Verify-in-the-Loop: Proof-Carrying AI Mathematics

A case study certifying, expanding, and stress-testing the Jacobian counterexample with Claude Code (Opus 4.8) and a signing verifier

Kyle Clouthier
Clouthier Simulation Labs
kyle@simgen.dev https://simgen.dev

July 25, 2026

Abstract

AI now generates mathematics, code, and claims faster than anyone can review them; the limiting resource is no longer generation but *trust*. This is a report on a way of working built for that regime, demonstrated on the hardest live example we could find. In July 2026 an AI helped Levent Alpöge produce a counterexample to the 87-year-old Jacobian Conjecture, and the community's reaction turned on one question: *can you trust mathematics that came out of a machine before anyone has checked it?* We answer it operationally. We report a workflow—a human directing *Claude Code* running *Claude Opus 4.8*, with every mathematical claim compiled and machine-checked by *Attestral*, a verifier that signs an ed25519 certificate only when its own checker passes—and we run that loop, in public view of its own receipts, on Alpöge's counterexample. The loop *verified* the object exactly; *reverse-engineered* its mechanism (a non-nilpotent, degree-3 étale endomorphism); found its *hidden cubic* (a three-cube-root fiber) and built an *infinite tower* of new counterexamples from it; *mapped* the surrounding construction space with a fold-parity obstruction and a uniqueness skeleton; *proved* its natural four-dimensional generalization *obstructed at every compensator degree* by a one-line degree count where a machine search had only sampled; *caught three of our own errors* mid-loop—including a finite-field prime silently collapsing to $p = 3$—and discarded them; and hit an *honest wall* on the nilpotent normal form, where a per-step-verified construction explodes past 200 variables. Days later the same loop, unchanged, verified the counterexample to the Gaussian Moments Conjecture posted in the same wave—evidence that the pattern, not the theorem, is what generalizes. The point is none of these results individually. The point is that **every one of them carries a certificate any reader can re-run offline**: the load-bearing structural theorems are proved in the *Lean kernel* (20 kernel proofs), the exploratory identities checked by *exact symbolic computation* (22 signed certificates), the two tiers never blurred—and the mathematics was proposed by AI and adjudicated by machine, so that the errors were caught by the same mechanism that signed the successes. An interactive companion with one-command re-verification is at https://simgen.dev/attestral/jacobian-counterexample/.

Contents

1 The problem this workflow exists to solve

Mathematics, code, and claims are now generated faster than humans can review them. Alpöge's counterexample [1, 6, 4] arrived as a short announcement with an AI credited as collaborator and no preprint. Within days Tao published a digestion [8], the Secret Blogging Seminar dissected the fiber [13], Zhang cataloged corollaries [9], and Shaska posted a graded analysis [10]. The popular coverage framed the moment as a referendum on trusting AI-produced mathematics.

The honest response to "I don't trust it" is not "trust me"—it is "here is the check; run it." This paper is a demonstration that such checks can be produced *at the speed the mathematics is generated*, by making the generator and the checker two different agents, and by signing the checker's verdict so the reader re-runs it rather than believing anyone. We built such a loop and pointed it at the most scrutinized AI-math result in the world. What follows is the account of that run: the apparatus first (§2), then the work as a sequence of episodes, each an exchange between a proposing AI and an adjudicating verifier, each ending in a signed receipt or a refutation (§§3–10), then the ledger the loop produced (§11) and an honest account of what the method can and cannot do (§12).

What we claim, and what we do not. We did *not* discover the counterexample; Alpöge did. Our contribution is the *workflow* and its auditable output, held to an explicit honesty ladder. The exploratory identities in the episodes below are exact symbolic computations—machine-checked and signed; we say "machine-checked," not "proved." The *load-bearing structural theory*, however—the Alpöge anatomy, the canonical-form/nilpotency equivalences (§9), and the four-dimensional obstruction (§7)—was subsequently re-proved in the *Lean 4 kernel*: 20 axiom-audited kernel proofs, every one listed in Table 2, which we *do* call "formally proved." A claim that is a theorem carries a Lean proof; a claim that is an exploratory identity carries an exact-symbolic certificate; we never present one as the other. Novelty we surveyed but do not overclaim: one structural result overlaps Shaska [10] and is credited, and the transport/cone step is elementary classical geometry. Importance a certificate cannot confer, and we claim none on the open plane case.

