Artificial intelligence for the working mathematician

Lecture 1: where we come from, and what happened

Mini-course on emerging applications of AI
in theoretical mathematics
ICMAT, Madrid, Spain

Damien Galant

Department of Mathematics
Brown University

Monday 14 September 2026

Plan of the mini-course

1. **Where we come from, and what happened.**
   Machines have been doing mathematics for a long time. What is genuinely new, and why it concerns you.

Plan of the mini-course

1. **Where we come from, and what happened.**
   Machines have been doing mathematics for a long time. What is genuinely new, and why it concerns you.

2. **What it looks like in practice.**
   Live demonstrations. Agents, code, search at scale.

Plan of the mini-course

1. **Where we come from, and what happened.**
   Machines have been doing mathematics for a long time. What is genuinely new, and why it concerns you.

2. **What it looks like in practice.**
   Live demonstrations. Agents, code, search at scale.

3. **What it does to us.**
   More practice, then: what this changes for our community — publishing, refereeing, training students — and the objections I take seriously.

Plan of the mini-course

1. **Where we come from, and what happened.**
   Machines have been doing mathematics for a long time. What is genuinely new, and why it concerns you.

2. **What it looks like in practice.**
   Live demonstrations. Agents, code, search at scale.

3. **What it does to us.**
   More practice, then: what this changes for our community — publishing, refereeing, training students — and the objections I take seriously.

One request

Please interrupt. This is a course, not a seminar talk, and I would much rather answer a question than finish a slide.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.
- Now a postdoc at Brown University, with *Javier Gómez-Serrano*.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.
- Now a postdoc at Brown University, with *Javier Gómez-Serrano*.
- I have been working on AI and mathematics for *about a year*.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.
- Now a postdoc at Brown University, with *Javier Gómez-Serrano*.
- I have been working on AI and mathematics for *about a year*.

The relevant disclosure

I identify as a mathematician, not as an AI researcher. I am here as a user of these tools, talking to other potential users.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.
- Now a postdoc at Brown University, with *Javier Gómez-Serrano*.
- I have been working on AI and mathematics for *about a year*.

The relevant disclosure

I identify as a mathematician, not as an AI researcher. I am here as a user of these tools, talking to other potential users.

This is the whole point of the course: everything I will show you, I learned in that year, *while doing my own mathematics*.

Who is speaking, and why it matters

- PhD in Mons (UMONS) and Valenciennes (UPHF), 2024.
  Nonlinear Schrödinger equations, variational methods, concentration-compactness, metric graphs.
- Now a postdoc at Brown University, with *Javier Gómez-Serrano*.
- I have been working on AI and mathematics for *about a year*.

The relevant disclosure

I identify as a mathematician, not as an AI researcher. I am here as a user of these tools, talking to other potential users.

These slides included. They were drafted with these tools — prose, TEX, figures, references. Every claim is mine, and so is every error.

This is the whole point of the course: everything I will show you, I learned in that year, *while doing my own mathematics*.

What this course is and is not

It is

- a report from a user;
- concrete, with live demonstrations;
- about *your* mathematics;
- opinionated. I will tell you when.

What this course is and is not

It is

- a report from a user;
- concrete, with live demonstrations;
- about *your* mathematics;
- opinionated. I will tell you when.

It is not

- a machine learning course;
- a survey of the literature;
- a sales pitch — lecture 3 is largely about what goes wrong;
- about replacing you.

What this course is and is not

It is

- a report from a user;
- concrete, with live demonstrations;
- about *your* mathematics;
- opinionated. I will tell you when.

It is not

- a machine learning course;
- a survey of the literature;
- a sales pitch — lecture 3 is largely about what goes wrong;
- about replacing you.

Warning

Anything I say about what these systems *currently* do is a snapshot, and it dates quickly. Take the specifics as "true in September 2026"; the general points should age better.

The thread of the three lectures

Machines have helped us for seventy years, in *three roles*:

The thread of the three lectures

Machines have helped us for seventy years, in *three roles*:

The machine as explorer. (§2)

It computes examples; you look at them and conjecture.

The thread of the three lectures

Machines have helped us for seventy years, in *three roles*:

The machine as explorer. (§2)

It computes examples; you look at them and conjecture.

The machine as verifier. (§3)

It checks an enormous case analysis that you designed.

The thread of the three lectures

Machines have helped us for seventy years, in *three roles*:

The machine as explorer. (§2)
It computes examples; you look at them and conjecture.

The machine as verifier. (§3)
It checks an enormous case analysis that you designed.

The machine as collaborator. (§4)
It proposes *candidates*: arguments, references, constructions.
Choosing the question and judging the answer stay yours.

The thread of the three lectures

Machines have helped us for seventy years, in *three roles*:

The machine as explorer. (§2)
It computes examples; you look at them and conjecture.

The machine as verifier. (§3)
It checks an enormous case analysis that you designed.

The machine as collaborator. (§4)
It proposes *candidates*: arguments, references, constructions.
Choosing the question and judging the answer stay yours.

Why start with the old stuff

The first two were *also* controversial when they appeared, for *some of the same* reasons — and both are standard today, if not universally loved. What is new is the rest: authorship, reproducibility, and who owns the tools.

Exploration has always been part of the job

Before there were machines, there were *computers* — and they were us.

Harvard College Observatory, c. 1890–1900.
Public domain.

Exploration has always been part of the job

Before there were machines, there were *computers* — and they were us.

Gauss later recalled that tables of primes led him, at fifteen, to guess that $\pi(x)$, the number of primes up to x, grows like $x/\log x$.

Harvard College Observatory, c. 1890–1900. Public domain.

Exploration has always been part of the job

Before there were machines, there were *computers* — and they were us.

Gauss later recalled that tables of primes led him, at fifteen, to guess that $\pi(x)$, the number of primes up to x, grows like $x/\log x$.

Riemann states *the* hypothesis in 1859, having "provisionally put aside" the search for a proof.

Harvard College Observatory, c. 1890–1900. Public domain.

Exploration has always been part of the job

Before there were machines, there were *computers* — and they were us.

Gauss later recalled that tables of primes led him, at fifteen, to guess that $\pi(x)$, the number of primes up to x, grows like $x/\log x$.

Riemann states *the* hypothesis in 1859, having "provisionally put aside" the search for a proof.

In 1932 **Siegel** goes through the unpublished papers and finds *pages of computation* of zeros.

Harvard College Observatory, c. 1890–1900. Public domain.

Exploration has always been part of the job

Before there were machines, there were *computers* — and they were us.

Gauss later recalled that tables of primes led him, at fifteen, to guess that $\pi(x)$, the number of primes up to x, grows like $x/\log x$.

Riemann states *the* hypothesis in 1859, having "provisionally put aside" the search for a proof.

In 1932 **Siegel** goes through the unpublished papers and finds *pages of computation* of zeros.

Harvard College Observatory, c. 1890–1900. Public domain.

Riemann did not only guess: *his notes show computation alongside the conjecture.*

Turing buys machinery for a conjecture

- 1939: Turing designs a *mechanical* zeta-function machine — gears and weights — to compute zeros. The war interrupts it.

