Semi-Autonomous Formalization of the Vlasov–Maxwell–Landau Equilibrium

Vasily Ilin [1]

Abstract

We present a complete LEAN 4 formalization of the equilibrium characterization in the Vlasov-Maxwell-Landau (VML) system, which describes the motion of charged plasma. The project demonstrates the full AI-assisted mathematical research loop: an AI reasoning model (Gemini DeepThink) generated the proof from a conjecture, an agentic coding tool (Claude Code) translated it into LEAN from natural-language prompts, a specialized prover (Aristotle) closed 111 lemmas, and the LEAN kernel verified the result. A single mathematician supervised the process over 10 days at a subscription cost of $325, writing zero lines of code.

The entire development process is public: all 229 human prompts, and 213 git commits are archived in the repository. We report detailed lessons on AI failure modes – hypothesis creep, definition-alignment bugs, agent avoidance behaviors – and on what worked: the abstract/concrete proof split, adversarial self-review, and the critical role of human review of key definitions and theorem statements. Notably, the formalization was completed before the final draft of the corresponding math paper was finished.

1. Introduction

Formal verification of mathematics in proof assistants like LEAN 4 (de Moura & Ullrich, 2021; The mathlib Community, 2020) has traditionally required deep expertise in both the mathematics and the proof assistant itself. The learning curve is steep: even experienced mathematicians report months of effort to formalize results that take pages to prove on paper (Urban, 2026). This bottleneck has limited formal verification to a small community of specialists.

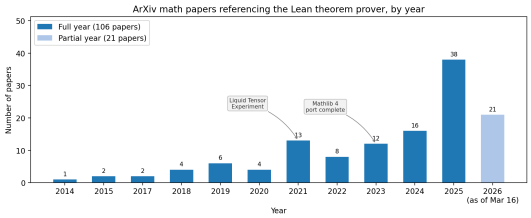

Figure 1. ArXiv papers using LEAN for formalization, by year. Data from the arXiv metadata snapshot (March 2026), filtered for references to the LEAN theorem prover.

Recent advances in AI-assisted theorem proving are changing this landscape rapidly (Figure 1). Large language models can now generate LEAN tactics (Chen et al., 2025), agentic coding tools can manage entire LEAN projects (Liu et al., 2026), and cloud-based automated theorem provers can close individual lemmas without human intervention (Achim et al., 2025).

Yet most published work on AI-assisted formalization focuses on competition mathematics (IMO, Putnam) or textbook results. Recent breakthroughs include AxiomProver autonomously generating a formal LEAN proof of Fel's open conjecture (Chen et al., 2026c) from a natural-language problem statement alone, GPT-5.2 generating informal proofs of open Erdős problems that were then autoformalized in LEAN by Aristotle (Sothanaphan, 2026), and Google DeepMind's Aletheia agent (Feng et al., 2026b;a) autonomously producing research papers in natural language. Liu et al. (2026) formalized effective Brascamp-Lieb inequalities (∼8K lines) with human experts collaborating on the LEAN code, and Bayer et al. (2025) formalized their own new theorem in Isabelle in parallel with the mathematical research, writing the proof assistant code themselves.

Our work is distinguished not by full autonomy – AxiomProver's Fel's conjecture result is more autonomous – but by the combination of several features that, to our knowledge, no single prior project shares (Table 1):

Code, prompts, and full development logs: https://github.com/Vilin97/aristotle/tree/landau.

[1]University of Washington, Seattle, WA, USA. Correspondence to: Vasily Ilin <vilin@uw.edu>.

3rd AI for Math Workshop, co-located with the 43rd International Conference on Machine Learning (ICML 2026), Seoul, South Korea. Copyright 2026 by the author(s).

Project	Prover	LOC	New thm	New defs	Team	AI role	Logs	Domain
Gauss (Math, Inc., 2026)	LEAN 4	181K[‡]	✗	✗	6+[§]	sorry elim.	✗	Geometry
Bayer & David (Bayer et al., 2025)	Isabelle	20K	✓	✓	15	none	✗	Num. theory
This work	LEAN 4	**10K**	✓	✓	**1**	**all code**	✓	**Math. phys.**
Numina (Liu et al., 2026)	LEAN 4	∼8K	✓	✓	13	collab.	✗	Analysis
AxiomProver (Chen et al., 2026b)	LEAN 4	4.3K	✓	∼	21	fully auton.	✗	Num. theory
AxiomProver (Chen et al., 2026c)	LEAN 4	2.7K	✓	∼	21	fully auton.	✗	Algebra
2HDM (Tooby-Smith, 2026)	LEAN 4	1.6K	✗[†]	✓	1	minimal	✗	Physics
Aristotle (Sothanaphan, 2026)	LEAN 4	1.4K	✓	✗	1	fully auton.	✗	Combin.
AxiomProver (Chen et al., 2026a)	LEAN 4	0.2K	✓	∼	5	fully auton.	✗	Alg. geom.