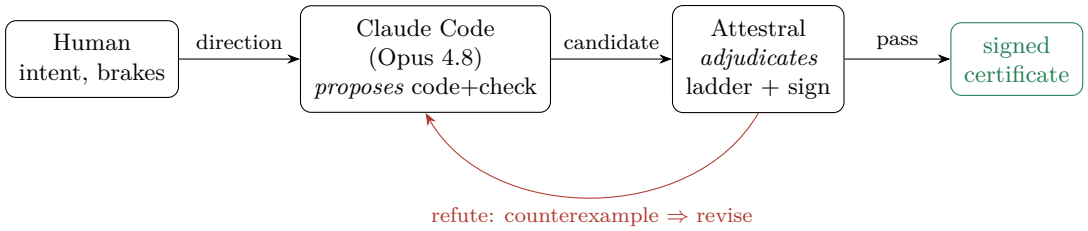

Figure 1: The loop. The proposer (an AI) and the adjudicator (a signing verifier) are different agents; a result becomes a certificate only when the checker, not the model, accepts it. A refutation is not a failure but the steering signal for the next proposal.

2 The apparatus: a human, Claude Code (Opus 4.8), and a signing verifier

The loop has three agents with strictly separated roles (Figure 1).

The human sets intent and direction ("verify this," "why does it fold three-to-one," "try the nilpotent normal form"), and—crucially—applies the brakes: honesty, scope, and the decision to stop.

Claude Code, running Claude Opus 4.8, is the *proposer*. It writes the candidate mathematics as *runnable code with its own check* (a symbolic identity, a property test, a fiber computation), reasons about structure, and revises from counterexamples. It can propose anything; it cannot certify anything.

Attestral is the *adjudicator*. Exposed to the model through a custom Model Context Protocol (MCP) server, it compiles the candidate, runs a language-specific checker ladder, and—only if the check passes—emits an `ed25519`-signed certificate binding the exact code, the verdict, and a content hash. A refutation returns the counterexample to the proposer.

Why the separation is the whole idea. Because the proposer cannot sign its own work, the output is *proof-carrying rather than model-asserted*. This is exactly the property the AI-math trust panic demands: you need not decide whether to believe the model, because the certificate re-runs the check on your machine. The same separation is what let the loop catch *our own* mistakes (§8): the adjudicator has no stake in the proposer being right.

What Attestral is. Attestral is a language-agnostic verification layer: it exposes, through the MCP surface the model calls, a checker ladder for ten languages (Rust, C, Python, TypeScript, JavaScript, Go, Lean, Dafny, F*, TLA$^+$), six of which reach a *formal* rung—Rust/Kani and C/CBMC (bounded model checking), the Lean kernel, Dafny and F* (SMT), and TLA$^+$/TLC. For a given candidate it runs the strongest rung that candidate can carry and, on success, signs a self-contained certificate. The engine internals (the ladder orchestration, the refine loop, the private signing seed) are not part of this paper; what the paper exposes—and all a reader needs—are the *outputs*.

What a certificate is, and what re-verifying it asks of you. A certificate is an exact reproducible computation bound by a SHA-256 content hash and an ed25519 signature. It is *not*

a proof-assistant proof; where the statement is a decidable polynomial identity, a passing verdict is decisive, but we say *machine-checked*, not *proved*. Certificates in this work come in two tiers, and the ledger (§11) labels every one: 23 were discharged on the *Python exact-symbolic ladder* (sympy, Hypothesis, CrossHair; Table 3)—22 exploratory identities plus the Gaussian-Moments companion (§10)—and 20 on the *Lean 4 kernel* rung (Mathlib, axiom-audited, no `sorry`) for the load-bearing theorems—the latter being formal proofs in the strict sense. The signature and the tier answer *different questions*: the ed25519 signature proves *authenticity* (this exact code, this verdict, issued under the published key, untampered since); the tier states *strength* (kernel-proved vs. machine-checked). A certificate is always both signed and tiered; neither substitutes for the other. Crucially, **re-verifying a certificate requires nothing from the engine.** The public verifier `verify_receipt.py` depends only on a hash library and an ed25519 library: it re-checks that the code hashes to the certificate, that the receipt is intact, and that the signature is valid under the published key—instant and offline—and can optionally re-run the (standard, public) ladder to reproduce the verdict. A reader therefore confirms our results with the public certificates, the published key, and a small standalone script, trusting neither us nor the AI, and without access to the engine.