Turing buys machinery for a conjecture

- 1939: Turing designs a *mechanical* zeta-function machine — gears and weights — to compute zeros. The war interrupts it.
- 1950: he runs the computation on the *Manchester Mark 1*, one of the first stored-program computers on Earth, and checks the Riemann hypothesis as far up the line $\Re s = \frac{1}{2}$ as the machine could reach.

Turing buys machinery for a conjecture

- 1939: Turing designs a *mechanical* zeta-function machine — gears and weights — to compute zeros. The war interrupts it.
- 1950: he runs the computation on the *Manchester Mark 1*, one of the first stored-program computers on Earth, and checks the Riemann hypothesis as far up the line $\Re s = \frac{1}{2}$ as the machine could reach.
- Published in 1953: *Some calculations of the Riemann zeta-function*.

Turing buys machinery for a conjecture

- 1939: Turing designs a *mechanical* zeta-function machine — gears and weights — to compute zeros. The war interrupts it.
- 1950: he runs the computation on the *Manchester Mark 1*, one of the first stored-program computers on Earth, and checks the Riemann hypothesis as far up the line $\Re s = \frac{1}{2}$ as the machine could reach.
- Published in 1953: *Some calculations of the Riemann zeta-function*.

The point

One of the very first uses of a general-purpose computer, by one of the people who invented the idea, was *to look for evidence about an open problem in pure mathematics.*

Turing buys machinery for a conjecture

- 1939: Turing designs a *mechanical* zeta-function machine — gears and weights — to compute zeros. The war interrupts it.
- 1950: he runs the computation on the *Manchester Mark 1*, one of the first stored-program computers on Earth, and checks the Riemann hypothesis as far up the line $\Re s = \frac{1}{2}$ as the machine could reach.
- Published in 1953: *Some calculations of the Riemann zeta-function*.

The point

One of the very first uses of a general-purpose computer, by one of the people who invented the idea, was *to look for evidence about an open problem in pure mathematics.*

Note that this proves nothing, and everyone knew it: Turing was hoping to find a *counterexample*.

Fermi–Pasta–Ulam–Tsingou, 1953

Los Alamos, on the MANIAC. Fermi wanted to *watch* a foundational belief of statistical mechanics happen, numerically.

Fermi–Pasta–Ulam–Tsingou, 1953

Los Alamos, on the MANIAC. Fermi wanted to *watch* a foundational belief of statistical mechanics happen, numerically.

$N-1$ masses on a line with fixed ends, $q_0 = q_N = 0$, nearest-neighbour springs and a small quadratic correction to the force: for $1 \leq j \leq N-1$,

$$\ddot{q}_j = (q_{j+1} - 2q_j + q_{j-1}) + \alpha\left[(q_{j+1} - q_j)^2 - (q_j - q_{j-1})^2\right].$$

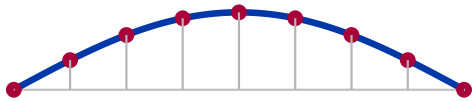
Fermi–Pasta–Ulam–Tsingou, 1953

Los Alamos, on the MANIAC. Fermi wanted to *watch* a foundational belief of statistical mechanics happen, numerically.

$N-1$ masses on a line with fixed ends, $q_0 = q_N = 0$, nearest-neighbour springs and a small quadratic correction to the force: for $1 \le j \le N-1$,

$$\ddot{q}_j = (q_{j+1} - 2q_j + q_{j-1}) + \alpha\left[(q_{j+1} - q_j)^2 - (q_j - q_{j-1})^2\right].$$

For $\alpha = 0$ the chain *decouples* into independent oscillators — the *normal modes*, $q_j \propto \sin(\pi kj/N)$ — each conserving its own energy. All the energy in mode 1 stays there forever.



mode 1: $\sin(\pi j/N)$

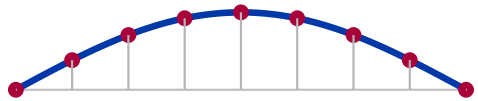
Fermi–Pasta–Ulam–Tsingou, 1953

Los Alamos, on the MANIAC. Fermi wanted to *watch* a foundational belief of statistical mechanics happen, numerically.

$N-1$ masses on a line with fixed ends, $q_0 = q_N = 0$, nearest-neighbour springs and a small quadratic correction to the force: for $1 \le j \le N-1$,

$$\ddot{q}_j = (q_{j+1} - 2q_j + q_{j-1}) + \alpha\left[(q_{j+1} - q_j)^2 - (q_j - q_{j-1})^2\right].$$

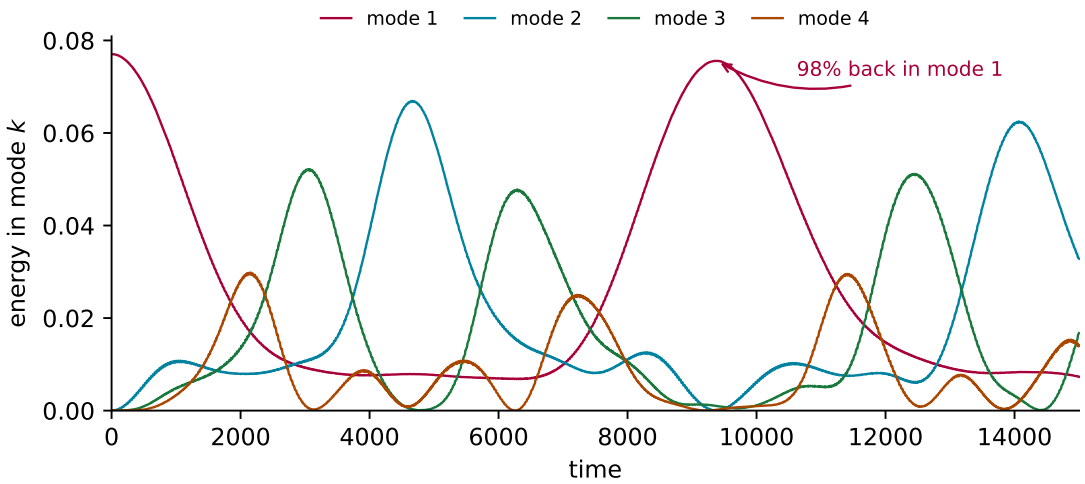
For $\alpha = 0$ the chain *decouples* into independent oscillators — the *normal modes*, $q_j \propto \sin(\pi kj/N)$ — each conserving its own energy. All the energy in mode 1 stays there forever.



