k,l}$. Repeat until the table stops changing.

Claim. The process stops in our lifetimes, after at most $O(n^6)$ operations. Call the resulting table T .

Claim. Anything fed in T is a monotone product in G : f was fed $\Rightarrow f \in M_1 := \langle \sigma_{i,j}\sigma_{k,l} \dots \sigma_{a,b} : \forall i, j, k \geq i \ \& \ \sigma_{i,j} \in T \rangle$

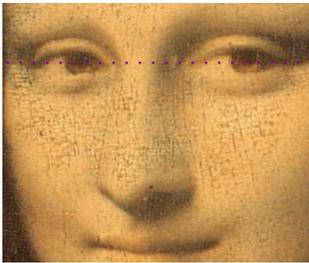
Homework Problem 1. Homework Problem 2.
Can you do cosets? Can you do categories (groupoids)?

The Results

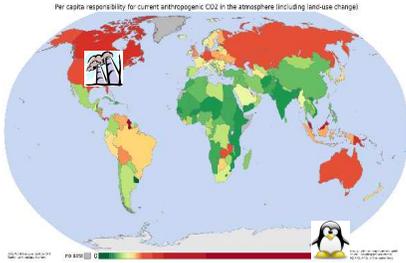
```
In[3]:= {Feed[8]}; Product[1 + Length[Select[Range[n], Head[#, #] == P &]], {i, n}] & /@ # &
Out[3]:= {4, 16, 1599935016986000, 21119142223872000, 4325200327489856000, 4325200327489856000}
```

<http://www.math.toronto.edu/~drorbn/Talks/CUMC-0907> and <http://www.math.toronto.edu/~drorbn/Misc/SchrienerSimsRubik/>

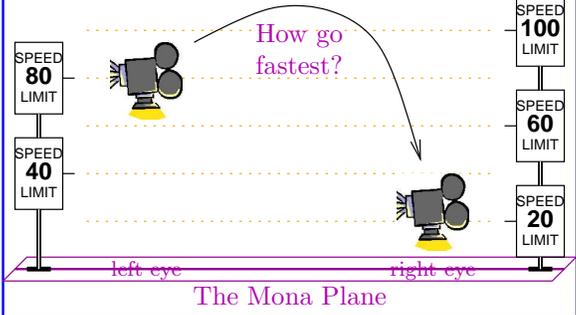
...or an Art Historian...



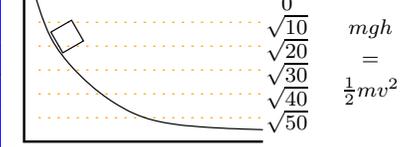
...or an Environmentalist.



Goal. Find the least-blur path to go from Mona's left eye to Mona's right eye in fixed time. Alternatively, fix your blur-tolerance, and find the fastest path to do the same. For fixed blur, our camera moves at a speed proportional to its distance from the image plane:



The brachistochrone

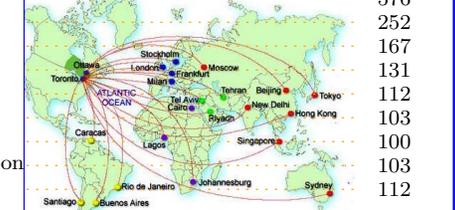


Bernoulli on Newton. "I recognize the lion by his paw".

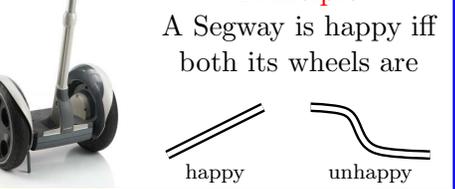
ParametricPlot3D[
Sin[u] Cos[v],
Sin[u] Sin[v],
Cos[u]
, {u, 0, pi}, {v, 0, 2pi}]

ParametricPlot3D[
Sech[u] Cos[v],
Sech[u] Sin[v],
u - Tanh[u]
, {u, 0, e}, {v, 0, 2pi}]

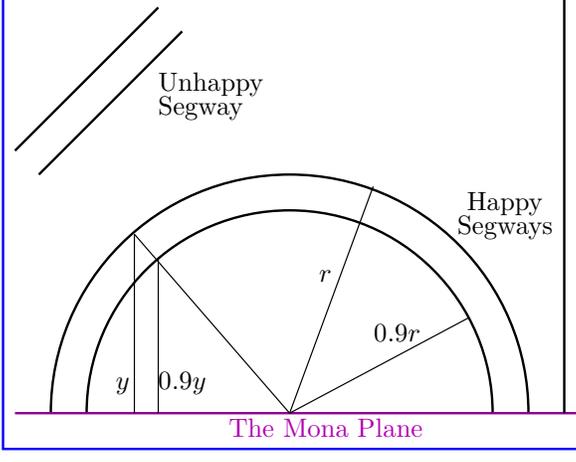
Flatlanders airline route map



The Happy Segway Principle



Happy camera-carrying Segways above the Mona Plane.



The Lobachevsky Space

Two parallels through one point

Parametrization

$$\theta'(t) = \sin \theta(t)$$

$$\Downarrow$$

$$\theta = 2 \arctan e^t$$

Rotations, translations and some basic geometry also occur

The Actual Code

```
x = p1.x-p2.x; y = p1.y-p2.y;
d1 = p1.d; d2 = p2.d;
norm = sqrt(x*x + y*y);
a = x/norm; b = y/norm;
x1p = a*x + b*y;
x0 = (x1p + (d1*d1-d2*d2)/x1p)/2;
r = sqrt((x1p-x0)*(x1p-x0)+d1*d1);
x1pp = (x1p-x0)/r; x2pp = -x0/r;
theta1 = acos(x1pp);
theta2 = acos(x2pp);
t1 = log(tan(theta1/2));
t2 = log(tan(theta2/2));
t3 = t1 + s*(t2-t1);
theta3 = 2*atan(exp(t3));
x3pp = cos(theta3);
d3pp = sin(theta3);
x3p = x0 + r*x3pp;
p3.d = r*d3pp;
p3.x = p2.x + a*x3p;
p3.y = p2.y + b*x3p;
```

Ops used. +, -, x, /, sqrt, cos, sin, tan, arccos, arctan, log, exp.