3 Episode 1 — verifying the object

The first thing the loop did was refuse to take the counterexample on faith. Claude Code wrote the map and its two defining checks; Attestral ran them exactly.

$$
\begin{aligned}
F_1 &= (1+xy)^3 z + y^2(1+xy)(4+3xy),\\
F_2 &= y + 3x(1+xy)^2 z + 3xy^2(4+3xy),\\
F_3 &= 2x - 3x^2 y - x^3 z.
\end{aligned}
$$

Claim 1 (the object is a counterexample)**.** $\det JF \equiv -2$ *(a nonzero constant), and the three distinct points* $(0,0,-\frac{1}{4}), (1,-\frac{3}{2},\frac{13}{2}), (-1,\frac{3}{2},\frac{13}{2})$ *all map to* $(-\frac{1}{4},0,0)$*: a local isomorphism everywhere that is not injective.*

✓ exact identity + collision `[377ceaee812182f4]`. Before trusting even the transcription, the loop cross-checked provenance two ways: the polynomials match John Cook's exposition verbatim, and a second, disjoint collision reported there, $(0,1,-4), (-2,1,2) \mapsto (0,1,0)$, reproduces exactly on our map. We are adjudicating Alpöge's object, not a mis-transcription (Figure 2).

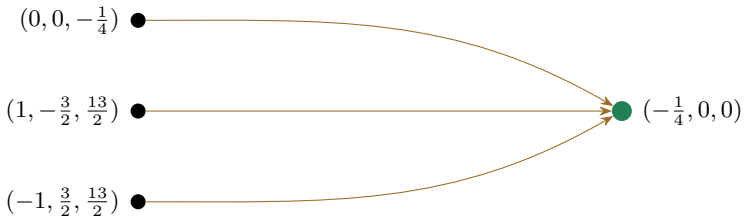

Figure 2: Episode 1's output in one picture, signed `377ceaee`.

4 Episode 2 — reverse-engineering the mechanism

The human asked *why* it works. The model proposed a factorization; Attestral checked it.

Claim 2 (mechanism). *$F = L \circ G$ with $L(x, y, z) = (z, y, 2x)$ linear ($\det L = -2$, so the "-2" is cosmetic). The essential map $G = L^{-1} \circ F$ has $\det \mathrm{J}\, G \equiv 1$, is not injective, and, writing $G = X + H$, has $\mathrm{J}\, H$ non-nilpotent.*

✓ $\det \mathrm{J}\, G = 1$, non-nilpotent [403eb3ed0ef74fab]. This is *why* the object evaded the classical program: the Drużkowski/Yagzhev searches combed the cubic-homogeneous *nilpotent* normal form, and Alpöge's small representative is non-nilpotent. We add the honest nuance the loop later forced on us (§9): by Bass–Connell–Wright a nilpotent counterexample must still exist in higher dimension, so this is a feature of the small representative, not a gap the search "missed."

5 Episode 3 — the hidden cubic

Asked what shape the non-injectivity takes, the model proposed a covering structure and computed the generic fiber; Attestral's exact Gröbner solve adjudicated it.

Claim 3 (degree-3 étale, cube-root fiber). *G is a local isomorphism at every point yet generically three-to-one: the fiber over $(2, 1, 1)$ is exactly three points whose z-coordinates are the three cube roots $-\frac{779}{4} + \omega R^{1/3} - \frac{45039}{16}\omega^{-1} R^{-1/3}$ ($\omega^3 = 1$) of a common radical—a Cardano/$\mathbb{Z}/3$ form. The counterexample carries a hidden variable satisfying a cubic; it is a cubic covering.*

An earlier run of this fiber solve was not persisted; rather than cite a hash that will not resolve, the loop *re-ran* the computation as a self-contained exact check—the lex Gröbner basis in shape position over $\mathbb{Q}$ (so each z-root lifts to exactly one (x, y)), the stated cubic and its nonzero discriminant, and the Cardano resolvent identity $R^2 + qR - (p/3)^3 = 0$ with $p/3 = \frac{45039}{16}$—and signed it: ✓ étale + degree-3 Cardano fiber [b8d5376ad450aa97]. (An earlier naive attempt to read the fiber degree off iterated resultants returned "21"; the loop caught the artifact and the Gröbner solve gave the true answer, three—a small instance of the adjudicator correcting the proposer.)

6 Episode 4 — building new ones

The human asked the loop to *construct*, not just analyze. The model observed a fixed point and proposed iteration.