mode 1: $\sin(\pi j/N)$

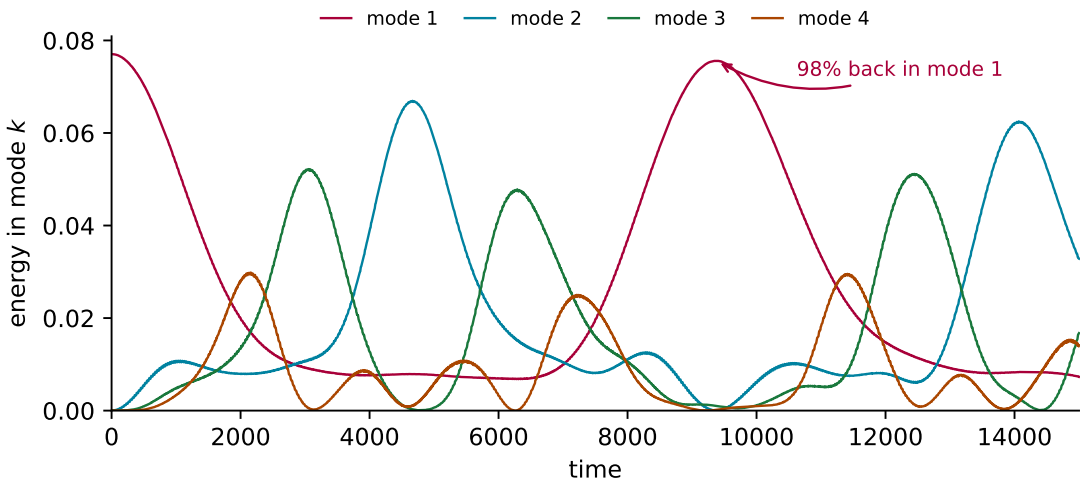
Switch on a small α. Statistical mechanics says the energy should end up *shared out roughly equally* among the modes, and stay that way: *they ran it to watch equipartition happen.*

What actually happened



Report LA-1940 (1955), recomputed with $N = 32$, $\alpha = 1/4$, all the energy initially in mode 1.

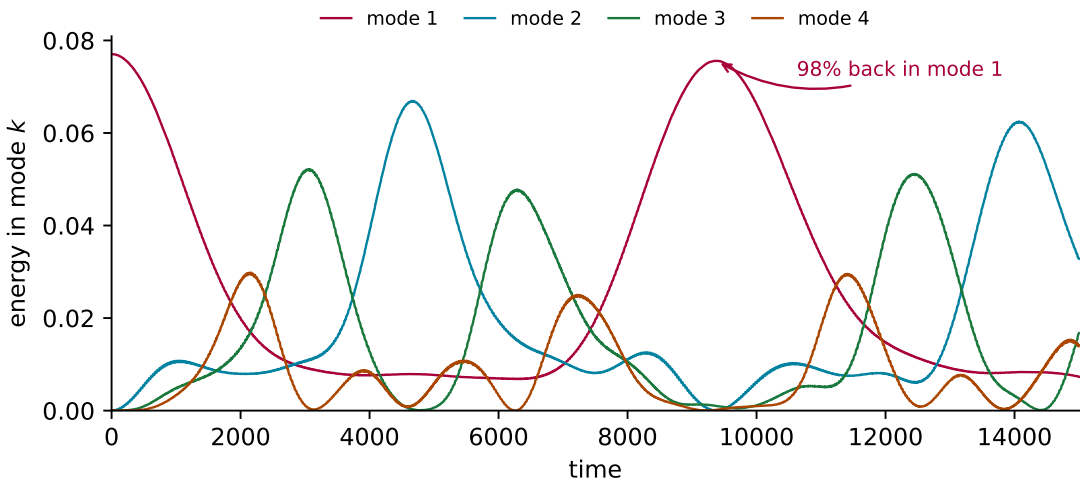
What actually happened



Report LA-1940 (1955), recomputed with $N = 32$, $\alpha = 1/4$, all the energy initially in mode 1.

The energy leaves mode 1, spreads over a few modes — and comes back: 98% *returns to mode 1*. No equipartition on the timescale observed.

What actually happened



Report LA-1940 (1955), recomputed with $N = 32$, $\alpha = 1/4$, all the energy initially in mode 1.

The energy leaves mode 1, spreads over a few modes — and comes back: 98% *returns to mode 1*. No equipartition on the timescale observed.

Twelve years later, Zabusky and Kruskal notice that the continuum limit is the *KdV equation*. Solving it *numerically*, they see localised waves re-emerge from a collision with the same shape and speed, up to a phase shift — *solitons*. Modern soliton theory takes off here.

Machines killing conjectures

Euler, 1769: a sum of $n-1$ or fewer n-th powers is never an n-th power.

Machines killing conjectures

Euler, 1769: a sum of $n-1$ or fewer n-th powers is never an n-th power.

Two hundred years later, Lander and Parkin run a direct search on a CDC 6600. *The entire paper:*

COUNTEREXAMPLE TO EULER'S CONJECTURE ON SUMS OF LIKE POWERS

BY L. J. LANDER AND T. R. PARKIN

Communicated by J. D. Swift, June 27, 1966

A direct search on the CDC 6600 yielded

$$27^5 + 84^5 + 110^5 + 133^5 = 144^5$$

as the smallest instance in which four fifth powers sum to a fifth power. This is a counterexample to a conjecture by Euler [1] that at least n nth powers are required to sum to an nth power, $n > 2$.

REFERENCE

1. L. E. Dickson, *History of the theory of numbers*, Vol. 2, Chelsea, New York, 1952, p. 648.

Bull. Amer. Math. Soc. **72** (1966), 1079.

. . . but brute force is rarely the point

For $n = 4$ the counterexample is far out of reach of a direct search.

. . . but brute force is rarely the point

For $n = 4$ the counterexample is far out of reach of a direct search.

Elkies (1988) uses *elliptic curves* to cut the search space down to something feasible, and then searches:

$$2\,682\,440^4 + 15\,365\,639^4 + 18\,796\,760^4 = 20\,615\,673^4.$$

. . . but brute force is rarely the point

For $n = 4$ the counterexample is far out of reach of a direct search.

Elkies (1988) uses *elliptic curves* to cut the search space down to something feasible, and then searches:

$$2\,682\,440^4 + 15\,365\,639^4 + 18\,796\,760^4 = 20\,615\,673^4.$$

Remember this one

Elkies did not replace thinking by computing. He *used* thinking to make it possible. *Good computer-assisted results often look like this.*

. . . but brute force is rarely the point

For $n = 4$ the counterexample is far out of reach of a direct search.

Elkies (1988) uses *elliptic curves* to cut the search space down to something feasible, and then searches:

$$2\,682\,440^4 + 15\,365\,639^4 + 18\,796\,760^4 = 20\,615\,673^4.$$

Remember this one

Elkies did not replace thinking by computing. He *used* thinking to make it possible. *Good computer-assisted results often look like this.*

We will meet exactly this shape again on Tuesday — with tools that write the search for you.

Conjectures built out of data

- **Birch and Swinnerton-Dyer** (early 1960s, on the EDSAC): for an elliptic curve they compute $\prod_{p \leq X} N_p/p$, where N_p counts the points of the curve modulo p, plot how it grows, and read a conjecture off the picture — now a Millennium Problem.

Conjectures built out of data

- **Birch and Swinnerton-Dyer** (early 1960s, on the EDSAC): for an elliptic curve they compute $\prod_{p \leq X} N_p/p$, where N_p counts the points of the curve modulo p, plot how it grows, and read a conjecture off the picture — now a Millennium Problem.
- **Integer relation algorithms** detect a linear relation with integer coefficients among given numbers. In 1995 Bailey, Borwein and Plouffe *find numerically*

$$\pi = \sum_{k \geq 0} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right),$$

giving the n-th hexadecimal digit of π without the previous ones. Found by search; proved easily afterwards.

