

Highlights

cert-machine: independent exact certification of machine-generated mathematics — the method and three benchmark exercises*

Carlos Toledo

- A machine's claim is read over a declared box and certified, refuted with its witness, or refused with the reason.
- 19 kissing-number bounds of an autonomous discovery system hold: every pair of 9.75×10^{15} decided exactly.
- OpenAI's Navier–Stokes Lean certificate: partial at the announced commit, certified at the next.
- Tolerance graders accept 89.5% of 4,000 provably wrong answers; a certificate grader accepts none.
- Every number is a macro from a public record; the build refuses when a record moves.

cert-machine: independent exact certification of machine-generated mathematics — the method and three benchmark exercises

Carlos Toledo^{a,*}

^a*Independent researcher, the cert-machine project,*

Abstract

cert-machine decides the numbers and claims that machines now propose — constants, counterexamples, packings, fitted distributions, operating points, formal certificates. Its one description is *Independent exact certification of machine-generated mathematics — exact arithmetic, no code shared with the claimant, refusal as a verdict*. A claim is read as a predicate over a declared box of inputs and receives one of three verdicts: CERTIFIED when finitely many exact or outward-rounded computations show the predicate at every point of the box, REFUTED when they show its negation, with the falsifying object, and REFUSED otherwise, with the reason and, where the declared evidence is split, one proved witness on each side and the threshold that would decide. The verdicts are sound under the inclusion property of interval extensions; the paper describes the arithmetic that secures it, the certifiers that reduce analytic questions to finitely many such decisions, and the verification discipline, enforced by code that refuses. Three benchmark exercises test the method on claims machines produced. The 19 kissing-number lower bounds of an autonomous discovery system: every pair of 9.75×10^{15} decided exactly, every bound found to hold, the claimant publishing its points at the system’s request. OpenAI’s Lean certificate for a Navier–Stokes blowup: read hypothesis by hypothesis, built, found partial

*Draft v0.2 of 2026-10-14, restructured on the reviewing group’s suggestions (materials and methods, three benchmark exercises, discussion, recommendations); v0.1.1 of 2026-10-09 is the previous shape. Every number in this manuscript is read at build time from the records named in the Data availability statement (repository commit f7662d3b8392); the prose is authored and should be reviewed like any other. The author list and the CRediT roles are to be agreed with the reviewing group. Machine-derived and not peer-reviewed.

*Corresponding author.

Email address: carlos@carlostoledo.co (Carlos Toledo)

at the announced commit and certified at the next. The graders of mathematical benchmarks: measured against 4,000 provably wrong answers, of which an absolute-tolerance grader accepts 89.5% and a certificate grader none, beside a table of the gaps between what OpenAI’s 719-manuscript release claims and what it checks. The contribution claimed is the system and what it decided; interval arithmetic is standard and is cited.

Keywords: machine-generated mathematics, interval arithmetic, exact arithmetic, certificates, verification, kissing numbers, formal proof, benchmark graders

1. Introduction

Machines now propose mathematics and engineering numbers at a rate no referee can read. Conjectures about fundamental constants are generated from numerical coincidences and supported by a probability argument (Raayoni et al., 2021); constructions in combinatorics and analysis are found by program search (Romera-Paredes et al., 2024; Novikov et al., 2025); autonomous research systems publish record-breaking configurations with their own verification files (Yang et al., 2026); long proofs arrive with machine formalisations (OpenAI, 2026b,a); and in engineering, surrogate models, optimisers and language models propose operating points and design values (Toledo, 2026d). Each arrives with its producer’s checker. A checker written by the producer shares the producer’s blind spots, and the published record shows the cost: a floating-point product printed as an exact constant, witnesses that hold only within a verifier’s tolerance, a checker that accepts what its own definition excludes, a pairwise condition proved by structure and never checked pair by pair, and a clause present in a paper’s theorem and absent from the formal statement that was proved (Toledo, 2026f).

cert-machine decides such claims independently. It reads the claim from its published statement and bytes, never runs the claimant’s program in the trust path, decides it in exact rational arithmetic or in outward-rounded interval arithmetic, and publishes one of three verdicts with the object that justifies it. Where it cannot decide, it says so, and the refusal is recorded like any other result. The design choice that runs through all of it is that the system is sound and deliberately incomplete. A procedure that must always answer cannot be both, for the questions machines propose include undecidable ones (Bastounis et al., 2026); a procedure that may refuse can. The cost of soundness is paid in refusals, and the system’s answer to that cost is to make every refusal informative: it names its reason, and, where

the declared evidence is split, the witnesses and the threshold that would decide it.

Where it stands.. Validated numerics and computer-assisted proof are the certifiers’ ancestry: interval arithmetic (Moore et al., 2009; Neumaier, 1990), verification methods in floating point (Rump, 2010, 1999), ball arithmetic (Johansson, 2017) and the proofs they made possible (Tucker, 2002; Hales et al., 2017). A proof assistant checks a proof against a stated theorem (de Moura and Ullrich, 2021; The mathlib Community, 2020); what it cannot check is whether the statement is the one intended, and a translation that compiles may not be faithful (Bastounis et al., 2026). Systems that propose mathematics (Raayoni et al., 2021; Romera-Paredes et al., 2024; Novikov et al., 2025; Yang et al., 2026) each come with an evaluator written by the producer, and the graders of mathematical benchmarks compare a number with a stored decimal within a tolerance (Hendrycks et al., 2021; Cobbe et al., 2021; Gao et al., 2023). What this system adds is not a technique but a setting: the computations are aimed at other parties’ claims, the outcome is one of three published words with a witness, and a failure to decide is a result. Section 4.3 returns to each neighbour.

This paper.. Section 2 gives the materials — the claims, their bytes, the machines — and the methods: the decision grammar and its soundness, the arithmetic that secures the hypothesis of that proof, the certifiers that reduce the questions machines ask to finitely many such decisions, and the verification discipline, enforced by code that refuses. Section 3 is a benchmark exercise in three parts, one per kind of machine-generated claim: the kissing-number bounds of an autonomous discovery system (mathematics), OpenAI’s Lean certificate for a Navier–Stokes blowup (formal proof), and the graders of mathematical benchmarks together with a table of what OpenAI’s 719-manuscript release claims beyond what it checks (computer science). Section 4 discusses what the three show, counts the independence the system can claim, places it beside existing practice and states its limits; Section 5 concludes and recommends. The contribution claimed is the system — the grammar, the certifiers that serve it, the discipline that guards it, the public record — and what it decided; interval arithmetic, the Krawczyk operator and exact geometric predicates are decades old (Moore et al., 2009; Krawczyk, 1969; Rump, 2010; Shewchuk, 1997), and nothing here claims them.

2. Materials and methods

2.1. Materials

The claims.. The system’s inputs are claims and the bytes they rest on. The public register holds 132 published claims from 25 sources — constants, constructions, bounds, formal statements, printed tables — each pinned by the digest of the file that states it (Toledo, 2026f). The three exercises of Section 3 draw on three bodies of material. The kissing configurations of the Qiushi Engine (Yang et al., 2026): 19 point sets in dimensions 25, 27, 32–39, 43, 45 and 49–55, up to 53,301,140 points, published as integer coordinates in five releases of the claimant’s repository (Oxelra-AI, 2026), every file pinned by the digest the claimant’s own manifest and the hosting platform report. OpenAI’s Navier–Stokes repository (OpenAI, 2026b) at its announced commit 8937a8f4 and its next, f9e8bc5b: 24 files pinned on 2026-09-09 (the two papers, the Clay statement, the Lean bridge files, the challenge statement, the announcement), and a Lean library of 2,486 files and 618,762 lines. OpenAI’s mathematics release (OpenAI, 2026a) at commit fd4aeeb2: 719 manuscripts in 372 families and 416 Comparator challenges, censused file by file. And the grader corpus: 104 facts of the register, each pinned to the certificate that decided it, from which the submissions of Section 3.3 are minted.

The machines.. Everything that finishes within about a day runs on an 8-core laptop with 16 GB of memory: the interval kernel and the certifiers in Node v24.14.1 (V8 13.6), the bulk pair checks in Python with numpy’s BLAS, the Lean builds. The 416 challenges of the release ran on GitHub’s hosted runners (4 vCPU, 16 GB, six hours); the challenges those could not finish were sent to rented CPU machines under a pre-registered budget cap, and none of them finished there either, which the record says. The seven largest kissing configurations, about 1.4×10^{15} pairs each, ran on one rented NVIDIA L40S at \$1.09 per hour for 6.8 GPU-hours (\$8.15 billed), by a kernel that was first required to reproduce the laptop’s answers on two smaller dimensions. Every machine that decided a record is named in it with its platform and software.

The records.. Every decision writes a record: the inputs by path and SHA-256, the verdict, the witness, the scope, the machine, the time, and the second implementation’s reading where one exists. The records are the materials of Section 3 and of every number in this paper: each number is a

macro written by a generator from the record that holds it, and the generator refuses to write when a record no longer supports the sentence that quotes it (Section 2.5).

2.2. The decision grammar

2.2.1. Claims over boxes

A *box* is a product $X = [\underline{x}_1, \bar{x}_1] \times \cdots \times [\underline{x}_n, \bar{x}_n]$ of closed intervals with rational endpoints; a *double* is a rational, so a box of doubles is a box. A *declared envelope* B is a box each of whose sides carries the source of its bounds — a survey, a data sheet, a chart. A number someone printed is the box its digits allow: a printed 0.0634 is $[0.06335, 0.06345]$, and a fit printed to four decimals is a box in parameter space.

Definition 1 (interval extension). Let $g : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$. A map G from boxes in D to intervals is an *interval extension* of g if $g(X) \subseteq G(X)$ for every box $X \subseteq D$ (the inclusion property).

The inclusion property is the one hypothesis of every verdict below, and Section 2.3 is about securing it. The extensions used are natural extensions — the formula evaluated in interval arithmetic — and the classical theorem that a composition of operations with the inclusion property has it (Moore et al., 2009; Neumaier, 1990) carries it from the primitives to the model.

Definition 2 (claim). A *claim* is a pair (B, φ) of a box and a predicate $\varphi(x) \equiv \bigwedge_j g_j(x) \leq c_j$ whose functions have interval extensions G_j (an equality, or an exact predicate such as the orientation of three points, is the special case decided in exact arithmetic). It is read in one of two ways:

universal the claim asserts $\varphi(x)$ for every $x \in B$ — the reading of a mathematical statement (“for all $n \leq N$ ”, “on $[0, 1]$ ”, “for every member of this family”, “for every pair of these points”);

epistemic the claim asserts $\varphi(x^\dagger)$ for an unknown true input $x^\dagger \in B$ — the reading of an engineering number, where B is the declared uncertainty about the input.

The two readings differ in what a single proved counterexample does. Under the universal reading it refutes the claim; under the epistemic reading it shows only that the declared evidence does not settle the question.