Claim 4 (infinite tower). *$(0, 0, -\frac{1}{4})$ is a fixed point of G and the common image of the colliding points, so $\det \mathrm{J}(G^n) \equiv 1$ and the collision persists: G^n is a counterexample of degree 3^n. Once one exists, infinitely many of unbounded degree do.*

✓ iterates are counterexamples [0bd2b1a1ac595eb9]. We are explicit that these are *iterates*—new maps that are genuinely counterexamples, but derived, not an independent new one. That frontier stays open, and the loop's later attempts to reach it honestly failed (§9).

7 Episode 5 — mapping the space, and checking the literature

Here the loop shifted from the object to its neighborhood. Alpöge's map is linear in z, $F = A(x, y)z + B(x, y)$; the model decomposed the Keller condition and Attestral certified each rung.

Lemma 5 (decomposition). *$\det \mathrm{J}F$ is a quadratic in z, constant iff* (C1) $\det[A_x, A_y, A] \equiv 0$; (C2) *a linear condition on the compensator B;* (C3) $\det[B_x, B_y, A] \equiv \text{const}$.

Transport (and an honest deflation). (C1) says the transport surface is a cone; ✓ cone completeness [8b9ffd621fabff3f]. The reason is an explicit identity: the transport family $A = x^n g(1/x + y)$ is a linear combination of its own partials,

$$A \;=\; \frac{x}{n}\,A_x + \frac{1}{nx}\,A_y \qquad \text{(every } n\text{, every directrix } g\text{)},$$

so $\det[A_x, A_y, A] \equiv 0$ is forced—a proof, not a check (a85c74a7). Behind it sits one grading operator $D = x\partial_x + \frac{1}{x}\partial_y$ with $DA = nA$, which collapses the compensator condition (C2) (6ce58634); solving (C2) in closed form, the entire compensator space is $B = a\,g(w) + b\,g'(w) + c(w)$ (with a a pure gauge), and the whole classification reduces to *one scalar equation*

$$C_3 \;=\; x^{\,n-1}\,(Db)\,\big(b\,W + \det[g, g', c']\big) \;=\; \text{const}, \qquad W = \det[g, g', g'']$$

(8b7b743a)—the equation the fold-parity and uniqueness results below interrogate. When the loop checked the literature, the "transport ⇔ cones" step turned out to be elementary classical developable-surface geometry—so we do *not* headline it as novel. The check-the-literature step is part of the loop, not an afterthought.

Obstructions. A homogeneous transport forces a zero constant term, so a counterexample needs a degree-0 anchor (Alpöge's 1): ✓ affine anchor [27643d18f88937c4]—which, on literature contact, *partially overlaps* Shaska's graded lemmas [10]; we credit it and claim only the z-linear reading. A z-linear counterexample needs *odd* fold degree (a discriminant forced to be a perfect square): ✓ fold-parity [d44b465472b3065c], refined by 6c645442, z-independent ac229d5a. This is why Alpöge's is cubic—the smallest odd fold.

Rigidity, and the loop auditing itself. A weight argument collapses the family to a rigid branch: ✓ uniqueness skeleton [680531790afdcb10].

Remark 6 (we corrected our own overclaim). An earlier draft asserted Alpöge's map is the *unique* z-linear counterexample. Re-examining our own certificates, the fold-parity and Wronskian results give only *necessary* conditions, and low-degree directrices meet them; odd folds $n \geq 5$ are *open*. The loop is built to audit its own prior claims, and here it did.

The z-quadratic completion shares the transport backbone and is obstructed through compensator degree 5 (2f7205c9, f03de165). In $\mathbb{C}^4$ the transport may be a developable hypersurface with a two-parameter directrix (c2cad3bd)—a strictly larger family than the $\mathbb{C}^3$ cone, and the natural place a different counterexample might live. We settled its most natural representative. For the polynomial-directrix anchor $A = x \cdot g(1 + xy, 1 + yw)$, the Keller closure factors as $C_4 = \det[B_x, B_y, B_w, x\,g] = x \cdot \det[B_x, B_y, B_w, g]$: divisible by x for *any* compensator B, hence never a nonzero constant.

Claim 7 ($\mathbb{C}^4$ obstruction, all degrees)**.** *The developable anchor $A = x \cdot g$ (homogeneous $\rho = x$, polynomial directrix) admits no z-linear Keller map at* any *compensator degree—the $\mathbb{C}^4$ analog of affine-anchor necessity.*

✓ Lean-kernel: cB_C4_never_nonzero_const [e8daa77e222d7cb2] (exact-symbolic corroboration 9c8fad22). A finite-field Gröbner search had agreed through degree 3 across two primes, but the degree count settles *all* degrees in one line—the pruning proof the methodology prefers to a larger search. What remains open is exactly the complementary family: a $\mathbb{C}^4$ counterexample would need a *degree-0 or pole anchor* (the four-dimensional analog of Alpöge's $1 + xy$ over x), not a polynomial directrix with homogeneous ρ.