Conjectures built out of data

- **Birch and Swinnerton-Dyer** (early 1960s, on the EDSAC): for an elliptic curve they compute $\prod_{p \leq X} N_p/p$, where N_p counts the points of the curve modulo p, plot how it grows, and read a conjecture off the picture — now a Millennium Problem.
- **Integer relation algorithms** detect a linear relation with integer coefficients among given numbers. In 1995 Bailey, Borwein and Plouffe *find numerically*

$$\pi = \sum_{k \geq 0} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right),$$

  giving the n-th hexadecimal digit of π without the previous ones. Found by search; proved easily afterwards.

In 1992 all of this got its own journal: *Experimental Mathematics*. It is still running.

Interlude: what exploration costs

Every story so far has the same shape:

guess $\longrightarrow$ test it on examples $\longrightarrow$ understand the failure $\longrightarrow$ better guess

Gauss on tables of primes, Riemann on his zeros, Fermi on his chain.

Interlude: what exploration costs

Every story so far has the same shape:

guess $\longrightarrow$ test it on examples $\longrightarrow$ understand the failure $\longrightarrow$ better guess

Gauss on tables of primes, Riemann on his zeros, Fermi on his chain.

Nothing about that shape has changed in two hundred years. What changes is *what one round of it costs* — and the testing step has always been the expensive one.

Riemann needed a new asymptotic formula before he could compute at all; Fermi needed one of the few computers on Earth.

Interlude: what exploration costs

Every story so far has the same shape:

guess $\longrightarrow$ test it on examples $\longrightarrow$ understand the failure $\longrightarrow$ better guess

Gauss on tables of primes, Riemann on his zeros, Fermi on his chain.

Nothing about that shape has changed in two hundred years. What changes is *what one round of it costs* — and the testing step has always been the expensive one.

Riemann needed a new asymptotic formula before he could compute at all; Fermi needed one of the few computers on Earth.

What I will show you on Tuesday

One round of this has become much cheaper, and much more accessible — for instance, it can now involve code you would not have written yourself.

Computer-assisted *proofs*: also not new, also not AI

Second role. Here the machine is not suggesting anything. You have reduced your theorem to a *finite* verification, and the verification is too large for a human.

Computer-assisted *proofs*: also not new, also not AI

Second role. Here the machine is not suggesting anything. You have reduced your theorem to a *finite* verification, and the verification is too large for a human.

The two hard parts

(i) *the reduction* — entirely human, and where the mathematics is;

(ii) *trusting the machine* — rounding errors, bugs, "did it check what I think it checked?".

Computer-assisted *proofs*: also not new, also not AI

Second role. Here the machine is not suggesting anything. You have reduced your theorem to a *finite* verification, and the verification is too large for a human.

The two hard parts

(i) *the reduction* — entirely human, and where the mathematics is;

(ii) *trusting the machine* — rounding errors, bugs, "did it check what I think it checked?".

The community has spent fifty years building a culture around (ii). *We will need the same for the new tools, and it does not exist yet.*

The four colour theorem (1976)

Appel and Haken produce 1834 configurations: every potential counterexample must contain one, and none of them can occur. A computer checks them, over a thousand hours.

The four colour theorem (1976)

Appel and Haken produce 1834 configurations: every potential counterexample must contain one, and none of them can occur. A computer checks them, over a thousand hours.

The reaction was not enthusiasm.

- "A proof must be *surveyable*" (Tymoczko, 1979): no human can check this, so is it a proof at all?
- "It gives no *understanding* of why four colours suffice."
- "How do we know the program is correct?"

The four colour theorem (1976)

Appel and Haken produce 1834 configurations: every potential counterexample must contain one, and none of them can occur. A computer checks them, over a thousand hours.

The reaction was not enthusiasm.

- "A proof must be *surveyable*" (Tymoczko, 1979): no human can check this, so is it a proof at all?
- "It gives no *understanding* of why four colours suffice."
- "How do we know the program is correct?"

Answers, eventually

Robertson–Sanders–Seymour–Thomas (1997) gave a cleaner, independently implemented proof; Gonthier (2005) a *fully formal proof in Coq*, shrinking what must be trusted to the statement and a small kernel.

The four colour theorem (1976)

Appel and Haken produce 1834 configurations: every potential counterexample must contain one, and none of them can occur. A computer checks them, over a thousand hours.

The reaction was not enthusiasm.

- "A proof must be *surveyable*" (Tymoczko, 1979): no human can check this, so is it a proof at all?
- "It gives no *understanding* of why four colours suffice."
- "How do we know the program is correct?"

Answers, eventually

Robertson–Sanders–Seymour–Thomas (1997) gave a cleaner, independently implemented proof; Gonthier (2005) a *fully formal proof in Coq*, shrinking what must be trusted to the statement and a small kernel.

Fifty years on, nobody doubts the theorem.

Rigorous numerics: intervals instead of floats

The idea, popularised by Moore's 1966 book: compute with *sets that provably contain* the number, rounding the endpoints *outwards*.

Rigorous numerics: intervals instead of floats

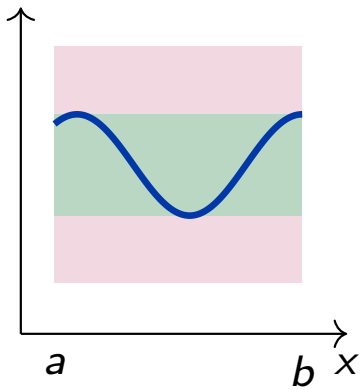
The idea, popularised by Moore's 1966 book: compute with *sets that provably contain* the number, rounding the endpoints *outwards*.

$$[a, b] + [c, d] = [a + c,\ \ b + d],$$
$$[a, b] \cdot [c, d] = [\min(ac, ad, bc, bd),\ \ \max(ac, ad, bc, bd)].$$

Rigorous numerics: intervals instead of floats

The idea, popularised by Moore's 1966 book: compute with *sets that provably contain* the number, rounding the endpoints *outwards*.

$$[a,b] + [c,d] = [a + c,\ \ b + d],$$
$$[a,b] \cdot [c,d] = [\min(ac, ad, bc, bd),\ \ \max(ac, ad, bc, bd)].$$



true range of f

what the computation returns

Consequence

If it returns $[0.31, 0.34]$ and you needed positivity, *you have a theorem* — not a numerical indication.

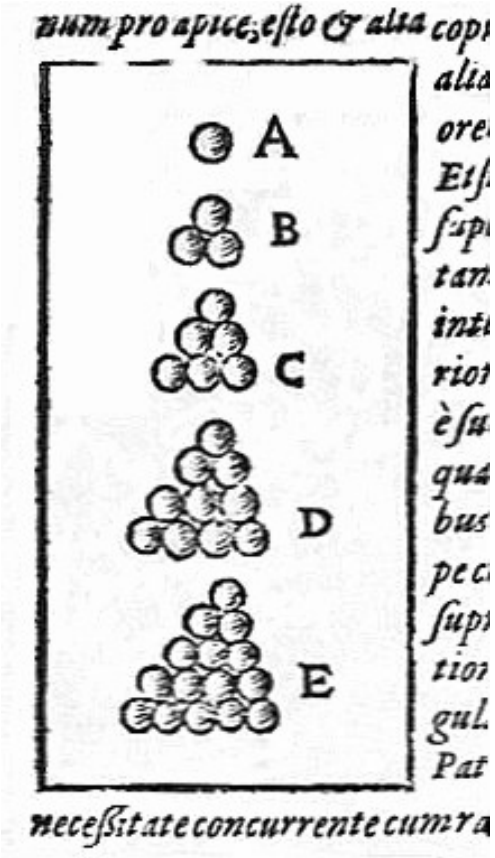
The catch: the enclosure is always correct, but can be uselessly wide.