2.2.2. Three verdicts

Definition 3 (verdicts). Let $\mathcal{X}$ be a finite cover of B by boxes. The claim is

certified if $\sup G_j(X) \leq c_j$ for every j and every $X \in \mathcal{X}$;

refuted under the universal reading, if some box $X^* \subseteq B$ (typically a thin box, a point) has $\inf G_j(X^*) > c_j$ for some j ; under the epistemic reading, if every $X \in \mathcal{X}$ has $\inf G_j(X) > c_j$ for some j , or the claim contradicts a bound that holds on all of B ;

refused otherwise. Under the epistemic reading a refusal is *split* when two points $x^-, x^+ \in B$ have been proved to satisfy and to violate φ respectively, and *unresolved* when the enclosures did not separate within the budget.

Proposition 1 (soundness). *If every G_j is an interval extension of g_j , then CERTIFIED implies $\varphi(x)$ for every $x \in B$; REFUTED implies, under the universal reading, $\neg\varphi(x)$ for some $x \in B$ and, under the epistemic reading, $\neg\varphi(x)$ for every $x \in B$; and a split REFUSED implies that neither $\forall x \in B \varphi(x)$ nor $\forall x \in B \neg\varphi(x)$ holds, so that $\varphi(x^\dagger)$ is not determined by the declared envelope.*

Proof. Let $x \in B$ and choose $X \in \mathcal{X}$ containing it. By the inclusion property $g_j(x) \in g_j(X) \subseteq G_j(X)$, hence $g_j(x) \leq \sup G_j(X) \leq c_j$ for every j , which is $\varphi(x)$. The refutations are the same argument with the inequality reversed, on X^* or on every member of the cover. For a split refusal, x^- and x^+ are points of B at which φ is proved true and false respectively; either could be $x^\dagger$. $\square$

An unresolved refusal claims nothing, and the grammar keeps it that way: a refusal is terminal, never retried at lower rigour, never converted into a probability and never dropped from the record (Toledo, 2026f). The only path from REFUSED to CERTIFIED is a new exact argument — a finer cover, a better extension, a different certifier, or, as Section 3.1 shows, the bytes the claimant had not yet published. The three words are not probabilities and there is no fourth.

2.2.3. Witnesses

Every verdict carries the object that justifies it, and the object is what is published. A certification carries its *certificate*: the inputs pinned by digest, the cover, the enclosures, and a statement of what would make it fail. The

certificate constructor of the interval toolkit refuses to build a certificate whose falsifier is not stated and a CERTIFIED that carries no evidence, on the principle that a certificate that cannot go red is decoration. A refutation carries the falsifying object, printed — a counterexample, an observation outside a region, a violated equation, a digit that is not the rounding of an enclosure. A split refusal carries its two witnesses. A certificate re-runs without the engine: its facts are exact or outward-rounded, and any arithmetic that rounds outward can check them.

2.2.4. Thresholds

A split refusal is the most useful outcome the grammar has, because the question it leaves open has a size.

Definition 4 (flip thresholds). Let the envelope depend on a declared scalar p — an operating choice, or one end of an interval that a measurement could narrow — through boxes $B(p)$, and let the verdict be monotone in p : if it is CERTIFIED at p it is CERTIFIED at every $p' \leq p$, and if it is REFUTED at p it is REFUTED at every $p' \geq p$. The *flip thresholds* are $p_C = \sup\{p : \text{CERTIFIED over } B(p)\}$ and $p_R = \inf\{p : \text{REFUTED over } B(p)\}$.

Proposition 2. *Under monotonicity, if the procedure has certified the claim over $B(\underline{p})$ and refuted it over $B(\bar{p})$, then the claim is CERTIFIED for every $p \leq \underline{p}$ and REFUTED for every $p \geq \bar{p}$, and $\underline{p} \leq p_C \leq p_R \leq \bar{p}$.*

The proof is the definition. The procedure computes $\underline{p}$ and $\bar{p}$ by bisection over rational p , each endpoint decided like any other claim, or in closed form where p enters additively, the result rounded outward to the precision reported; the engineering demonstrator does both, the pump discharge in closed form because the wellhead pressure is the discharge plus a model term, and two inspection thresholds by bisection (Toledo, 2026d). The threshold turns a refusal into a measurement or an operating change with a known effect on the verdict: at this discharge the rule is proved, at that roughness it is refuted.

2.2.5. Compositions

A receipt with several checks is REFUTED if any check is refuted, otherwise REFUSED if any is refused, otherwise CERTIFIED; by Proposition 1 this is sound for their conjunction. The public register composes the three verdicts into words that each say what they are made of (Table 1); none is a fourth verdict. Two consequences are enforced rather than recommended (Toledo, 2026f). A scope sentence is load-bearing, because a CERTIFIED without the

word	composed of	the witness the row carries
CERTIFIED	—	a certificate with its falsifier; re-runs without the engine
REFUTED	—	the falsifying object, printed
REFUSED	—	the reason; for a split refusal, two witnesses and the thresholds
PARTIAL	certified + refused	which fragment certified and which core was not reached
NEEDS DATA	refused, cause with the claimant	the bytes that would decide the row, named
MIXED	certified + refuted, one record	the counts of both, and the items refuted
REPAIRED	refuted + certified	the object as printed refuted; the object beside it certified
QUEUED	not a verdict	nothing; the row is not counted as decided

Table 1: The verdicts and their compositions in the public register (Toledo, 2026f). Engineering surfaces write PROVED for CERTIFIED; a configuration whose published bytes are already the exact object is WITNESSED; a ranking of families says DECIDED; the words for a printed number — REPRODUCED, CONSISTENT, OFF THE MAXIMUM, OUTSIDE ITS SUPPORT — refine the three for a comparison of two boxes (Toledo, 2026c).

scope beside it can say more than the record, as when the computational fragment of a claim is certified and its analytic core is not touched. And the kinds of refusal are never summed: the instrument declining, the claimant having published nothing, a budget exhausted, a cell whose box did not close and an audit stating its reach are different behaviours with different denominators.

2.3. The arithmetic

Proposition 1 has one hypothesis, the inclusion property, and the arithmetic exists to secure it. Two kinds are used, and the choice is made per quantity: exact rationals where a tie must be decided or the object is rational, outward-rounded intervals where the quantity is transcendental or the data are long.

Exact rationals.. Numbers are reduced fractions of arbitrary-precision integers. Doubles convert to them losslessly; linear systems are solved by exact elimination, which reports a singular matrix rather than a small pivot; conversion back to a double is for display only. Rational arithmetic decides every sign it is asked, and it is used for the corners of multilinear forms

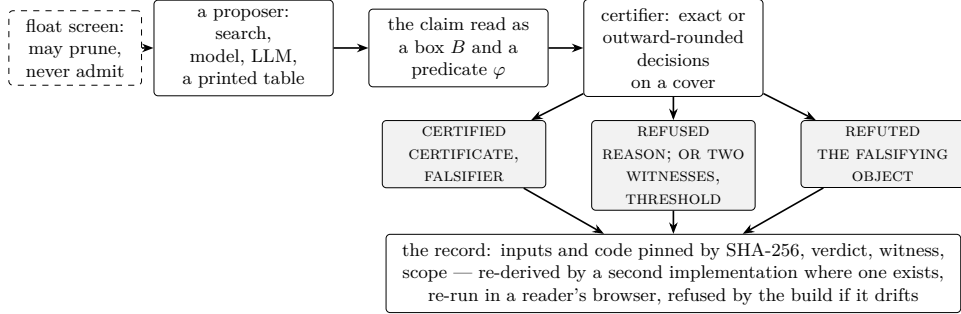


Figure 1: The system. A proposer is never opened and never trusted; its output is read as a claim over a box and decided by a certifier; the verdict, its witness and the pinned inputs form a record that re-runs without the engine.

(Section 2.4.4), the expected counts of a benchmark, polynomial identities, and the ends of enclosures whose comparison must be exact. Decimal literals — the digits a claimant printed — are read as the rationals they denote and kept as strings beside their double images, so that a vertex printed to three decimals and an observation printed to four can be found to coincide (Toledo, 2026c).

Exact integers in floating-point hardware.. A product of integer vectors is itself an integer, and floating-point hardware computes it exactly when no partial sum can leave the range the format holds exactly: 2^{24} for single precision, 2^{53} for double, 2^{11} for the half precision of tensor cores. The bound used is the Cauchy–Schwarz one — every partial sum of products of two vectors is at most the product of their lengths — checked against the data before anything runs and written into the record. This is what lets 10^{15} pairs of integer points be decided exactly on a GPU (Section 3.1): the arithmetic is floating point, the result is an integer held exactly, and the test applied to it is the exact integer test.

Outward rounding on doubles.. IEEE-754 arithmetic in round-to-nearest returns, for each of $+$, $-$, $\times$, $\div$, the double nearest the exact result, which therefore lies within half a unit in the last place of the computed one (IEEE, 2019; Goldberg, 1991). JavaScript cannot set the rounding mode, so directed rounding is emulated: each computed endpoint is moved one unit in the last place outward by stepping its bit pattern. The trust base of this emulation is stated once: the correctly rounded $+$, $-$, $\times$, $\div$, $\sqrt{}$ that IEEE-754 mandates and V8 implements, V8’s arbitrary-precision integers, and nothing in `Math.*`, which ECMA-262 leaves “implementation-approximated” (a 2017

proposal to require one-ulp accuracy was closed without adoption); the Node version that produced every record is written into the record (v24.14.1, V8 13.6, whose `Math.*` is the `fdlibm` port, faithful to within one ulp and not correctly rounded; V8 moved these functions to a correctly rounded library in May 2026, which is why the version is material; correctly rounded libraries are a recent engineering achievement (Sibidanov et al., 2022)). The operations are those of the basic set of the interval standard (IEEE, 2018); the implementation is not a conforming one and carries no decorations.

Proposition 3. *Let $\circ \in \{+, -, \times, \div\}$, let A, B be intervals of doubles — pairs of non-NaN doubles with lower end at most upper end, an infinite endpoint allowed as a bound — with $0 \notin B$ when $\circ = \div$, and let ℓ and h be the round-to-nearest values of the smallest and largest of the endpoint combinations $a \circ b$. If no endpoint combination is indeterminate ($0 \cdot \infty$, $\infty - \infty$), then $A \circ B \subseteq [\text{down}(\ell), \text{up}(h)]$, where down and up step one double toward $\mp\infty$ and fix $\mp\infty$; an exact result beyond the largest double rounds to $\pm\infty$, which the step fixes, so overflow is enclosed.*

Proof. For these operations the extremes of $a \circ b$ over a product of intervals are attained at endpoint combinations (monotonicity in each argument on each sign region, with $0 \notin B$ for division). The exact extreme lies within half the spacing of the doubles on its side of its round-to-nearest value, and the adjacent double on that side is a whole spacing away (subnormal spacing included), so it lies on the correct side of $\text{down}(\ell)$ and $\text{up}(h)$. $\square$

The domain of the proposition is enforced, not assumed: a malformed operand (a NaN, an inverted pair) or an indeterminate result is refused by exception and never decided, because a consumer written as “not proved above the bar, hence below it” would otherwise certify on a NaN (until 2026-10-09 nothing checked this, and the batteries skipped non-finite results; enforcing it exposed one battery check that had been passing on malformed boxes). Division by an interval that contains zero is refused, not extended; a square is clamped at zero; a non-integer power is refused, and so is an integer exponent of 2^{31} or more, where the implementation’s exponent arithmetic would wrap. The widening costs about one bit per operation and nothing at the accuracy the claims need; the domain checks cost a few comparisons per operation (ten million operations in 0.18 s). Where more precision is required — a constant to be certified to twelve digits, a Catalan or Li_2 value — a separate arbitrary-precision arithmetic carries a binary mantissa of arbitrary length with true directed rounding (floor and ceiling of the shifted integer), and converts its results to doubles outward with an exact check.