The open frontier, stated explicitly. Where a genuinely new counterexample could still hide, as the certificates leave it: **(i)** *odd folds $n \geq 5$, z-linear*—the necessary conditions (n odd, W a perfect square) are met by the natural first candidate $g = (w^5, 5w^4, -1)$, but its polynomiality closure fails within the searched range of the skeleton certificate; existence at $n \geq 5$ is open. **(ii)** *Dropping the $\mathbb{Z}/2$ symmetry*—the equivariant sector is where our rigidity evidence lives; the non-symmetric z-linear sector is unconstrained by it. **(iii)** *Beyond z-linear*—the transport theory is z-linear-specific (the z-quadratic backbone is forced onto the same cones, but higher structure is open). **(iv)** *The $\mathbb{C}^4$ pole anchor* above. **(v)** *The plane, $\mathbb{C}^2$*, on which we claim nothing. An explicit frontier is the honest complement of a classification: it says exactly what the certificates do *not* close.

8 Episode 6 — the loop catching our own errors

The most important evidence that the method works is not a success but the failures it stopped.

The BCW self-error. The proposer built a degree-reduction to construct a nilpotent counterexample. × refuted: its own Jacobian check showed det non-constant off a measure-zero section. It was discarded, not published. A second, subtler reconstruction ("replace a factor g by a new variable w") passed a toy but × refuted: det was non-constant on Alpöge—constant only on the surface $w = g$. Again caught, again discarded. Both are exactly the kind of plausible-looking construction that ships in an unverified workflow.

The $p = 3$ prime collapse. Mapping the $\mathbb{C}^4$ frontier (§7), a command-line argument aliased the working prime onto the compensator-degree flag, silently running the entire Gröbner closure over GF(3) instead of GF(32003)—and it returned a clean-looking "obstructed" verdict anyway. × refuted: verify-the-verifier caught it: the nullspace rank differed from the two-prime re-run, exposing the bad field. The verdict was discarded and recomputed at proper primes before any claim. A poisoned solver that still prints a plausible answer is exactly the failure a signing verifier exists to stop.

In a workflow where the generator signs its own work, all three errors ship. Here none did, because the agent that signs is not the agent that proposes—and because the discipline checks its own tooling before trusting its verdicts.

9 Episode 7 — the honest wall: the nilpotent normal form

The fullest test of the method was an attempt that did *not* succeed, and reporting it is the point. By Bass–Connell–Wright, the cubic-homogeneous *nilpotent* form (✓ $\det(I + \mathrm{J}\,H) = 1 \Leftrightarrow$ nilpotent [2ac04a4d5fe8aedf]) is the exact shape the field tried for decades to prove invertible; Zhang [9] asks to *see a small explicit one*. The loop ran the full method at it:

1. **Verify the verifier first.** We certified the *checker* that judges candidates (H cubic-homogeneous, $\mathrm{J}\,H$ nilpotent, non-injective): ✓ sound on knowns [04acac9165f3201a] — it accepts a triangular nilpotent cubic as a valid form yet flags it injective, rejects a non-nilpotent decoy, and detects Alpöge's collision.
2. **Direct construction, refuted.** Screening candidates by fiber size, every Drużkowski nilpotent map through dimension 4 is invertible, and the $\mathbb{Z}/3$-equivariant cubic-homogeneous family on $\mathbb{C}^3$ collapses to triangular (invertible) maps across all 33 nilpotency branches. × refuted: low dimension

forces injectivity via a recoverable-invariant filtration. The counterexample needs high dimension—reproducing the classical low-dimensional verification of the conjecture.

3. **Sourced the real reduction; verified the step.** The human pushed to source the correct construction. Campbell's form [5]—remove a product $a \cdot b$ by $(f_1, \dots) \mapsto (f_1 - (y+a)(z+b), \dots, y+a, z+b)$—has a Schur complement collapsing *exactly* to JF: ✓ det preserved (-2) on Alpöge through a step `[verified]`. The literature's "nonconstructive" label is beaten *for a single step*.
4. **The wall.** Naive automated execution explodes past 200 auxiliary variables before terminating. A *small* explicit nilpotent counterexample is precisely the optimization Zhang flags as open; the loop quantified the difficulty rather than resolving it.