What rigorous numerics has proved

- **Tucker (1998–2002)**: the Lorenz attractor really is a strange attractor — *Smale's 14th problem*, settled with interval arithmetic.

What rigorous numerics has proved

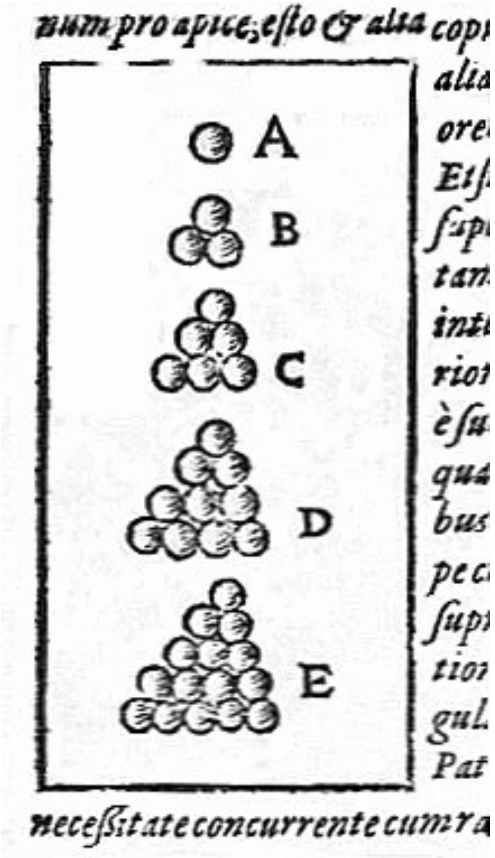
- **Tucker (1998–2002)**: the Lorenz attractor really is a strange attractor — *Smale's 14th problem*, settled with interval arithmetic.
- **Hales (1998–2014)**: the Kepler conjecture — no sphere packing beats the greengrocer's. The *Annals* referees declared themselves "99% certain" and stopped; Hales answered with *Flyspeck*.



Kepler's own figure, *Strena Seu de Nive Sexangula* (1611). Public domain.

What rigorous numerics has proved

- **Tucker (1998–2002)**: the Lorenz attractor really is a strange attractor — *Smale's 14th problem*, settled with interval arithmetic.
- **Hales (1998–2014)**: the Kepler conjecture — no sphere packing beats the greengrocer's. The *Annals* referees declared themselves "99% certain" and stopped; Hales answered with *Flyspeck*.
- **Fluid dynamics**: blow-up for 3D Euler *with boundary* (Chen–Hou — their part II is titled *Rigorous Numerics*), and much more by *Javier Gómez-Serrano* and collaborators.



Kepler's own figure, *Strena Seu de Nive Sexangula* (1611). Public domain.

What a SAT solver is

Input: a Boolean formula in conjunctive normal form — variables $x_1, \ldots, x_n$, each true or false, and an *and* of *or*s:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge \cdots$$

Output: an assignment making it true, or the answer that none exists.

What a SAT solver is

Input: a Boolean formula in conjunctive normal form — variables $x_1, \ldots, x_n$, each true or false, and an *and* of *or*s:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge \cdots$$

Output: an assignment making it true, or the answer that none exists.

The NP-complete problem (Cook 1971, Levin 1973): 2^n assignments, and no polynomial-time algorithm known. *And yet* solvers settle instances with millions of variables — real instances are not worst cases.

What a SAT solver is

Input: a Boolean formula in conjunctive normal form — variables $x_1, \ldots, x_n$, each true or false, and an *and* of *or*s:

$$(x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee x_3) \wedge \cdots$$

Output: an assignment making it true, or the answer that none exists.

The NP-complete problem (Cook 1971, Levin 1973): 2^n assignments, and no polynomial-time algorithm known. *And yet* solvers settle instances with millions of variables — real instances are not worst cases.

How a mathematician uses one

Let x_i mean "i is red". For each Pythagorean triple (a, b, c) with $c \leq n$, add

$$(x_a \vee x_b \vee x_c) \wedge (\neg x_a \vee \neg x_b \vee \neg x_c),$$

which forbid an all-red and an all-blue triple. A satisfying assignment is a colouring; *unsatisfiable is a theorem.*

Enormous searches: SAT solvers

Theorem (Heule–Kullmann–Marek, 2016)

$\{1, \ldots, 7824\}$ *splits into two parts with no Pythagorean triple inside a part.* $\{1, \ldots, 7825\}$ *does not.*

Enormous searches: SAT solvers

Theorem (Heule–Kullmann–Marek, 2016)

$\{1, \ldots, 7824\}$ *splits into two parts with no Pythagorean triple inside a part.* $\{1, \ldots, 7825\}$ *does not.*

The proof is a certificate of about *200 terabytes* — at the time, the largest proof ever produced.

Enormous searches: SAT solvers

Theorem (Heule–Kullmann–Marek, 2016)

$\{1, \ldots, 7824\}$ *splits into two parts with no Pythagorean triple inside a part.* $\{1, \ldots, 7825\}$ *does not.*

The proof is a certificate of about *200 terabytes* — at the time, the largest proof ever produced.

Who checks 200 terabytes?

Nobody reads it. The solver emits a *certificate*, and a small, independently written (by now formally verified) checker replays it. *You trust the checker and the encoding, not the solver.*

Formal proof assistants

State the theorem, and write the proof, in a language a machine checks — every step, all the way down to the axioms. Coq, Isabelle, *Lean*.

Formal proof assistants

State the theorem, and write the proof, in a language a machine checks — every step, all the way down to the axioms. Coq, Isabelle, *Lean*.
The trust argument is the one from the last slide, and it *shrinks* the trusted base rather than removing it: you still trust the *statement, the definitions and the axioms* — but the proof itself is checked by a kernel of a few thousand lines.

Formal proof assistants

State the theorem, and write the proof, in a language a machine checks — every step, all the way down to the axioms. Coq, Isabelle, *Lean*.
The trust argument is the one from the last slide, and it *shrinks* the trusted base rather than removing it: you still trust the *statement, the definitions and the axioms* — but the proof itself is checked by a kernel of a few thousand lines.

The price: you must build the library first

Every definition you use has to exist already. *Mathlib*, Lean's library, is that effort — and it is why formalising a paper is now sometimes a matter of weeks rather than years.

Formal proof assistants

State the theorem, and write the proof, in a language a machine checks — every step, all the way down to the axioms. Coq, Isabelle, *Lean*.
The trust argument is the one from the last slide, and it *shrinks* the trusted base rather than removing it: you still trust the *statement, the definitions and the axioms* — but the proof itself is checked by a kernel of a few thousand lines.

The price: you must build the library first

Every definition you use has to exist already. *Mathlib*, Lean's library, is that effort — and it is why formalising a paper is now sometimes a matter of weeks rather than years.

Two landmarks. The Feit–Thompson odd order theorem (Coq, 2012): 255 pages of mathematics, six years of work. And Scholze's *Liquid Tensor Experiment* (Lean, 2022), begun because *he wanted a proof of his own checked*.