Elementary and special functions. The certifiers of this paper trust no library function, because the platform’s `Math.exp` and `Math.log` carry no accuracy guarantee in ECMAScript — with one exception that Section 4.4 names: an older helper that encloses the platform’s cosine by a derived pad rather than by a series is still called from six files, among them the radii-polynomial certifier of Section 2.4, and its retirement for the series below is scheduled with the re-run of those records. Each function is a truncated series evaluated in interval arithmetic plus an explicit bound on the omitted tail: `exp` after reduction $x = k \ln 2 + r$, $|r| < 1$, by twenty-four Taylor terms and the tail $|r|^{25}/25! (1 - |r|/26)^{-1}$; `log` after $x = m 2^e$ by the series of `atanh` in $v = (m-1)/(m+1) \in [0, 1/3)$, whose one-sided tail is valid because $v \geq 0$; `sin` and `cos` after reduction by an interval enclosure of $\pi/2$, with the box’s enclosure forced to ± 1 wherever it may contain an extremum; `ln 2` and π in exact rationals by `atanh(1/3)` and Machin’s formula, converted outward. Monotone functions on wide boxes are evaluated at the endpoints. The special functions follow the same pattern: Γ by Spouge’s formula with its error bound taken ten times over; $\ln \Gamma$, ψ and ψ' by Binet’s expansion after shifting the argument above twenty, the first omitted term as the bound; the regularized incomplete gamma function by its series with a geometric tail; Bessel functions by their power series with a certified ratio bound; `erfc` by Laplace’s continued fraction, whose consecutive convergents bracket the value (Abramowitz and Stegun, 1964); and inverses (Φ^{-1}, P^{-1}) by bisection that moves an end only on a certified sign. The tails are computed in round-to-nearest and are many orders below the one-ulp widening that absorbs them; that margin, not the tail’s own rounding, is what makes them safe, and it is stated where it is used.

Long sums. A running interval sum rounds outward at each of n additions, so its width grows linearly in n ; at the 175,320 hourly values of one buoy it is a thousand times what the terms carry. Each endpoint stream is instead summed by the compensated Sum2 algorithm (Dekker, 1971; Ogita et al., 2005) and the error bound of Ogita et al. (2005) is added outward with a factor of two to spare, which is what makes a likelihood over a hundred thousand data an enclosure narrow enough to rank families by (Toledo, 2026c).

Derivatives. Gradients and Hessians that a certifier needs come from second-order forward differentiation over the same interval arithmetic (a value, a gradient and a Hessian carried through every operation), or from univariate Taylor jets, so that the derivative a proof uses is an extension of the function

the proof is about. The remaining families carry hand-written scores and Hessians, evaluated in the same interval arithmetic.

2.4. The certifiers

Each certifier below turns a question that is analytic in form into finitely many decisions of the kind Proposition 1 covers. None is new; the assembly is, together with the rule that each refuses rather than guesses.

2.4.1. A zero, exactly one

Theorem 1 (Krawczyk (Krawczyk, 1969; Neumaier, 1990; Rump, 2010)). *Let $f : D \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$ be continuously differentiable, $X \subseteq D$ a box, $\hat{x} \in X$, Y a nonsingular matrix and $F'(X)$ an interval extension of the Jacobian over X . If*

$$K(X) = \hat{x} - Yf(\hat{x}) + (I - YF'(X))(X - \hat{x}) \subseteq \text{int } X,$$

then f has exactly one zero in X .

The certifier takes $\hat{x}$ from a floating-point search, Y as an approximate inverse of the Jacobian at $\hat{x}$, evaluates $K(X)$ with every datum enclosed by its neighbouring doubles, and inflates X for a bounded number of rounds; if the inclusion never holds the zero is REFUSED with the reason “no contraction”. A sibling certifier for fixed points in weighted sequence spaces uses the radii-polynomial form of the same argument and requires the contraction factor to be proved below one before it states uniqueness.

2.4.2. A maximum in a box

Proposition 4. *Let ℓ be twice continuously differentiable on a convex box X , let $\nabla \ell$ have exactly one zero θ^* in X , and let the Hessian of ℓ be negative definite at every point of X . Then θ^* is the unique maximiser of ℓ on X .*

Proof. A function whose Hessian is negative definite on a convex set is strictly concave there, and a critical point of a strictly concave function is its unique maximiser on the set. $\square$

Negative definiteness over the box is proved one of two ways: by Sylvester’s criterion on the interval Hessian, its leading principal minors enclosed and proved of alternating sign over all of X (for up to three parameters, with the two enclosures of each symmetric entry intersected); or, where the box’s minors are too wide, at the candidate point together with a proof that every Hessian over X is nonsingular — $|I - AH(X)|u < u$ componentwise for a

positive vector u and an approximate inverse A — so that, X being connected and the eigenvalues continuous, no Hessian over X can change its inertia. This is how a maximum-likelihood fit becomes a certificate (Toledo, 2026c). The statement is local: that no better maximum exists elsewhere in the family remains the search’s claim, and every fit is stated with that scope.

2.4.3. A supremum at a limit

Some families have no interior maximum on some data: the likelihood rises toward a limit member, and an optimiser prints the last iterate as the fit.

Proposition 5. *Let $\ell(\mu, \sigma, Q)$ be continuous on $B \times [0, q_1]$ and differentiable in Q on $B \times (0, q_1]$, and let $\partial\ell/\partial Q < 0$ there. Then for every $(\mu, \sigma) \in B$ and every $Q \in (0, q_1]$, $\ell(\mu, \sigma, Q) < \ell(\mu, \sigma, 0)$.*

The proof is the mean value theorem. With $Q = 1/\sqrt{\alpha}$ the generalized gamma reaches the lognormal at $Q = 0$ (Prentice, 1974), and in these coordinates $Q = 0$ is an ordinary point; a proof that the Q -score is negative over a neighbourhood of the lognormal fit refuses the family at its edge *with the proof* that every member there is less likely than its limit. A threshold on α in place of this proof hands a different family the verdict at 544 of 7,767 cell-blocks of the extreme-value atlas (Toledo, 2026c), which is the cost of a threshold measured.

2.4.4. A statement for every input in a box

Four reductions decide a universal statement over a box.

A finite box, exhaustively. Where the box is a finite set — the pairs of a point configuration, the cases of a theorem up to a bound — every element is decided, and the only question is the arithmetic: exact integers, or floating point under a bound that makes it exact (Section 2.3). The work is proportional to the set, and a set too large for the budget is a refusal, never a sample.

Bisection with a proposer. The box is split until each piece’s enclosure decides, a floating-point heuristic choosing the coordinate to split and never a verdict; a depth budget bounds the work, and an exhausted budget is an unresolved refusal. To publish a split refusal, the corners are evaluated as thin boxes and one proved witness is kept on each side.

Monotone implicit functions. Where a model quantity is the root of an equation monotone in its parameters, the root’s range over the box is bracketed by the roots at two corners, and each end counts only when the

equation’s sign just beyond it is proved. Colebrook’s friction law is handled this way: with $x = 1/\sqrt{f}$ the root of $h(x) = x + (2/\ln 10) \ln(a + bx)$, increasing in x , a and b , every root over the box lies in $[x^*(a_+, b_+), x^*(a_-, b_-)]$, and a floating-point Newton iteration only locates the ends (Toledo, 2026d).

Multilinear forms.

Lemma 1. *Let f be multilinear on a box $X \subset \mathbb{R}^n$ (affine in each variable when the others are fixed). Then $f(X) = [\min_v f(v), \max_v f(v)]$, the minimum and maximum taken over the 2^n vertices of X .*

Proof. Fix all variables but one; f is affine in it, so its extremes over that side are at the side’s ends. Induct over the variables to reach a vertex for each extreme. f is continuous and X connected, so every value between is attained. $\square$

An emissions-abatement estimate, a sum of products of premises each declared as an interval, is such a form. The kernel checks multilinearity syntactically — every term a monomial in distinct variables, or the corner rule refuses to apply — and evaluates the 2^n corners in exact rationals; the enclosure is then the exact range, not an outer bound, and the two corners at its ends are the witnesses of a split refusal.

2.4.5. A sign, exactly

Geometric predicates — which side of an edge a point lies on — are signs of small determinants. Each is evaluated first in double precision and trusted only when its magnitude clears a forward-error bound proved for the inputs (Higham, 2002; Fortune and Van Wyk, 1996): with coordinates bounded by M and $u = 2^{-53}$, the computed 2×2 orientation determinant is within $48uM^2$ of the exact one. Below the bound, the sign is recomputed exactly in integers, each decimal literal scaled to a fixed sixty-four places; a literal that would need more is refused rather than rounded. A point on an edge is reported ON, an answer a floating-point test does not have. This is the static filter of exact geometric computation (Shewchuk, 1997; Yap, 1997), written against the literals a file contains rather than against their doubles. Over a contour benchmark it decided 6.37×10^8 predicates, 237,173 of them in integers (Toledo, 2026c).

2.4.6. A polynomial, nonnegative

A claim $p \geq \gamma$ is certified by an identity $p - \gamma = \sum_i c_i q_i^2$ with rational polynomials q_i and rational weights $c_i \geq 0$, checked by exact coefficient comparison; a negative weight or a nonzero residual refuses, and the residual

is printed. Finding the decomposition is a semidefinite program (Parrilo, 2003; Peyrl and Parrilo, 2008) that stays outside the trust path: the solver proposes, the identity decides. A separate refuter evaluates p at an exact witness point and refutes the claim if $p(w) < \gamma$.

2.4.7. *A minimum of a cosine polynomial*

The minimum over x of $\sum_{a \in A} \cos(ax)$ is the minimum over $y \in [-1, 1]$ of $P_A(y) = \sum_a T_a(y)$, a polynomial with integer Chebyshev coefficients. The certifier isolates the critical points with a Sturm chain of the square-free part of P'_A on a slightly enlarged interval, refines each by interval Newton in rationals with outward balls around exact Horner values, encloses P_A on each critical interval with the bound $a^2(a^2 - 1)/3$ of V. Markov on the second derivative, adds the endpoints ± 1 exactly, and returns the hull of the smallest enclosures. A stronger certificate identifies the minimum as an algebraic number — a resultant, a Sturm count of one inside the enclosure, irreducibility modulo a prime — and is CERTIFIED only when all of these hold.