We state the outcome plainly: existence is guaranteed, the per-step construction is verified, and a usable small explicit map is beyond what we obtained. An honest wall, reported with its receipts, is a result of the method—not a gap to hide.

10 Episode 8 — the same loop, days later, on a second conjecture

A case study of one object cannot show that the loop, rather than the object, is what matters. The same week supplied the test. Prompted by Alpöge's announcement—via the theorem of Derksen, van den Essen, and Zhao connecting Keller maps to Gaussian moments [11]—Long posted explicit *small counterexamples to the Gaussian Moments Conjecture* [12]. The identical apparatus, unchanged, ran on it: for independent standard real Gaussians x_1, x_2, t and $Z = (x_1 + ix_2)/\sqrt{2}$, $W = (x_1 - ix_2)/\sqrt{2}$, the polynomial $P_3 = (1+Z)\big(W - \frac{1}{2}(2+Z)t^2\big)$ has every moment zero, $\mathbb{E}[P_3^m] = 0$, yet $\mathbb{E}[Z\,P_3^m] = m! \neq 0$—hypothesis satisfied, conclusion broken. Every expectation reduces to exact rational arithmetic by Wick's theorem ($\mathbb{E}[N(0,1)^k] = (k-1)!!$ for even k, else 0); no sampling. ✓ exact Wick identities, $m \leq 4$ `[d804dce81c34c732]`. We mark the ceiling: our check is exact through $m = 4$; the all-m statement is proved in [12], not by our certificate. A different mathematical domain—probabilistic moment identities rather than polynomial algebra—adjudicated same-day by the same loop is the beginning of an answer to "does this generalize?", and the honest form of that answer: one further domain, exactly checked, ceiling stated.

11 The ledger: what the loop produced, and how to re-run it

The workflow's output is not prose to be believed but a set of receipts to be re-run. Two things must not be conflated, so the ledger separates them explicitly. **Every certificate is ed25519-signed—being "signed" is not the distinction. The distinction is the rung.** The bundle holds **43 certificates**:

- **20 Lean-kernel formal proofs** (tier `proven`): the Lean 4 kernel, with Mathlib, accepted the theorem—axiom-audited, no `sorry`. Every one is enumerated in Table 2.
- **23 exact-symbolic computations** (tier `not_refuted`): exact sympy/Gröbner identities over $\mathbb{Q}$ or GF(p)—decisive for the decidable statements they check, but *not* proof-assistant proofs. These carry the episode narrative (Table 1) and include the Gaussian-Moments companion (§10).

Most headline claims exist at *both* rungs: the exploration ran on the fast exact-symbolic ladder, and the load-bearing statement was then re-proved in the kernel. Table 1 maps each episode to

its exact-symbolic certificate and points into Table 2 where a kernel twin exists. One command re-checks everything (content hash, ed25519 signature, checker verdict):

```
# from the published artifact bundle (Zenodo / simgen.dev) -- needs Python + pynacl only
python verify_all.py # all 43, grouped by rung
python verify_receipt.py certs/377ceaee812182f4.json
```

Episode / claim	Rung	Certificate
1. object is a counterexample	exact + kernel twin (T2.A1–A2)	`377ceaee812182f4`
2. mechanism $F = L \circ G$, non-nilpotent	exact + kernel twin (T2.A3, A9)	`403eb3ed0ef74fab`
2'. symmetry + triple fiber	exact	`4aaf7bd86638824b`
3. hidden cubic / degree-3 étale (Cardano fiber)	exact	`b8d5376ad450aa97`
4. infinite tower G^n	exact + kernel twin (T2.A4)	`0bd2b1a1ac595eb9`
5. transport / cones (elementary)	exact + kernel twin (T2.A5–A6)	`8b9ffd62`, `a85c74a7`, `6ce58634`, `8b7b743a`
5. affine anchor (overlap: Shaska)	exact	`27643d18f88937c4`
5. fold-parity / Wronskian / z-indep.	exact	`d44b4654`, `6c645442`, `ac229d5a`
5. uniqueness skeleton	exact	`680531790afdcb10`
5. z-quadratic completion	exact	`2f7205c9`, `f03de165`
5. $\mathbb{C}^4$ obstruction, all degrees	**kernel** (T2.D) + exact corrob.	`e8daa77e`, `8d48e301`; `9c8fad22`
6. three self-errors (incl. p=3)	refuted & discarded	(no cert, by design)
7. nilpotent verifier	exact	`04acac9165f3201a`
7. canonical form $\Leftrightarrow$ nilpotent	exact + **kernel** (T2.B)	`2ac04a4d5fe8aedf`
8. Gaussian Moments (Long), $m \leq 4$	exact (companion)	`d804dce81c34c732`

Table 1: The episodes and their headline certificates. "exact" = exact-symbolic (`not_refuted`); "kernel" = Lean-kernel formal proof (`proven`); "kernel twin" points to the corresponding row of Table 2.