The case for it: finite simple groups

The classification of the finite simple groups: announced complete in 1983, with a gap closed only in *2004*.

The case for it: finite simple groups

The classification of the finite simple groups: announced complete in 1983, with a gap closed only in *2004*.
Some ten thousand journal pages, hundreds of papers, a hundred authors, half a century.

The case for it: finite simple groups

The classification of the finite simple groups: announced complete in 1983, with a gap closed only in *2004*.
Some ten thousand journal pages, hundreds of papers, a hundred authors, half a century.

The uncomfortable part

Nobody alive has read all of it. Its correctness is a property of a community rather than of any person — and it is used as a black box across much of mathematics. The four colour objection inside out: there, no human could check the *computation*; here, the *mathematics*.

The case for it: finite simple groups

The classification of the finite simple groups: announced complete in 1983, with a gap closed only in *2004*.
Some ten thousand journal pages, hundreds of papers, a hundred authors, half a century.

The uncomfortable part

Nobody alive has read all of it. Its correctness is a property of a community rather than of any person — and it is used as a black box across much of mathematics. The four colour objection inside out: there, no human could check the *computation*; here, the *mathematics*.

For more on Lean, see the courses by Mitch and María Inés.

Summary of the first two roles

	Explorer	**Verifier**
What it gives you	a guess	a certificate
Rigour	none, and that is fine	total
Who does the math	you	you
Cost of an error	you waste time	the theorem is wrong
Was it contested?	mildly	*violently*
Is it now normal?	completely	mostly

Summary of the first two roles

	Explorer	**Verifier**
What it gives you	a guess	a certificate
Rigour	none, and that is fine	total
Who does the math	you	you
Cost of an error	you waste time	the theorem is wrong
Was it contested?	mildly	*violently*
Is it now normal?	completely	mostly

Keep this table in mind

In lecture 3 we will fill in a third column. Many of the objections you will want to raise against the new tools have already been raised against one of these two.

"Artificial intelligence" is an old phrase

1955. McCarthy, Minsky, Shannon and Rochester need a name for a funding proposal and invent *"artificial intelligence"*; the Dartmouth workshop follows in 1956. They estimate that ten people could make serious progress on language and abstraction *over two summer months*.

"Artificial intelligence" is an old phrase

1955. McCarthy, Minsky, Shannon and Rochester need a name for a funding proposal and invent *"artificial intelligence"*; the Dartmouth workshop follows in 1956. They estimate that ten people could make serious progress on language and abstraction *over two summer months*.
It has been seventy years.

"Artificial intelligence" is an old phrase

1955. McCarthy, Minsky, Shannon and Rochester need a name for a funding proposal and invent *"artificial intelligence"*; the Dartmouth workshop follows in 1956. They estimate that ten people could make serious progress on language and abstraction *over two summer months*. *It has been seventy years.*

Learning was in the proposal from the start — the perceptron follows in 1958 — but what the phrase *delivered in mathematics* for forty years was search and symbolic manipulation: Newell and Simon's **Logic Theorist** (1956) on *Principia Mathematica*, and McCune's EQP settling the *Robbins conjecture* (1996), open since 1933.

"Artificial intelligence" is an old phrase

1955. McCarthy, Minsky, Shannon and Rochester need a name for a funding proposal and invent *"artificial intelligence"*; the Dartmouth workshop follows in 1956. They estimate that ten people could make serious progress on language and abstraction *over two summer months*. *It has been seventy years.*

Learning was in the proposal from the start — the perceptron follows in 1958 — but what the phrase *delivered in mathematics* for forty years was search and symbolic manipulation: Newell and Simon's **Logic Theorist** (1956) on *Principia Mathematica*, and McCune's EQP settling the *Robbins conjecture* (1996), open since 1933.

Where the boundary is

EQP is a cousin of a SAT solver — the verifier of §3. *The break comes when learned representations take over*, on the next slide.

What changed: learning instead of programming

The old paradigm

A human writes down the rules. The machine applies them. Chess: an evaluation function, hand-designed, plus search (Deep Blue, 1997).

What changed: learning instead of programming

The old paradigm

A human writes down the rules. The machine applies them. Chess: an evaluation function, hand-designed, plus search (Deep Blue, 1997).

The new paradigm

A human writes down an *architecture* and an *objective*. The machine finds the rules by *fitting parameters to data*.

What changed: learning instead of programming

The old paradigm

A human writes down the rules. The machine applies them. Chess: an evaluation function, hand-designed, plus search (Deep Blue, 1997).

The new paradigm

A human writes down an *architecture* and an *objective*. The machine finds the rules by *fitting parameters to data*.

The turning point is usually dated to **2012** (image recognition). Design choices mattered, but so did *more data and more compute*.

What changed: learning instead of programming

The old paradigm

A human writes down the rules. The machine applies them. Chess: an evaluation function, hand-designed, plus search (Deep Blue, 1997).

The new paradigm

A human writes down an *architecture* and an *objective*. The machine finds the rules by *fitting parameters to data*.

The turning point is usually dated to **2012** (image recognition). Design choices mattered, but so did *more data and more compute*.

Very little of the underlying mathematics was new.

The *Alpha*–series, in order

Every one of them from *Google DeepMind*.

- **AlphaGo** (2016) beats Lee Sedol. In game 2, move 37 is a play no professional would make; commentators assume a bug. It wins the game.

  Then **AlphaGo Zero** (2017): same strength and better, trained *without a single human game*.

The *Alpha*–series, in order

Every one of them from *Google DeepMind*.

- **AlphaGo** (2016) beats Lee Sedol. In game 2, move 37 is a play no professional would make; commentators assume a bug. It wins the game.

  Then **AlphaGo Zero** (2017): same strength and better, trained *without a single human game*.

- **AlphaFold** (2020–21) predicts protein structure at experimental accuracy. Nobel Prize in Chemistry, 2024.

The *Alpha*–series, in order

Every one of them from *Google DeepMind*.

- **AlphaGo** (2016) beats Lee Sedol. In game 2, move 37 is a play no professional would make; commentators assume a bug. It wins the game.

  Then **AlphaGo Zero** (2017): same strength and better, trained *without a single human game*.

- **AlphaFold** (2020–21) predicts protein structure at experimental accuracy. Nobel Prize in Chemistry, 2024.

- **AlphaTensor** (2022), **FunSearch** (2023): the machine searches over *programs* and improves known constructions — including a new lower bound for the cap set problem, with Ellenberg as a co-author.

The *Alpha*–series, in order

Every one of them from *Google DeepMind*.

- **AlphaGo** (2016) beats Lee Sedol. In game 2, move 37 is a play no professional would make; commentators assume a bug. It wins the game.

  Then **AlphaGo Zero** (2017): same strength and better, trained *without a single human game*.

- **AlphaFold** (2020–21) predicts protein structure at experimental accuracy. Nobel Prize in Chemistry, 2024.

- **AlphaTensor** (2022), **FunSearch** (2023): the machine searches over *programs* and improves known constructions — including a new lower bound for the cap set problem, with Ellenberg as a co-author.

- **AlphaGeometry** (2024) solves 25 of 30 IMO geometry problems, roughly a gold medallist's level.