2.4.8. *A formal proof, built and read*

A formal certificate is two claims. That the proof is accepted is decided by building it at the pinned commit, with the toolchain and library versions the claimant pinned, and asking the kernel which axioms the result depends on; where the claimant’s checker allows it, the proof is replayed by a second kernel written independently of the first (Lean FRO, 2026; ammkern, 2026). That the statement proved is the one claimed is not a computation: the statement is read hypothesis by hypothesis against the published theorem and the library’s definitions, the reading is recorded with its slips and weak points, and it never issues a verdict of its own — the verdict is the kernel’s, scoped by what the reading found absent. Section 3.2 is the worked case.

2.4.9. *Proposers*

Every certifier above takes its candidate from a floating-point search — Nelder–Mead and Levenberg–Marquardt for a likelihood, Newton for a root, sampling for a minimum, a semidefinite solver for a decomposition, a language model for a construction — and none of these is ever in the trust path. A proposer may be wrong, slow or adversarial; it can cost a refusal, never a false verdict. The same rule governs forecasts in the system’s applications: a forecaster enters as a proposer of boxes, is graded against what is later observed, and stops being admitted when it misses its declared coverage.

2.5. The verification discipline

Proposition 1 makes the verdicts as sound as the code that computes the extensions. The discipline is how the system guards that code. Its rules are stated as engineering because each is a line of code that refuses — a battery that exits non-zero, a page builder that will not write its page, a numbers generator that will not write a paper’s macros — and this section says where each refusal sits.

Screens prune, never admit.. Floating point decides only what is worth certifying. Every family of the generation loop declares a cheap screen that may only discard candidates and a certifier that is the only authority; the loop can skip on a screen and can record a hit only from the certifier. A screen is itself a claim that it drops nothing it should keep, so the campaign runner plants known answers that must pass every stage and certify before a run may start, refuses a run whose filters lose one, and requires a deliberately simple baseline beside every campaign so that a clever proposer is measured against something (Toledo, 2026e).

Red controls must fire.. A check that has never gone red is decoration. A red control is a deliberately wrong variant of a test that must be rejected, and a battery whose red controls do not all fire fails, whatever its other checks say. Examples: a determinant whose double-precision value is $+3.6 \times 10^{-15}$ and whose exact sign is negative, which the filter must send to exact arithmetic; a “gate” that evaluates the floating-point model at the centre of the box, which is what a point pipeline does and which the corners must catch; a missing inner certificate, which must refuse a theorem; a bar on the wrong side of an endpoint, which must be refused by name; a seventh point of equal norm in the plane, which the pairwise test must find. The test target runs 130 batteries, 83 in JavaScript and 47 in Python, and a wiring check refuses a tree in which that target and the control build’s list differ. In the six-lane audit of machine-written manuscripts, 21 mutation controls stand beside 92 named checks, and the campaign runner ships 19 ways to cheat its own reward signal, each caught by a named control (Toledo, 2026f). The methods note records 16 defects the project shipped and then found: 10 caught by an impossible number, a calibration, a control, a byte pin or an outside read, and 6 by the adversarial read of the code made for this paper (Section 4.4), counted apart (Toledo, 2026e).

Calibrate before you decide.. A new path — a new machine, a new arithmetic, a new kernel — decides nothing until it has reproduced a decision

already made by an old one. The GPU kernel of Section 3.1 was required to return the laptop’s exact contact counts on two smaller configurations before it was allowed to decide the seven large ones; the Leech lattice’s 196,560 and Eg’s 240 are re-derived at the start of every kissing run; and a pre-registration states, before a paid run, what will be measured, what counts as a result, and at what cost the run stops.

Pin the bytes.. Every claim is decided against a held byte sequence named by path and SHA-256 and re-hashed at decision time: the claimant’s file, the dataset, the printed table transcribed digit for digit, and the certifying modules themselves, whose digests a ledger records and whose workers refuse other bytes. A page that reports a record refuses to build when a pinned input has changed, when the record was written by a different version of its verifier, or when its battery does not pass; an application build that regenerates a record and finds it different from the committed one refuses and leaves the committed record in place. Files lifted from the laboratory this system grew out of come through a manifest of their source paths and digests.

Independence from the claimant.. The claimant’s program is never in the trust path: the decision is re-derived from the published statement and the published bytes, and where a claimant’s script was read to settle which identity a manuscript meant, the record says so and the script was not executed. An unqualified “no shared code” would not survive a check, because two of the system’s own instruments reuse its own code across a producer–checker line; the precise form adopted is that the system *shares no code with the claimant*, and every certificate re-runs with no engine present (Toledo, 2026h).

Same-author second implementations.. Where a verdict is the product, a second program written from the specification, in another language and arithmetic and sharing no code, re-derives it (Table 2). These are N-version checks by the same author from the same specification on the same inputs: they catch a slip in one arithmetic and cannot catch a shared misreading, and they are counted apart from re-derivations by other parties (Section 4.2), of which the record holds 1. Not every certifier has one; Section 4.4 names those that do not.

Records re-run.. A record is not believed because it was written; it is re-derived. Published records re-run in the reader’s browser by the bytes that wrote them and are compared with the record — by digest for a day of

decision	the two implementations	recorded agreement
engineering gate	JavaScript intervals; Python decimal at fifty digits	inside the gate’s enclosures at 176 inputs, tightest margin 9.0×10^{-12} bar
abatement kernel	JavaScript exact rationals; Python fractions	8 scenario versions character for character, 2,400 interior points
wave-window decisions	JavaScript exact rationals; a clean-room Python verifier from the written specification	every published decision of a day re-decided, the build refusing on one disagreement
kissing pairs	numpy or a GPU kernel on integer rows; plain Python integers and a BigInt JavaScript reading on seeded samples	every sample’s contact count equal in all three, in all 19 configurations
δ_3 certificates	the claimant’s Python verifier; a clean-room JavaScript verifier	5 of 5 certificates verified by both (Toledo, 2026i)
a Hankel-determinant wall	a lifted arbitrary-precision engine; a standard-library cross-check	51 rows re-run bit-identical, 6 of 6 cross-checks agreeing

Table 2: Decisions re-derived by a second implementation that shares no code with the first — written by the same author from the same specification, so an N-version check rather than an independent re-derivation. Each agreement is read from the record that holds it.

wave-window decisions and a cell of the extreme-value atlas, field by field for an engineering receipt and an abatement registry. A rerun kit lists 31 commands that re-derive the register’s records or verify them with the standard library alone; its one recorded execution, on 2026-10-02, ran 23 of them in about 22 minutes on a laptop and compared 88 register rows before and after: 0 moved, 0 failed. And the system’s own results are offered to outside parties to re-certify without its code; 1 such rerun is recorded so far, the $\lambda(4)$ audit of Section 3.4.

Honest counting. Every row is derived from a record; a claim with no record gets no row. An aggregate audit counts once and names its most frequent defect. A correction the system adds beside a refuted claim is never counted among the audited corpus’s rows; a queued row is not a decided row; refusal kinds are never merged into one total; and every number in this paper and its companions is a macro written by a generator that refuses when a record

no longer supports the sentence that quotes it (Toledo, 2026f).

3. Benchmark exercise

Three exercises test the method on claims that machines produced, one per kind: a configuration (mathematics), a formal proof (Navier–Stokes), and a grader (computer science). Each is told the same way — the claim, how it is read, how it is decided, the verdict with its witness, what it cost, and what the claimant did next — and every number is a field of the record that decided it.

3.1. Mathematics: nineteen kissing-number bounds from an autonomous discovery system

The claim.. In September 2026 a group at Zhejiang University published, with a paper describing an autonomous multi-agent research system, new lower bounds for the kissing number (Conway and Sloane, 1999) in 19 dimensions (25, 27, 32–39, 43, 45 and 49–55) (Yang et al., 2026): in each, a configuration of unit vectors no two of which are closer than sixty degrees, larger than the previously recorded one by between one point and a few thousand. The configurations are built from lattices, codes, sections and projections; their counts are proved by moment identities and the angular condition by structure, and the release’s own verification files record that the pairwise inner products were not checked exhaustively.

The reading.. A kissing configuration is the simplest universal claim there is: a finite set of points and, for every pair, one inequality. Read as Definition 2, the box is the finite set of pairs and the predicate is $\langle x, y \rangle \leq \frac{1}{2}|x||y|$. With integer coordinates the predicate is exact: for rows of squared norms A and B with inner product p , the angle is at least sixty degrees if and only if $p \leq 0$ or $4p^2 \leq AB$ — a test that also rejects a repeated or parallel row, so that distinctness of directions is part of it — and a contact, exactly sixty degrees, is $4p^2 = AB$.

The decision.. Two configurations, in dimensions 25 and 27, were published with the paper as exact objects and decided at once. The other seventeen had no published point list and read NEEDS DATA, the bytes that would decide them named; a request on the repository’s public thread was answered with the dimension-43 and -45 points five days later and, after those were decided, with every remaining configuration within hours (Oxelra-AI, 2026). Each file is pinned by the digest the claimant’s manifest and the hosting

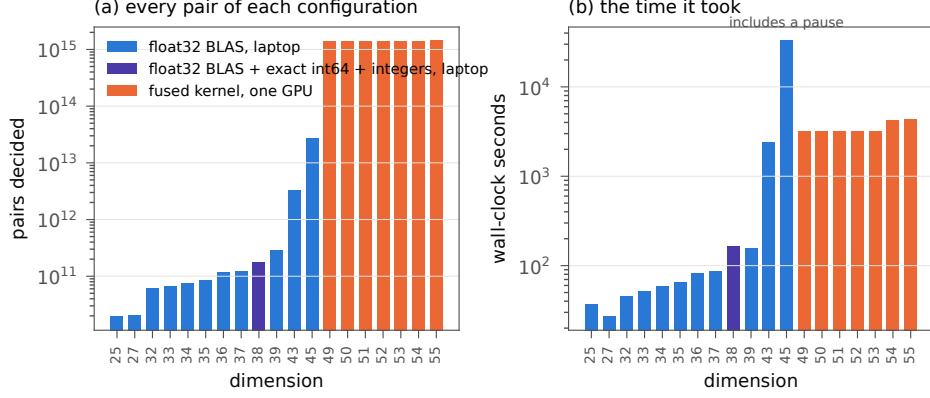


Figure 2: The 19 configurations: (a) the pairs decided in each dimension and (b) the wall-clock time, by arithmetic (certs/kissing-wave.json). The seven largest ran on one GPU; the dimension-45 bar includes a pause.