	Lean theorem	Statement	Cert
A. Alpöge's map—complete anatomy (9)			
A1	`alpoge_jacobian_det`	$\det \mathrm{J}F \equiv -2$ (constant, nonzero)	`763cdfa1`
A2	`alpoge_triple_collision`	three distinct points $\to (-\frac{1}{4}, 0, 0)$	`0252cdf7`
A3	`alpoge_essential_det_one`	$\det \mathrm{J}\,G \equiv 1$ (essential map étale)	`99bc534c`
A4	`alpoge_tower_fixed_point`	$G(0,0,-\frac{1}{4}) = (0,0,-\frac{1}{4})$ (tower engine)	`18fcad33`
A5	`alpoge_transport_C1`	$\det[A_x, A_y, A] \equiv 0$ (transport/cone)	`34530383`
A6	`alpoge_transport_grading`	$3xA = x^2 A_x + A_y$ ($DA = 3A$, graded cone)	`876c5bee`
A7	`alpoge_keller_C2`	$C_2 \equiv 0$ (z^1 coefficient)	`8b03f68b`
A8	`alpoge_keller_C3`	$C_3 \equiv -2$ (z^0 coefficient)	`315fe679`
A9	`alpoge_JH_nonnilpotent`	$\mathrm{tr}(\mathrm{J}\,H) \neq 0$ (non-nilpotent)	`c973a028`
B. Canonical-form / nilpotency machinery (7)			
B1	`det_one_add_nilpotent`	nilpotent $f \Rightarrow \det(1+f) = 1$ (any dim., any field)	`c6771ba5`
B2	`isNilpotent_of_charpoly_eq_X_pow`	charpoly $= X^n \Rightarrow$ nilpotent (Cayley–Hamilton)	`b8981a6b`
B3	`det_one_add_nilpotent_3x3`	3×3 canonical form, arbitrary entries	`7ff14e16`
B4	`unitriangular_det_one` (`+_nilpotent`)	strictly upper-triangular: $\det(1+N) = 1$, $N^3 = 0$	`83938afa`
B5	`homogeneous_pos_eq_C_imp_zero`	homogeneous, positive degree, = const $\Rightarrow 0$	`86c3c243`
B6	`eq_zero_of_add_eq_zero_of_isHomogeneous`	distinct-degree homogeneous sum $0 \Rightarrow$ each 0	`5b14c4f9`
B7	`three_homogeneous_sum_eq_zero`	$e_1 + e_2 + e_3 = 0$ (deg 2,4,6) $\Rightarrow$ all 0	`5770b618`
C. The converse insight (1)			
C1	`alpoge_converse_fails`	concrete M: $\det(I+M) = 1$ yet $\mathrm{tr}\,M \neq 0$	`a83644b6`
D. The $\mathbb{C}^4$ obstruction (3)			
D1	`c4_cB_developable`	the $\mathbb{C}^4$ developable transport family	`e528092b`
D2	`det_updateCol_smul_dvd`	column-divisibility determinant lemma	`8d48e301`
D3	`cB_C4_never_nonzero_const`	C_4 divisible by x at *every* degree	`e8daa77e`

Table 2: All 20 Lean-kernel formal proofs (tier `proven`; Mathlib, axiom-audited, no `sorry`). A1+A2 together kernel-prove that the map is a Jacobian counterexample over $\mathbb{Q}$; B gives the full equivalence nilpotent $\Leftrightarrow$ charpoly $= X^n$; D settles the $\mathbb{C}^4$ candidate at all degrees.

All 43 certificates live in one folder, ed25519-signed under public key `8e803000c50431bb…3699f38`, and `verify_all.py` re-verifies them grouped by rung so the two tiers stay distinct. Table 3 lists the machine checks used.