Closer to home: singularities in fluids

Does a smooth solution of 3D Euler *without boundary* stay smooth forever? A *singularity* would mean gradients blowing up at a finite time T. An approach is to look for a *self-similar* one,

$$u(x,t) = \frac{1}{(T-t)^{\alpha}}\, U\left(\frac{x - x_0}{(T-t)^{\beta}}\right),$$

for a *profile* U and exponents α, β.

Closer to home: singularities in fluids

Does a smooth solution of 3D Euler *without boundary* stay smooth forever? A *singularity* would mean gradients blowing up at a finite time T. An approach is to look for a *self-similar* one,

$$u(x,t) = \frac{1}{(T-t)^{\alpha}}\, U\left(\frac{x - x_0}{(T-t)^{\beta}}\right),$$

for a *profile* U and exponents α, β.
Euler scaling forces $\alpha + \beta = 1$; then substituting removes the time variable and U solves a *stationary* equation. *Finding a blow-up becomes finding a function.*

Closer to home: singularities in fluids

Does a smooth solution of 3D Euler *without boundary* stay smooth forever? A *singularity* would mean gradients blowing up at a finite time T. An approach is to look for a *self-similar* one,

$$u(x,t) = \frac{1}{(T-t)^{\alpha}}\, U\!\left(\frac{x-x_0}{(T-t)^{\beta}}\right),$$

for a *profile* U and exponents α, β.
Euler scaling forces $\alpha + \beta = 1$; then substituting removes the time variable and U solves a *stationary* equation. *Finding a blow-up becomes finding a function.*

Stable versus unstable

A *stable* profile attracts nearby data, so running the PDE forward finds it. An *unstable* one has growing directions — forward evolution will not find it; you solve the profile equation instead.

Machine learning finds the candidate (2023)

Wang, Lai, Gómez-Serrano and Buckmaster (2023): represent U *by a neural network* U_θ and minimise the PDE residual. Self-similar candidates for Boussinesq and 3D Euler — and, for Córdoba–Córdoba–Fontelos, *the first unstable one found this way*.

Machine learning finds the candidate (2023)

Wang, Lai, Gómez-Serrano and Buckmaster (2023): represent U *by a neural network* U_θ and minimise the PDE residual. Self-similar candidates for Boussinesq and 3D Euler — and, for Córdoba–Córdoba–Fontelos, *the first unstable one found this way*.

Which kind of machine learning this is

Not a language model. A small network trained from scratch for *one* equation, whose output is *a function* and not a sentence — *domain-specific* machine learning.

Machine learning finds the candidate (2023)

Wang, Lai, Gómez-Serrano and Buckmaster (2023): represent U *by a neural network* U_θ and minimise the PDE residual. Self-similar candidates for Boussinesq and 3D Euler — and, for Córdoba–Córdoba–Fontelos, *the first unstable one found this way*.

Which kind of machine learning this is

Not a language model. A small network trained from scratch for *one* equation, whose output is *a function* and not a sentence — *domain-specific* machine learning.

The template of the whole course

The network produces a candidate; a computer-assisted proof (§3) must then certify it — a step *still to come*. *Machine proposes, mathematics disposes.*

The sequel (2025)

The same four authors, now with the *Google DeepMind* team — *Discovery of Unstable Singularities* (22 authors, 2025).

The sequel (2025)

The same four authors, now with the *Google DeepMind* team — *Discovery of Unstable Singularities* (22 authors, 2025).

What they found

New *families* of unstable self-similar blow-up solutions, for the incompressible porous media equation and for 3D Euler with boundary, plus an empirical formula relating the blow-up rate to the order of instability.

The sequel (2025)

The same four authors, now with the *Google DeepMind* team — *Discovery of Unstable Singularities* (22 authors, 2025).

What they found

New *families* of unstable self-similar blow-up solutions, for the incompressible porous media equation and for 3D Euler with boundary, plus an empirical formula relating the blow-up rate to the order of instability.

Architectures chosen for the problem, plus a high-precision Gauss–Newton optimiser. *For some of the profiles* the residual reaches near double-float machine precision, limited only by GPU round-off.

The sequel (2025)

The same four authors, now with the *Google DeepMind* team — *Discovery of Unstable Singularities* (22 authors, 2025).

What they found

New *families* of unstable self-similar blow-up solutions, for the incompressible porous media equation and for 3D Euler with boundary, plus an empirical formula relating the blow-up rate to the order of instability.

Architectures chosen for the problem, plus a high-precision Gauss–Newton optimiser. *For some of the profiles* the residual reaches near double-float machine precision, limited only by GPU round-off.

Why the precision is the point

That is what a *computer-assisted proof* needs as input (§3). The network proves nothing; it delivers an object sharp enough to be proved about.

The sequel's sequel (September 2026)

As I write, 9 September 2026 — and still moving.

Alpöge and Buckmaster, continuing a program of Córdoba and Martínez-Zoroa: finite-time blow-up *with a smooth forcing term* for incompressible porous media, Boussinesq, and 3D Euler — each one *formalised in Lean*.

The sequel's sequel (September 2026)

As I write, 9 September 2026 — and still moving.

Alpöge and Buckmaster, continuing a program of Córdoba and Martínez-Zoroa: finite-time blow-up *with a smooth forcing term* for incompressible porous media, Boussinesq, and 3D Euler — each one *formalised in Lean*.

Almost immediately after, OpenAI announced blow-up for *Navier–Stokes* itself, with a paper and a Lean formalisation, both public.

The sequel's sequel (September 2026)

As I write, 9 September 2026 — and still moving.

Alpöge and Buckmaster, continuing a program of Córdoba and Martínez-Zoroa: finite-time blow-up *with a smooth forcing term* for incompressible porous media, Boussinesq, and 3D Euler — each one *formalised in Lean*.
Almost immediately after, OpenAI announced blow-up for *Navier–Stokes* itself, with a paper and a Lean formalisation, both public.

Read the problem statement, not the headline

Fefferman asks for *one of four* statements. (A), (B): smoothness forever, force *zero*. *(C), (D): break-down, and these permit a smooth force.* The claim is (C) and (D) — on Clay's list, and not the question most analysts have in mind.

The sequel's sequel (September 2026)

As I write, 9 September 2026 — and still moving.

Alpöge and Buckmaster, continuing a program of Córdoba and Martínez-Zoroa: finite-time blow-up *with a smooth forcing term* for incompressible porous media, Boussinesq, and 3D Euler — each one *formalised in Lean*.
Almost immediately after, OpenAI announced blow-up for *Navier–Stokes* itself, with a paper and a Lean formalisation, both public.

Read the problem statement, not the headline

Fefferman asks for *one of four* statements. (A), (B): smoothness forever, force *zero*. *(C), (D): break-down, and these permit a smooth force.* The claim is (C) and (D) — on Clay's list, and not the question most analysts have in mind.

In 2025 the machine delivered an object *sharp enough to be proved about*; now the claim is *the proof itself*.

Whose result is it?

The route was opened by *Diego Córdoba* (ICMAT) and *Luis Martínez-Zoroa* (CUNEF), in a direction almost nobody else was taking. By 2023 they had Euler blow-up with a *rough* force. *What remained was to make the force smooth* — and smooth is what Clay requires.