platform report, and the tool refuses a file that does not match. For every configuration the row count and the counts per squared norm are checked against the claimant’s, the rows are shown to span exactly d dimensions (the rank over $\mathbb{Q}$ of the exact Gram matrix $X^T X$), and every pair is decided: on the laptop by single-precision BLAS tiles, exact because every partial sum is bounded by the largest squared norm (at most 192, against 2^{24}); in dimension 38, where 1,932 rows carry coordinates of 36-digit squared norm, by exact 64-bit integers against those rows (every partial sum below $\sqrt{48N} < 2^{63}$) and by Python integers among them; and in dimensions 49–55, about 1.4×10^{15} pairs each, on one rented GPU by a fused kernel that computes and reduces each 128×128 block of products in registers, half-precision inputs with single-precision accumulation, every value an integer below 2^{11} . The GPU path was first required to reproduce the laptop’s contact counts on dimensions 32 and 39 (358,430,976 and 325,490,632) and did. In every dimension a seeded sample of rows was decided again in plain Python integers with the test written out, and by a separate JavaScript implementation in arbitrary-precision integers, and the three agree on every contact count. Dimensions 43 and 45 were decided a second time in the form the construction states — integer coefficient rows over the claimant’s published lattice basis and its exact Gram matrix, every row checked to be a point of the section or projected fibre the construction defines — with the same contacts.

d	points	pairs decided	at exactly 60°	arithmetic	time
25	197,580	19,518,829,410	446,074,805	float32 BLAS	37 s
27	201,567	20,314,526,961	444,139,680	float32 BLAS	27 s
32	347,584	60,407,144,736	358,430,976	float32 BLAS	45 s
33	363,968	66,236,170,528	368,310,912	float32 BLAS	52 s
34	384,196	73,803,091,110	380,481,792	float32 BLAS	59 s
35	409,676	83,917,007,650	315,579,800	float32 BLAS	1 min
36	484,760	117,495,886,420	404,388,832	float32 BLAS	1 min
37	498,024	124,013,703,276	392,397,168	float32 BLAS	1 min
38	591,900	175,172,509,050	379,853,270	float32 BLAS, exact int64, integers	3 min
39	763,668	291,594,025,278	325,490,632	float32 BLAS	3 min
43	2,553,792	3,260,925,512,736	4,467,331,098	float32 BLAS	40 min
45	7,380,090	27,232,860,514,005	19,832,543,736	float32 BLAS	9.1 h
49	52,430,156	1,374,460,602,877,090	965,197,554,294	fused GPU kernel	53 min
50	52,458,468	1,375,945,406,224,278	964,169,164,800	fused GPU kernel	53 min
51	52,500,930	1,378,173,799,181,985	962,629,120,340	fused GPU kernel	53 min
52	52,585,772	1,382,631,682,125,106	959,560,825,568	fused GPU kernel	53 min
53	52,698,696	1,388,576,253,700,860	955,491,412,444	fused GPU kernel	53 min
54	52,923,774	1,400,462,900,739,651	947,433,469,328	fused GPU kernel	1.2 h
55	53,301,140	1,420,505,735,999,230	934,069,080,624	fused GPU kernel	1.2 h

Table 3: The 19 configurations of (Yang et al., 2026), every pair decided (certs/kissing-wave.json). Times are wall-clock: the laptop’s eight cores for the BLAS rows (the dimension-45 run includes a pause), one NVIDIA L40S for the fused kernel. In all, 9,752,282,639,769,360 pairs and 6,716,665,650,099 contacts; no pair closer than sixty degrees.

The verdict.. WITNESSED in all 19 dimensions (Table 3, Figure 2): 9.75×10^{15} pairs decided, 6.72×10^{12} of them at exactly sixty degrees (the exact counts are in the table), none closer; every count as published; every span exactly d . Every bound in the paper holds.

The cost.. The 12 laptop rows took 35,954 seconds of wall-clock in all; the 7 GPU rows 6.8 GPU-hours at \$1.09 per hour, \$8.15 billed (corpus/kissing-wave/qiushi-gpu-2026-10-12/pod.json). The exactness argument is one inequality per path, checked against the data before the run; nothing in the cost is spent on rigour.

What the claimant did next.. It published its points, in exactly the form the check needed, and linked the check from both of its READMEs beside its results table. What the check does not say is also recorded: “decided here” is not “the record” — the table of kissing-number bounds that the field maintains (Cohn, 2026) lists higher values in dimensions 25–31 and 33, from constructions whose points have not been published and which this system has not decided.

3.2. Navier–Stokes: a Lean certificate read against its claim

The claim.. In September 2026 OpenAI announced a proof of finite-time blowup for the forced three-dimensional Navier–Stokes equations — one of the four alternatives of the Clay problem — with a Lean repository of 2,486 files and 618,762 lines (the Navier–Stokes library, an Euler library and the challenge statements), proved against a challenge statement that DeepMind’s Formal Conjectures project had written months earlier (OpenAI, 2026b; Google DeepMind, 2026). The announcement’s claim was that the theorem had been machine-checked.

The reading.. A formal certificate is two claims (Section 2.4): that the kernel accepts the proof, which a build decides, and that the statement proved is the theorem claimed, which no build decides. The second is read: Clay’s wording (Fefferman, 2000), the paper’s Theorem 1.1, and the formal statement’s hypotheses and conclusions, side by side, with every difference recorded — and the reading issues no verdict of its own, so that a weak point in a 166-page manuscript is a note and not a refutation.

The decision.. 24 files were pinned on 2026-09-09. The repository was built here at the announced commit 8937a8f4 over 2,486 modules of its own (11,251 with the library it imports) in 140 minutes on the laptop, with only the standard axioms reported by the kernel; the claimant’s comparator accepted both challenge theorems; a non-vacuity file of 7 declarations compiled against the result. The reading found the formal statement to be Clay’s alternatives (C) and (D) with nothing added to the competitor class and nothing dropped from the data conditions; 6 steps of the manuscript re-derived by hand; 5 slips in its arguments recorded, none affecting a conclusion; 13 weak points named. And it found one clause — the uniform energy bound $\sup_{0 \leq t < 1} \|u(t)\|_{L^2} < \infty$, present in the paper’s Theorem 1.1 — absent from the formal conclusions at that commit.

The verdict.. PARTIAL at 8937a8f4 (Figure 3): the computational fragment certified by two kernels, the clause not reached, the scope sentence naming it. On 2026-10-06 the next upstream commit, f9e8bc5b (188 files changed, 25,143 lines added), was decided again by the same rule: the clause is now a conclusion of a theorem typed as the breakdown statement, and the row reads CERTIFIED, with the PARTIAL kept as its history (Toledo, 2026a).

The cost.. The build, 140 minutes; the reading, days of a person’s time, which the record does not measure and the paper does not claim as a method.

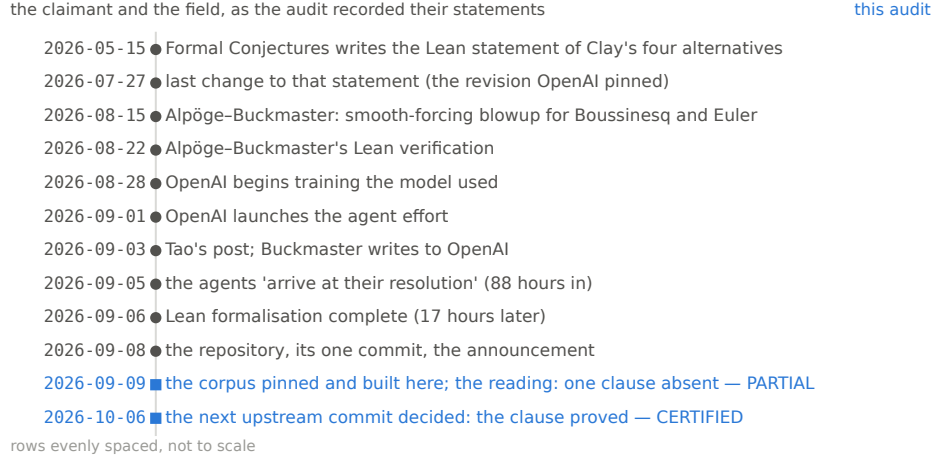


Figure 3: The Navier–Stokes claim in time: the claimant’s and the field’s dates as the audit recorded them from their statements (above the line; corpus/navier-stokes/audit.json), and the audit’s own (below).

What the claimant did next.. The upstream commit proved the clause. An independent paper on the same release argues the general point — that Lean verification of an autoformalised statement does not by itself establish the natural-language theorem (Bastounis et al., 2026); the reading here is the finite instance of it, with the clause named.

3.3. Computer science: the false positives of tolerance graders, and a release audited

The claim.. A grader in a mathematical benchmark asserts that a submitted number is the answer. Nearly every one compares the number with a stored decimal within a tolerance (Hendrycks et al., 2021; Cobbe et al., 2021; Gao et al., 2023; Hugging Face, 2025); the claim is that a submission inside the window is right, and the window is the grader’s author’s choice.

The reading.. The reference answer is itself a claim, read as the box its digits allow or, where a certified enclosure of the quantity exists, as that enclosure. A grader is then a decision procedure that can be graded: a submission provably outside the enclosure that it accepts is a false positive proved, not suspected, and a submission provably inside that it rejects is a false negative proved.

grader	its test	accepts (of 4,000 wrong)	rejects (of 418 right)
absolute-tolerance	$(v - \text{key} < \text{tol})$	89.5%	0.5%
relative-tolerance	$(v - \text{key} / \text{key} < \text{tol})$	90.3%	0.5%
exact-match	$(v == \text{key})$	0.0%	24.4%
enclosure	$(v \text{ in certified } [\text{lo}, \text{hi}])$	0.0%	0.0%

Table 4: Four graders on the same submissions (certs/envs-record.json). The tolerance graders accept most of what is provably wrong; exact string matching rejects a quarter of what is provably right; the enclosure grader — is the submission inside the certified interval — is right on every one.

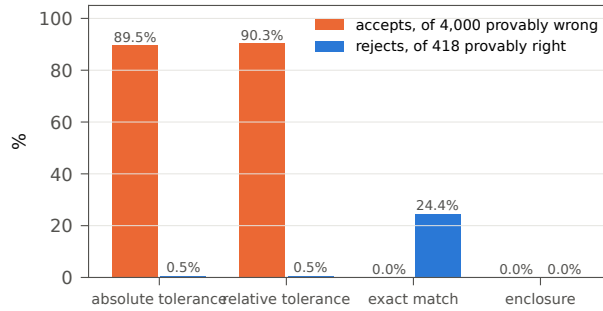


Figure 4: Four graders on the same submissions (certs/envs-record.json): what each accepts of the provably wrong and rejects of the provably right.

The decision.. From 104 facts of the register, each pinned to the certificate that decided it, 4,000 submissions were minted that lie provably outside their enclosures — inside the tolerance window but outside the certificate, at the window’s edge, sign-flipped, truncated, off by a digit — and 418 that lie provably inside. Four graders were run on all of them (Table 4, Figure 4). 34 forgeries were planted in the minting pipeline as red controls; 0 leaked. One of the canaries is not synthetic: it is a number a problem thread published.