Check	Role
mypy, Ruff	static typing and lint gate on every checked artifact
pytest, Hypothesis	exact identity assertions and property-based tests
CrossHair	symbolic / concolic execution of the checked predicates
SymPy; GF(p) FFT; Macaulay2	exact polynomial algebra; finite-field determinant searches; Gröbner obstruction screens (solver-scouted, bounded)
Lean 4 + Mathlib (kernel, axiom-audited)	**formal proofs** of the load-bearing theorems: the nilpotency canonical form and the $\mathbb{C}^4$ obstruction
SHA-256 + ed25519; `verify_receipt.py`	signing and offline re-verification (hash, signature, checker re-run)

Table 3: The checks used. The Lean-kernel rung supplied the formal proofs of the load-bearing theory; the remaining formal rungs (Kani, CBMC, Dafny, F*, TLC) are part of Attestral but were not required for this work and are not claimed.

12 What the method is good at, and what it is not

Reporting the shape of the tool honestly is part of the result.

- **Strong at:** independent verification at the speed of generation; reverse-engineering structure; constructing derived objects (the tower); mapping a construction space; and—decisively— catching the proposer's own errors, because proposer and adjudicator are separate agents.
- **Weak at:** inventing genuinely novel theorems or beating famous searches; and producing *small* explicit constructions where the bottleneck is optimization, not verification (§9). Attestral is a verifier and a guide, not a better searcher.
- **Honest by construction:** two tiers never blurred—Lean-kernel *proved* for the load-bearing theorems, exact-symbolic *machine-checked* for the exploratory identities (§2); overlaps credited (§7); unsigned exact-computation results labeled and, where load-bearing, re-run and signed (the degree-3 fiber, §5); solver-scouted bounds marked and, when a degree count is available, replaced by an all-degree proof (§7); the tooling itself verified before its verdicts are trusted (the p=3 catch, §8); the literature checked before novelty is claimed; and walls reported with receipts (§9).

13 Conclusion

An AI helped find a counterexample to an 87-year-old conjecture, and the world asked whether it could be trusted. We answered by running a different loop—a human directing Claude Code (Opus 4.8), every claim adjudicated and signed by a verifier that has no stake in the model being right—and by showing what it produced: an independently verified object, its mechanism, its hidden cubic, an infinite tower, a certified map of its construction space, a four-dimensional frontier proved obstructed at every degree in the Lean kernel, three of our own errors caught and thrown away, and an honest wall—and, days later, the same loop run unchanged on a second falling conjecture. The theorems are the evidence—20 of them formal Lean-kernel proofs, the rest exact-symbolic, the two tiers never blurred. The method—proof-carrying mathematics, generated by AI and adjudicated by machine, where you re-run the check instead of trusting the author—is the contribution.

Artifacts. The complete artifact bundle—all 43 signed certificates, the Lean 4 sources, the published verification key, and the standalone offline verifier—is archived at Zenodo (DOI: 10.5281/zenodo.21567375) and mirrored with an interactive companion at https://simgen.dev/attestral/jacobian-counterexample/. Re-verify everything with `python verify_all.py` (Python + `pynacl` only), or any single certificate with `python verify_receipt.py certs/<hash>.json`. https://simgen.dev

References

[1] O.-H. Keller. *Ganze Cremona-Transformationen.* Monatsh. Math. Phys. 47, 1939.

[2] H. Bass, E. H. Connell, D. Wright. *The Jacobian conjecture: reduction of degree and formal expansion of the inverse.* Bull. AMS 7(2), 1982.

[3] L. M. Drużkowski. *An effective approach to Keller's Jacobian conjecture.* Math. Ann. 264, 1983.

[4] A. van den Essen. *Polynomial Automorphisms and the Jacobian Conjecture.* Birkhäuser, 2000.

[5] L. A. Campbell. *Reduction theorems for the Strong Real Jacobian Conjecture.* arXiv:1303.3853, 2013.

[6] S. Smale. *Mathematical Problems for the Next Century.* Math. Intelligencer 20, 1998.

[7] L. Alpöge. *A counterexample to the Jacobian Conjecture in dimension three.* Announcement, 19–20 July 2026.

[8] T. Tao. *A digestion of the Jacobian conjecture counterexample.* Blog, 21 July 2026.

[9] Z. Zhang. *Direct Consequences of the Three-Dimensional Counterexample.* 2026.

[10] T. Shaska. *Graded Keller maps and the Jacobian Conjecture.* arXiv:2607.20210, 2026.

[11] H. Derksen, A. van den Essen, W. Zhao. *The Gaussian Moments Conjecture and the Jacobian Conjecture.* Israel J. Math. 219(2), 917–928, 2017.

[12] C. D. Long. *Small Counterexamples to the Gaussian Moments Conjecture.* arXiv:2607.18186, 2026.

[13] *The new counterexample to the Jacobian conjecture.* Secret Blogging Seminar, 20 July 2026.