Whose result is it?

The route was opened by *Diego Córdoba* (ICMAT) and *Luis Martínez-Zoroa* (CUNEF), in a direction almost nobody else was taking. By 2023 they had Euler blow-up with a *rough* force. *What remained was to make the force smooth* — and smooth is what Clay requires.

What the machines did, precisely

They closed that one gap. *Both* teams — Alpöge–Buckmaster and OpenAI — worked inside the Córdoba–Martínez-Zoroa program.

Whose result is it?

The route was opened by *Diego Córdoba* (ICMAT) and *Luis Martínez-Zoroa* (CUNEF), in a direction almost nobody else was taking. By 2023 they had Euler blow-up with a *rough* force. *What remained was to make the force smooth* — and smooth is what Clay requires.

What the machines did, precisely

They closed that one gap. *Both* teams — Alpöge–Buckmaster and OpenAI — worked inside the Córdoba–Martínez-Zoroa program.

Fefferman, who wrote the problem, calls them *"the heroes of the story"*; Buckmaster writes, *"I believe Luis Martínez-Zoroa deserves a Fields Medal."*

¡Muchas gracias!

Questions — and objections — are very welcome.

Kraftwerk, Zürich, 1976 — the year of Appel and Haken.
Photo Ueli Frey, CC BY-SA 3.0.

References

The machine as explorer

Turing A. M.
Some calculations of the Riemann zeta-function. Proc. London Math. Soc. (3) **3** (1953), 99–117.

Fermi E., Pasta J., Ulam S. (with Tsingou M.)
Studies of nonlinear problems. Los Alamos report LA-1940 (1955).

Zabusky N. J., Kruskal M. D.
Interaction of solitons in a collisionless plasma. Phys. Rev. Lett. **15** (1965), 240–243.

References

The machine as explorer (continued)


Lander L. J., Parkin T. R.
Counterexample to Euler's conjecture on sums of like powers. Bull. Amer. Math. Soc. **72** (1966), 1079.


Bailey D., Borwein P., Plouffe S.
On the rapid computation of various polylogarithmic constants. Math. Comp. **66** (1997), 903–913.

References

The machine as verifier


Appel K., Haken W.
Every planar map is four colorable. Illinois J. Math. **21** (1977), 429–567.


Gonthier G.
Formal proof — the four-color theorem. Notices Amer. Math. Soc. **55** (2008), 1382–1393.


Tucker W.
A rigorous ODE solver and Smale's 14th problem. Found. Comput. Math. **2** (2002), 53–117.

References

The machine as verifier (continued)


Hales T. et al.
A formal proof of the Kepler conjecture. Forum Math. Pi **5** (2017), e2.


Heule M. J. H., Kullmann O., Marek V. W.
Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. SAT 2016, LNCS **9710**, 228–245.


Wang Y., Lai C.-Y., Gómez-Serrano J., Buckmaster T.
Asymptotic self-similar blow-up profile for three-dimensional axisymmetric Euler equations using neural networks. Phys. Rev. Lett. **130** (2023), 244002.


Wang Y. et al. (22 authors, incl. Buckmaster T., Gómez-Serrano J., Lai C.-Y.)
Discovery of Unstable Singularities. arXiv:2509.14185 (2025).

References

September 2026: the preprints


Alpöge L., Buckmaster T.
Blowup for the Boussinesq equations with smooth forcing. Preprint, September 2026.
https://cims.nyu.edu/~tristanb/boussinesq.pdf


Alpöge L., Buckmaster T.
Blowup for the Euler equations with smooth forcing. Preprint, September 2026.
https://cims.nyu.edu/~tristanb/euler.pdf


Alpöge L., Buckmaster T., Coiculescu M. P.
Extending the Córdoba–Martínez-Zoroa IPM blow-up to uniformly space-time smooth forcing. Preprint, September 2026.
https://cims.nyu.edu/~tristanb/ipm.pdf

References

September 2026: the claim, and how to check it


Fefferman C. L.
Existence and smoothness of the Navier–Stokes equation. The official Clay statement, (A)–(D).
https://www.claymath.org/wp-content/uploads/2022/06/navierstokes.pdf


OpenAI.
On the Navier–Stokes Millennium Prize Problem. Sept. 2026.
https://openai.com/index/navier-stokes-solution/


OpenAI. *Lean certificates. Go and check it.*
https://github.com/openai/NavierStokesAndEuler

References

September 2026: what to read about it

Kakaes K.
AI has solved one of math's $1 million Millennium Prize problems. *Quanta Magazine*, 8 September 2026 — with Fefferman, Córdoba and Martínez-Zoroa on the record.
https://www.quantamagazine.org/ai-has-solved-one-of-maths-1-million-millennium-prize-problems-20260908/

Tao T.
Finite time blowup with smooth forcing term for the incompressible porous medium, Boussinesq, and incompressible Euler equations. Blog post, 7 September 2026.
https://terrytao.wordpress.com/2026/09/07/

References

The machine as collaborator


Cybenko G.
Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems **2** (1989), 303–314.


Vaswani A. et al.
Attention is all you need. NeurIPS 2017.


Silver D. et al.
Mastering the game of Go with deep neural networks and tree search. Nature **529** (2016), 484–489.

References

The machine as collaborator (continued)


Romera-Paredes B. et al.
Mathematical discoveries from program search with large language models. Nature **625** (2024), 468–475.


Trinh T. H., Wu Y., Le Q. V., He H., Luong T.
Solving olympiad geometry without human demonstrations. Nature **625** (2024), 476–482.


Bloom T.
Erdős problems. `https://www.erdosproblems.com`

References

Watchable, and one benchmark


AlphaGo: The Movie. Google DeepMind (2017).
https://www.youtube.com/watch?v=WXuK6gekU1Y


The Thinking Game. Google DeepMind (2024) — DeepMind and AlphaFold.
https://www.youtube.com/watch?v=d95J8yzvjbQ


deedy.
IMO 2026: autonomous AI model comparison, with per-problem audit trails and independent grading.
https://github.com/deedy/imo-2026

Image credits

Everything below is public domain or used under the licence stated.

- **Harvard College Observatory computers.** Harvard University Archives. Public domain, via Wikimedia Commons, *Astronomer Edward Charles Pickering's Harvard computers*.
- **Kepler's stacking figure.** *Strena seu de nive sexangula* (1611). Public domain, via Wikimedia Commons, *Kepler conjecture 2*.
- **Lander and Parkin (1966).** Bull. Amer. Math. Soc. **72**, 1079. Open access: `https://projecteuclid.org/journals/bulletin-of-the-american-mathematical-society-new-series/volume-72/issue-6/Counterexample-to-Eulers-conjecture-on-sums-of-like-powers/bams/1183528522.full`
- **Alpöge's post.** `https://x.com/__alpoge__/status/2079028340955197566`
- **Kraftwerk, Zürich, 1976.** Photo Ueli Frey, CC BY-SA 3.0, via Wikimedia Commons: `https://commons.wikimedia.org/wiki/File:Kraftwerk_by_Ueli_Frey_(1976).jpg`
- **FPUT and FrontierMath plots.** Generated by the scripts in this repository; FrontierMath data from Epoch AI, *AI Benchmarking Hub* (CC BY 4.0).