The verdict.. The absolute-tolerance grader accepts 89.5% of the provably wrong submissions and the relative-tolerance grader 90.3%: each is REFUTED as a decision procedure, by 4,000 witnesses. Exact string matching rejects 24.4% of the provably right. The enclosure grader accepts 0.0% of the wrong and rejects 0.0% of the right, and is CERTIFIED on this corpus. The adversarial set is not written by hand: a certified enclosure mints it without limit, which is why the certificates had to exist first. A ledger of 92 model rollouts across 11 models exists beside this measurement (Toledo,

2026b); it is a measurement of graders, not of models, and the paper claims nothing about the latter.

The table.. The same release that carries the Navier–Stokes certificate carries 719 machine-written manuscripts in 372 families, with 416 Comparator challenges — Lean statements a proof must satisfy — as its formal spine (OpenAI, 2026a). Every challenge was re-run here under Lean’s kernel and under a second kernel written independently of it (ammkrn, 2026): 398 accepted by both, 18 refused for resources (a stack, a memory or a time limit, on hosted runners and then on rented machines), none rejected. Every formal statement was read against the theorem its manuscript claims: 127 of 242 match; 34 state less than, or other than, what they claim — 11 different, 23 narrower without saying so — and 38 narrower and declared; 39 are not linked to a claim. Of 66 manuscripts whose headline rests on a finite object, 44 were re-decided exactly here, 7 refused, and 15 assert a witness the release does not publish. The 14 gaps between what the release claims and what it checks are ranked in Table 5, found on 2026-10-10 at commit fd4aeeb2 and pinned to file and line before any fix; 6 are headline theorems carried by no checked statement. Nothing in the release was found false.

3.4. *The record beyond the three exercises*

The exercises are drawn from a larger record, and the method runs both ways. The register holds 132 published claims from 25 sources, decided between 2026-08-25 and 2026-10-05: 107 certified, 2 refuted, 9 partial, 5 mixed, 6 repaired, 2 needing data the claimant has not published, and 1 queued and not counted; 107 of the 131 decided rows name nothing wrong (Toledo, 2026f). The defects that appear are of the verifier’s kind: a constant posted to a problem thread as a bare decimal is a floating-point product and parts from the certified enclosure at its 12th decimal place, by a witness that needs one integer inequality over the exact partial product of the primes up to 400,000; witnesses hold only within a verifier’s tolerance in 5 rows; one checker accepts what its own definition excludes. The laboratory’s own theorems are decided by the same rule: $\delta_3 = 117/2192$ (Toledo, 2026i) rests on certificates that two programs sharing no code re-decide, under a battery of 76 checks with 40 red controls; the value of $\lambda(4)$ (Toledo, 2026j) decides 2,231 finite sets across 9 families, and on 2026-09-30 an outside audit with no shared code (rainrzk, 2026) re-derived the argument from the write-up, re-certified every finite case in interval arithmetic and swept the gcd-reduced sets up to 80 as a proof-independent control; its three documentation findings were folded in and no step broke. In engineering, a recommendation to

severity	the gap
1	Hilbert’s sixteenth problem: the uniform bound is in no Lean statement
1	Kaplansky: the headline’s group is torsion-free; the Lean group is required to have torsion
1	Penrose inequalities: no Penrose inequality is stated
1	Coarse assembly: the headline is about the maximal map; the Lean is about the reduced one
1	Projected Lax: the formalized cone has an exact lift, by the family’s own third manuscript
1	Across the release: 34 of 242 formalized families state less than, or other than, what they claim
2	63 displayed definitions Comparator never compares — one of them carries ‘the plane is not 5-colorable’
2	Scope notes credit the formalization with theorems no challenge states
2	The independent kernel was switched on for 2 of 416 challenges — and cannot finish the largest as shipped
3	15 headline witnesses are asserted but not published
3	Papers cite checkers and data the release does not ship
3	Shipped checkers that stop short of what their paper claims — each step re-decided here
3	Printed numbers the text alone does not determine
4	Scope notes that are stale or contradict their own challenges

Table 5: What the release claims beyond what it checks (corpus/openai-math/findings.json), ranked by severity (1 highest). Each row is pinned to the file and line at commit fd4aeb2 and carries the ask that would close it (Toledo, 2026g).

raise water injection by 4.2% on a deep-water flowline is REFUSED (Toledo, 2026d): inputs in the declared envelope give wellhead pressures of 361.6 and 344.2 bar against a limit of 360 bar, each proved at a corner, and the rule is PROVED at a pump discharge of 204.0 bar or less; of 7 injected faults, 6 are refuted and 1 refused. In metocean statistics, 40,778 of 46,602 maximum-likelihood fits over the open-sea cells of a public global hindcast are certified maxima, 5,751 generalized-gamma fits are refused with the proof of Proposition 5, and of the 176 numbers a contour benchmark printed, 173 are the exact count (Toledo, 2026c). Table 6 summarises the record by domain.

domain	what is decided	the check that shares no code
mathematics	132 published claims: constants, constructions, bounds, formal statements	the claimant’s checker is never run; one theorem of the laboratory re-certified by an outside group
engineering	an operating recommendation over a declared envelope; 7 injected faults	a fifty-digit decimal implementation, inside the enclosures at 176 inputs
metocean	46,602 likelihood fits; 176 printed contour counts	a library’s two answers decided as claims; the counts re-derivable in any exact arithmetic

Table 6: The record by domain. Every entry is a row of a public ledger that the build re-derives; the companion papers report each in full (Toledo, 2026f,d,c).

4. Discussion

4.1. What the three exercises show

The defects are the verifier’s kind.. No bound was false, no theorem was false, and no grader lies: in each exercise the gap is between what was claimed and what was checked. A pairwise condition proved by structure and never checked pair by pair; a clause in a theorem’s statement that the formal statement did not carry; a window around a reference answer wider than what the quantity is known to be. These are not the errors a referee reads for, and each was closed by a finite computation whose only hard part was deciding to make it.

The producer’s checker shares the producer’s blind spots.. In every case the claimant had a checker, and in every case it stopped exactly where the claim exceeded it: the release’s verification files record “pairwise inner products checked: false”; the formal statement proved what it stated and not the clause it omitted; the grader accepted what its window admitted. An independent decision is not a better checker; it is a checker that was not written by the claimant and therefore does not know where to stop.

Refusal is a result, and it has a size.. Seventeen of the 19 kissing rows read NEEDS DATA for a month, each naming the bytes that would decide it; the request that named them was answered and the rows closed — an opacity measured, then removed. The Navier–Stokes reading refuses to issue

a verdict and defers to the kernel, so that the clause it found absent became a scope sentence and not an accusation; the next commit closed it. Of the 416 challenges, 18 stay refused for resources, each with the limit that stopped it, and 15 of 66 finite cores stay refused for a witness the release does not publish; the kinds are never summed. The cost of soundness is paid in these refusals, and each says what would convert it.

The cost of exactness is small.. The pairwise check of 9.75×10^{15} pairs cost \$8.15 of GPU time and a few hours of a laptop; the rigour in it is one inequality per path, verified before the run. The kernel runs of 416 challenges cost hosted compute within a budget cap. Exactness here is cheap because the objects are integers and the arguments are bounds, not because the computations are small; what is not cheap, and what this paper does not claim as a method, is the reading of a formal statement against a theorem, which is a person’s days.

A request changes the artifact.. The kissing exercise shows a mechanism the grammar makes possible: a NEEDS DATA row names bytes, the claimant publishes them, the row closes, and the claimant links the decision from its own release. The record of a claim is then no longer only the claimant’s; it carries an outside decision the claimant chose to cite. That is the setting the system is built for, and it is the one outcome a checker written by the producer cannot have.

4.2. Independence, counted

The word “independent” carries three different weights here, and they are counted apart. The system shares no code with any claimant, and every certificate re-runs with no engine present: that is the first weight, and it holds for every row. Same-author second implementations (Table 2) re-derive a verdict in another language and arithmetic from the same specification: they catch a slip in one arithmetic and cannot catch a shared misreading, and that is the second weight. Re-derivation by another party with its own code is the third, and the record holds 1 such rerun, the $\lambda(4)$ audit of Section 3.4. In the exercises: the kissing decisions are of the first weight with a second-weight check in two further arithmetics on every configuration and a calibration of the new path against the old; the formal certificates are replayed by two kernels written independently of each other, which is the third weight for the kernel’s claim and no weight at all for the reading; the graders are decided against certificates the system itself wrote, which is why the measurement is offered with its corpus and its red controls rather than as a result about any model.

4.3. Relation to existing practice

Validated numerics and computer-assisted proof.. Interval arithmetic (Moore et al., 2009; IEEE, 2018), interval methods for nonlinear systems (Neumaier, 1990), verification methods in floating point (Rump, 2010, 1999), the arbitrary-precision and rigorous-integration libraries (Revol and Rouillier, 2005; Johansson, 2017; Kapela et al., 2021) and the computer-assisted proofs they made possible (Tucker, 2002; Hales et al., 2017) are the certifiers’ ancestry, and the system uses them as such. One rung above this system stand the re-certifications of a computer-assisted proof with independent, formally checked code — the Lorenz attractor (Tucker, 2002; Immler, 2018), the nonlinear inequalities of the Kepler proof (Solovyev and Hales, 2013), interval bounds proved inside a proof assistant (Martin-Dorel and Melquiond, 2016); the exact-class certificates of this system are in a form such a checker could take, and the interval-class ones are not yet (Section 4.4). What it adds is not a technique but a setting: the computations are aimed at other parties’ claims, the outcome is one of three published words with a witness, and a failure to decide is a result.

Formal proof.. A proof assistant checks a proof against a stated theorem (de Moura and Ullrich, 2021; The mathlib Community, 2020); what it cannot check is whether the statement is the one intended, and a translation that compiles may not be faithful (Bastounis et al., 2026). The system complements formal proof in two ways: it decides the finite and numerical claims that formal libraries do not yet reach, and it audits formal claims at the level where they can fail — the statement read hypothesis by hypothesis, the build reproduced, the axioms asked of the kernel, the proof replayed by an independent kernel (Toledo, 2026a).

Exact geometric computation.. Floating-point filters with exact fallbacks are standard in computational geometry (Fortune and Van Wyk, 1996; Shewchuk, 1997; Yap, 1997); the contribution here is only to apply them where a benchmark’s number is decided, and to keep ON as an answer. The exact integer products of Section 2.3 are the same idea run the other way: a bound that makes the fast arithmetic exact, so that no fallback is needed.

Verification, validation and uncertainty quantification.. Verification and validation qualify a model; uncertainty quantification gives a dispersion (Oberkamp and Roy, 2010). Neither decides whether a given number holds for every input the engineer has declared, which is the question the epistemic reading of Definition 2 answers; the two are complementary, and the demonstrator sets them side by side (Toledo, 2026d).

Machine discovery and its evaluation.. Systems that propose mathematics — conjectures from numerical coincidences (Raayoni et al., 2021), constructions from program search (Romera-Paredes et al., 2024; Novikov et al., 2025), configurations from an autonomous research loop (Yang et al., 2026) — each come with an evaluator written by the producer, and the benchmarks that score language models on mathematics grade by tolerance (Hendrycks et al., 2021; Cobbe et al., 2021; Gao et al., 2023). The system sits downstream of all of them as an independent decider, and the register and the three exercises are the measure of what that independence finds.

4.4. *What is not claimed, and the limits*

Not claimed.. Interval arithmetic, the Krawczyk operator, floating-point filters, sums of squares and Sturm sequences are not claimed; they are cited. The validity of a declared model is not claimed: a PROVED engineering number holds for the declared model over the declared envelope, not for the sea or the reservoir. A kissing bound “decided here” is a statement about one published configuration, not about the record in that dimension. Nothing is certified in the regulatory sense; the word is used in the mathematical sense of Definition 3 only. Nothing is a probability.

Locality and incompleteness.. A certified maximum is the maximum in its box; a global statement over a family would need a branch-and-bound over the family’s whole parameter space, which the system does not yet run. The system refuses what it cannot decide, and the refusal rate is a property of each campaign, published with it.

Soundness rests on code.. Proposition 1 is a theorem about interval extensions; whether a program computes one is a fact about the program. The defences — second implementations, red controls, pinned bytes, calibrations, outside re-runs (Section 2.5) — reduce that risk and do not remove it, and they are not uniform: some certifiers have a second implementation that shares no code, and some, among them the point-in-polygon decider, the cosine-minimum certifier and the likelihood certifier, are re-run in the browser by the same bytes, which checks the record and not the code.

What a read of the code for this paper found, and what was closed.. Two reads were made for this paper, the author’s on 2026-10-08 and an adversarial one on 2026-10-09; the second found defects under the system’s headline sentence, and they were closed the same day with the correction published beside the old numbers (the repository’s record `certs/erratum-2026-10-09.json` pins the numbers, the commit and the page bytes that carried

them). *Closed*: the closed-form hunt of the generation loop compared a floating-point candidate — a square root, a constant times a rational, a power — against the certified enclosure and counted a miss as a refutation, exact for a rational candidate and not for the rest, so that 54.6 million of the loop’s “refutations” were float comparisons; every candidate is now an exact rational or an enclosure and a refutation is a disjointness, with the same vocabulary and a re-run of the whole corpus. The loop’s grammar had a word, REJECT, that covered both a proved failure and an enclosure straddling the bar, and any unknown word fell into it; one module now holds the three words and the one predicate of Definition 3, a straddle is REFUSED, an unknown word is REFUSED and counted (71 of the cosine family’s 1,579 decisions were straddles so counted). One family certified “exceeds the champion” by comparing lower end with lower end; the bar is now the champion’s enclosure. The interval kernel had no NaN or infinity semantics and a power routine that wrapped at exponents of 2^{31} ; both refuse. The linear branch of the radii-polynomial certifier returned success without the interval check its other branch makes, and accepted a negative Lipschitz bound; the certificate constructor accepted empty falsifiers and undefined evidence; the aggregate test target finished successfully whatever its batteries printed and the control page was written with a red battery shown. All of these refuse now, and none had moved a verdict reported in the companion papers. *Open, and named*: the helper that encloses the platform’s cosine by a derived pad is called from six files including the radii-polynomial certifier of Section 2.4, whose records are re-run when it is retired; one quadrature routine that is rigorous on dyadic cells of $[0, 1]$ is called on another interval by one instrument; one battery labels some positive assertions as red controls; in the engineering demonstrator a PROVED consistency check means that the claim meets an outer enclosure, and the discharge thresholds are rounded outward to a tenth of a bar in floating point rather than in directed rounding; the transcendental tails are computed in round-to-nearest and are safe by a margin of twenty orders rather than by directed rounding; the same-author second implementations of Table 2 are not independent re-derivations; and the exact-class certificates alone are in a form a program other than the engine can check. Each is recorded in the repository’s debt register with the instrument it affects, and the system’s own rule applies to them — a check that has not been made is not a check.

5. Conclusions and recommendations

cert-machine decides the numbers and claims that machines propose with three verdicts, each carrying the object that justifies it, sound under one hypothesis that the arithmetic is built to secure. Its certifiers reduce the questions that arrive — a zero, a maximum, a statement for every input, a sign, a nonnegative polynomial, a minimum, a formal proof — to finitely many exact or outward-rounded decisions; its discipline keeps them honest by code that refuses; and its record is public and re-runs. The three exercises are what the method finds when it is pointed at machine-generated claims of three kinds: every one of 19 record configurations holds, pair by pair; a formal proof is a proof of what it states and not of the clause it omits, until the clause is proved; and the graders that score machines on mathematics accept most of what is provably wrong. What the system offers those who produce such claims is not a second opinion but a different object beside their number: a verdict with its witness, in which a refusal says what would decide it.

Recommendations.. To producers of machine-generated mathematics: publish the object, not only its description — the points of a configuration in one integer form, the witness a theorem rests on, the data a fit was made from — with a digest beside each file, because the finite check is cheap once the bytes exist and impossible until they do; state the scope of every check in the release itself (“pairwise inner products checked: false” is the right sentence, and it was the one that made the exercise of Section 3.1 possible); and when a formal statement is narrower than the theorem it accompanies, say so where the theorem is stated. To those who maintain the tables and records of a field: distinguish a bound that has been decided from outside from one that has only been claimed, and ask for the points. To those who build benchmarks and graders: where a reference answer has a certified enclosure, grade by the enclosure and not by a window, because the window’s false positives can be minted without limit; where it has none, say which digits of the key are known; and plant red controls, because a grader that has never rejected a planted wrong answer has not been tested. To those who verify: refuse where you cannot decide, name the reason, and count the kinds of refusal apart — a result that says what would decide it is worth more than a probability that nothing could.

Declaration of competing interest

The author declares no competing financial or personal interests. The Qiushi Engine group, OpenAI and the maintainers of the benchmarks and tables named here have not been consulted on this draft; the exchange with the Qiushi Engine group took place on the public thread cited, and their READMEs link the checks of Section 3.1. The reviewing group is being consulted on this manuscript.

CRedit authorship contribution statement

Carlos Toledo: conceptualization, methodology, software, validation, investigation, data curation, writing — original draft, writing — review and editing. The author list and the roles are to be agreed with the reviewing group.

Data availability

Every record, certifier, figure source and table source is public in the cert-machine repository (<https://github.com/carlostoledo1891/cert-machine>, commit f7662d3b8392): the register `certs/claims-ledger.json`; the kissing configurations' decisions `certs/kissing-wave.json` and the GPU pod's results and cost `corpus/kissing/wave/qiushi-gpu-2026-10-12/`; the Navier–Stokes audit `corpus/navier-stokes/` (the manifest of pinned files, the build, the reading, the upstream decision); the openai/math audit `corpus/openai-math/` and `certs/openai-math-*.json`; the grader measurement `certs/envs-record.json`; the batteries, the rerun kit and the outside reruns under `certs/` and `corpus/`. The macros of this manuscript are written by `tools/paper-numbers/cert-machine-method.js` and by the generators of the companion papers from those records, and a record that no longer supports a sentence makes the build refuse. The claimants' data are cited at their sources and pinned by digest in the records; the Qiushi Engine's point sets are CC BY 4.0, and the OpenAI repositories carry their own licences.

Declaration of generative AI and AI-assisted technologies

During the preparation of this work the author used large language models (Claude, Anthropic) to draft code, exposition and this manuscript in a certified-verification setting: constructions and prose were proposed with

the help of the models, and every quantitative claim is decided by outward-rounded interval arithmetic or exact rational arithmetic, with adversarial controls that must fire for a record to be accepted. After using the tool the author reviewed and edited the content and takes full responsibility for the content of the publication. The records, not the models, are the evidence offered; they are public and re-runnable.

Funding

This work received no external funding. The rented compute of Section 2.1 (\$8.15 for the seven largest kissing configurations; the challenge re-runs within a pre-registered cap) was paid by the author.

Appendix A. Reproduction

Every constant in this paper is interpolated from a certificate record in the repository, so the document cannot quote a number the machine did not certify. The three exercises re-derive with the commands below, each of which downloads the claimant’s files and refuses them unless they match the pinned digests.

- Kissing, dimensions 32–37 and 39: `python3tools/run-kissing-configs.py<d>` (one to three minutes each on an eight-core laptop); dimension 38: `python3tools/run-kissing-config38.py` (three minutes); dimensions 43 and 45: `python3tools/run-kissing-sections.py<d>` (forty minutes and about six hours of computation); dimensions 49–55: `tools/runpod-qiushi-lifting.sh` on a CUDA machine (6.8 GPU-hours on one NVIDIA L40S), which calibrates on 32 and 39 before deciding anything; the Gram-form re-decision of 43 and 45: `python3tools/run-kissing-gram.pyqiushi<d>`.
- Navier–Stokes: the build and the kernel runs as `corpus/navier-stokes/build.json` records them (140 minutes on the laptop at the pinned toolchain); the reading is `corpus/navier-stokes/audit.json`; the upstream decision `corpus/navier-stokes/upstream-f9e8bc5b.json`.
- The graders: `nodeinstruments/envs/battery.js` re-mints the submissions from the pinned facts and re-grades them; the openai/math kernel runs are the workflow `.github/workflows/openai-math-kernel.yml` (hosted runners) and `tools/runpod-openai-math.sh`

(rented machines), the statement readings and finite decisions the readers under `tools/`.

- The whole register: the rerun kit (31 commands, `RERUN.md`); its recorded execution of 2026-10-02 ran 23 of them in 22 minutes.

The references of this paper are themselves checked: `tools/check-bib.js` resolves every entry at its source (Crossref, DataCite, the arXiv API or a fetch), records the resolution in `corpus/bib/`, and the build’s battery refuses an entry with no verified row or whose title the source did not match.

References

- Abramowitz, M., Stegun, I.A., 1964. Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. volume 55 of *Applied Mathematics Series*. National Bureau of Standards, Washington. URL: <https://personal.math.ubc.ca/~cbm/aands/>.
- ammkrm, 2026. `nanoda_lib`: a type checker for Lean 4 export files. URL: https://github.com/ammkrm/nanoda_lib. commit 3a240721, with the patch of issue #44.
- Bastounis, A., Circelli, F., Hansen, A.C., 2026. Navier–Stokes lost in translation: why Lean verification of AI autoformalisation does not guarantee correct natural language proofs. [arXiv:2610.08144](#).
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., Schulman, J., 2021. Training verifiers to solve math word problems. [arXiv:2110.14168](#).
- Cohn, H., 2026. Kissing numbers. URL: <https://cohn.mit.edu/kissing-numbers>. the table of bounds, read 2026-10-11; the extract is pinned in the repository.
- Conway, J.H., Sloane, N.J.A., 1999. Sphere Packings, Lattices and Groups. 3rd ed., Springer, New York. doi:10.1007/978-1-4757-6568-7.
- Dekker, T.J., 1971. A floating-point technique for extending the available precision. *Numerische Mathematik* 18, 224–242. doi:10.1007/BF01397083.
- Fefferman, C.L., 2000. Existence and smoothness of the Navier–Stokes equation. URL: <https://www.claymath.org/millennium/navier-stokes-equation/>. the Clay Mathematics Institute’s official problem description.

- Fortune, S., Van Wyk, C.J., 1996. Static analysis yields efficient exact integer arithmetic for computational geometry. *ACM Transactions on Graphics* 15, 223–248. doi:10.1145/231731.231735.
- Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac’h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., Zou, A., 2023. EleutherAI/lm-evaluation-harness: major refactor. doi:10.5281/zenodo.10256836. the Language Model Evaluation Harness, version 0.4.0, Zenodo.
- Goldberg, D., 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys* 23, 5–48. doi:10.1145/103162.103163.
- Google DeepMind, 2026. Formal conjectures: a collection of formalized conjectures in Lean. URL: <https://github.com/google-deepmind/formal-conjectures>.
- Hales, T., Adams, M., Bauer, G., Dang, T.D., Harrison, J., Hoang, L.T., Kaliszyk, C., Magron, V., McLaughlin, S., Nguyen, T.T., Nguyen, Q.T., Nipkow, T., Obua, S., Pleso, J., Rute, J., Solovyev, A., Ta, T.H.A., Tran, N.T., Trieu, T.D., Urban, J., Vu, K., Zumkeller, R., 2017. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi* 5, e2. doi:10.1017/fmp.2017.1.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., Steinhardt, J., 2021. Measuring mathematical problem solving with the MATH dataset. *arXiv:2103.03874*. neurIPS 2021 Datasets and Benchmarks.
- Higham, N.J., 2002. Accuracy and Stability of Numerical Algorithms. 2nd ed., SIAM, Philadelphia. doi:10.1137/1.9780898718027.
- Hugging Face, 2025. Math-Verify: a mathematical expression evaluation system for LLM benchmarks. URL: <https://github.com/huggingface/Math-Verify>.
- IEEE, 2018. IEEE standard for interval arithmetic (simplified). doi:10.1109/IEEESTD.2018.8277144. IEEE Std 1788.1-2017.
- IEEE, 2019. IEEE standard for floating-point arithmetic. doi:10.1109/IEEESTD.2019.8766229. IEEE Std 754-2019.

- Immler, F., 2018. A verified ODE solver and the Lorenz attractor. *Journal of Automated Reasoning* 61, 73–111. doi:10.1007/s10817-017-9448-y.
- Johansson, F., 2017. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE Transactions on Computers* 66, 1281–1292. doi:10.1109/TC.2017.2690633.
- Kapela, T., Mrozek, M., Wilczak, D., Zgliczyński, P., 2021. CAPD::DynSys: a flexible C++ toolbox for rigorous numerical analysis of dynamical systems. *Communications in Nonlinear Science and Numerical Simulation* 101, 105578. doi:10.1016/j.cnsns.2020.105578.
- Krawczyk, R., 1969. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing* 4, 187–201. doi:10.1007/BF02234767.
- Lean FRO, 2026. Comparator. URL: <https://github.com/leanprover/comparator>. commit d03acab1.
- Martin-Dorel, É., Melquiond, G., 2016. Proving tight bounds on univariate expressions with elementary functions in Coq. *Journal of Automated Reasoning* 57, 187–217. doi:10.1007/s10817-015-9350-4.
- Moore, R.E., Kearfott, R.B., Cloud, M.J., 2009. *Introduction to Interval Analysis*. SIAM, Philadelphia. doi:10.1137/1.9780898717716.
- de Moura, L., Ullrich, S., 2021. The Lean 4 theorem prover and programming language, in: *Automated Deduction – CADE 28*, Springer. pp. 625–635. doi:10.1007/978-3-030-79876-5_37.
- Neumaier, A., 1990. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge. doi:10.1017/CB09780511526473.
- Novikov, A., Vū, N., Eisenberger, M., Dupont, E., Huang, P.S., Wagner, A.Z., Shirobokov, S., Kozlovskii, B., Ruiz, F.J.R., Mehrabian, A., Kumar, M.P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., Balog, M., 2025. AlphaEvolve: a coding agent for scientific and algorithmic discovery. *arXiv:2506.13131*.
- Oberkampff, W.L., Roy, C.J., 2010. *Verification and Validation in Scientific Computing*. Cambridge University Press, Cambridge. doi:10.1017/CB09780511760396.
- Ogita, T., Rump, S.M., Oishi, S., 2005. Accurate sum and dot product. *SIAM Journal on Scientific Computing* 26, 1955–1988. doi:10.1137/030601818.

- OpenAI, 2026a. math: manuscripts and Lean challenges. URL: <https://github.com/openai/math>. commit fd4aeeb2.
- OpenAI, 2026b. NavierStokesAndEuler. URL: <https://github.com/openai/NavierStokesAndEuler>. commits 8937a8f4 and f9e8bc5b.
- Oxelra-AI, 2026. Qiushi-Engine-Kissing-Number-Research: the complete point sets. URL: <https://github.com/Oxelra-AI/Qiushi-Engine-Kissing-Number-Research>. releases section-vectors-2026-10-11, configuration-vectors-2026-10-12 and lifting-vectors-2026-10-12, and issue #1; data CC BY 4.0.
- Parrilo, P.A., 2003. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming* 96, 293–320. doi:10.1007/s10107-003-0387-5.
- Peyrl, H., Parrilo, P.A., 2008. Computing sum of squares decompositions with rational coefficients. *Theoretical Computer Science* 409, 269–281. doi:10.1016/j.tcs.2008.09.025.
- Prentice, R.L., 1974. A log gamma model and its maximum likelihood estimation. *Biometrika* 61, 539–544. doi:10.1093/biomet/61.3.539.
- Raayoni, G., Gottlieb, S., Manor, Y., Pisha, G., Harris, Y., Mendlovic, U., Haviv, D., Hadad, Y., Kaminer, I., 2021. Generating conjectures on fundamental constants with the Ramanujan Machine. *Nature* 590, 67–73. doi:10.1038/s41586-021-03229-4.
- rainrzk, 2026. erdos510-lambda4-audit: an independent re-certification of the $\lambda(4)$ argument with no shared code. URL: <https://github.com/rainrzk/erdos510-lambda4-audit>. posted 2026-09-30 on teorth/erdosproblems issue #392.
- Revol, N., Rouillier, F., 2005. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliable Computing* 11, 275–290. doi:10.1007/s11155-005-6891-y.
- Romera-Paredes, B., Barekatin, M., Novikov, A., Balog, M., Kumar, M.P., Dupont, E., Ruiz, F.J.R., Ellenberg, J.S., Wang, P., Fawzi, O., Kohli, P., Fawzi, A., 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 468–475. doi:10.1038/s41586-023-06924-6.

- Rump, S.M., 1999. INTLAB — INTerval LABoratory, in: Csendes, T. (Ed.), *Developments in Reliable Computing*. Kluwer, Dordrecht, pp. 77–104. doi:10.1007/978-94-017-1247-7_7.
- Rump, S.M., 2010. Verification methods: rigorous results using floating-point arithmetic. *Acta Numerica* 19, 287–449. doi:10.1017/S096249291000005X.
- Shewchuk, J.R., 1997. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete & Computational Geometry* 18, 305–363. doi:10.1007/PL00009321.
- Sibidanov, A., Zimmermann, P., Glondou, S., 2022. The CORE-MATH project, in: *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*, pp. 26–34. doi:10.1109/ARITH54963.2022.00014.
- Solovyev, A., Hales, T.C., 2013. Formal verification of nonlinear inequalities with Taylor interval approximations, in: *NASA Formal Methods (NFM 2013)*, Springer. pp. 383–397. doi:10.1007/978-3-642-38088-4_26.
- The mathlib Community, 2020. The Lean mathematical library, in: *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, ACM. pp. 367–381. doi:10.1145/3372885.3373824.
- Toledo, C., 2026a. An audit of the 2026 Navier–Stokes finite-time blowup claim: the formal statement read against Clay’s wording and Mathlib’s definitions, the Lean proof built and its axioms asked of the kernel. URL: <https://carlostoledo.co/reports/navier-stokes.html>. draft; report at <https://carlostoledo.co/reports/navier-stokes.html>.
- Toledo, C., 2026b. Certificate-grounded evaluation: when the answer key is a proof. URL: <https://carlostoledo.co/paper/certified-evaluation.pdf>. draft.
- Toledo, C., 2026c. Certified extreme-value analysis of significant wave height: maximum-likelihood fits and return levels over a public global hindcast, and environmental contours decided exactly. URL: <https://carlostoledo.co/paper/p1-return-levels.pdf>. draft v0.2.
- Toledo, C., 2026d. A decision procedure for machine-proposed engineering numbers: three-valued verdicts over a declared uncertainty envelope, with witnesses and the threshold that decides. URL: <https://carlostoledo.co/paper/p2-decision-procedure.pdf>. draft.

- Toledo, C., 2026e. How the bugs were found: the methods note. URL: <https://carlostoledo.co/reports/methods-note.html>. formerly “None by reading code”; on 2026-10-09 an adversarial read of the code found six, counted apart.
- Toledo, C., 2026f. Independent exact certification of machine-generated mathematics: the first register. URL: <https://carlostoledo.co/paper/register.pdf>. draft.
- Toledo, C., 2026g. OpenAI’s math release: what it claims beyond what it checks. URL: <https://carlostoledo.co/reports/openai-math.html>.
- Toledo, C., 2026h. Positioning: the decisions. URL: <https://github.com/carlostoledo1891/cert-machine.notes/positioning-decisions-2026-09-03.md> in the repository.
- Toledo, C., 2026i. The twelve-block colouring is optimal: $\delta_3 = 117/2192$ — a computer-assisted proof, its certificates re-decided by two programs that share no code. URL: <https://carlostoledo.co/paper/delta3.pdf>. draft; the certificates are archived with the repository release, doi:10.5281/zenodo.23171167.
- Toledo, C., 2026j. The value of $\lambda(4)$: a machine-derived proof completing Mercer’s Section-5 program. doi:10.5281/zenodo.22225861. report at <https://carlostoledo.co/reports/lambda4.html>.
- Tucker, W., 2002. A rigorous ODE solver and Smale’s 14th problem. *Foundations of Computational Mathematics* 2, 53–117. doi:10.1007/s002080010018.
- Yang, S., Zhao, R., Wu, J., Wang, Y., Chen, F., Zhu, K., Li, W., Li, Z., Li, Y., Hong, S., Pan, Y., Yang, J., Deng, T., Mi, J., Chen, H., Yang, Y., 2026. Large-scale autonomous discovery of kissing number constructions. *arXiv:2609.35051*.
- Yap, C.K., 1997. Towards exact geometric computation. *Computational Geometry* 7, 3–23. doi:10.1016/0925-7721(95)00040-2.