Table 1. Recent research-level formalizations with publicly available code, sorted by LOC. "New thm" = theorem previously unpublished. "New defs" = required definitions or theory not in the proof library (✓ = substantial, ∼ = lightweight, ✗ = none). "AI role": sorry elim. = eliminated sorry's in human-written scaffold; all code = AI generated all proof-assistant code from NL prompts; collab. = human-AI pair programming; fully auton. = AI from NL problem statement alone. "Logs" = full development process public (all commits, prompts, and interaction logs). [†]Discovered an error in the published proof. [‡]∼22K lines (definitions, statements) written by human experts; Gauss eliminated ∼160K lines of sorry's in 3 weeks. [§]No paper; 6+ named contributors on blog post.

- **Scale and depth.** 34 files and 10K+ lines of LEAN, in hard analysis: Coulomb singularity cancellation, Leibniz integral rule, energy methods on the torus.
- **New theory.** The theorem cannot even be *stated* with existing MATHLIB – the Landau operator, torus differential structure, and VML system had to be defined from scratch.
- **New result.** The characterization of smooth steady states of the full VML system with Coulomb collisions and electromagnetic coupling does not appear in the prior literature in this precise form and is, to our knowledge, new as a stated and proved result.
- **Fully documented process.** A single mathematician wrote zero code while actively steering the formalization over 10 days; every prompt, commit, and tool call is public.

The contributions are:

1. A demonstration of the full AI-assisted mathematical research loop: from an open conjecture, through AI-generated resolution and detailed proof (Gemini DeepThink, requiring no mathematical corrections), to a complete machine-verified formalization (Claude Code + Aristotle) – all with the human acting solely as supervisor. To our knowledge, this is the first project to close this entire pipeline on a research-level result in hard analysis.
2. A complete LEAN 4 formalization of a new theorem in mathematical physics (the VML steady-state characterization), verified with zero `sorry`'s.
3. A detailed, fully transparent account of the process: every prompt, every commit, every tool call is public.
4. Lessons learned about the capabilities and failure modes of current AI tools for formalization, including a definition-alignment bug caught by an expert reviewer post-completion.
5. Evidence that research-level formalization is now accessible to mathematicians without deep LEAN expertise, at minimal cost ($200 Claude subscription + $125 Gemini subscription + $0 for Aristotle).

2. The Theorem

2.1. The Physical System

The Vlasov-Maxwell-Landau (VML) system is a system of PDEs modeling the dynamics of a charged plasma. The full time-dependent system evolves a particle distribution function $f(t, x, v)$, an electric field $E(t, x)$, and a magnetic field $B(t, x)$ via the Vlasov equation coupled to Maxwell's equations with the Landau collision operator.

We are interested in *steady states*: solutions with no time dependence. Setting all time derivatives to zero, the unknowns reduce to $f(x, v) > 0$ on $\mathbb{T}^3 \times \mathbb{R}^3$, $E(x)$, and $B(x)$, and the system becomes:

$$v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = \nu\, Q(f, f), \tag{1}$$

$$\nabla \times B = J = \int_{\mathbb{R}^3} v\, f\, dv, \tag{2}$$

$$\nabla \cdot E = \int_{\mathbb{R}^3} f\, dv - \rho_{\text{ion}}, \tag{3}$$

$$\nabla \cdot B = 0, \tag{4}$$

where $Q(f, f)$ is the Landau collision operator with the Coulomb kernel $\Psi(r) = r^{-3}$. Note that Faraday's law $\partial_t B = -\nabla \times E$ becomes $\nabla \times E = 0$ at equilibrium; this is not listed as a hypothesis because it is derived in the proof (it follows from the other equations).

2.2. Main Result

Theorem 1 (Global steady state of the VML system)**.** *Consider the Vlasov–Maxwell–Landau system with collision*

frequency $\nu > 0$ on the periodic spatial domain $x \in \mathbb{T}^3$, velocity space $v \in \mathbb{R}^3$, and uniform neutralizing background density $\rho_{\text{ion}} > 0$. Let $f(x, v) > 0$ be a sufficiently smooth steady-state solution with finite total energy and finite entropy, and assume that f and its derivatives decay sufficiently fast as $|v| \to \infty$. Then the distribution is a spatially uniform, zero-drift Maxwellian,

$$f_\infty(v) = \frac{\rho_{\text{ion}}}{(2\pi T_\infty)^{3/2}} \exp\left(-\frac{|v|^2}{2T_\infty}\right),$$
$$E_\infty = 0,$$
$$B_\infty = \frac{1}{|\mathbb{T}^3|} \int_{\mathbb{T}^3} B_{\text{init}}(x)\, dx.$$

Here B_∞ is the conserved spatial mean of the initial magnetic field, and

$$T_\infty = \frac{2}{3\rho_{\text{ion}}|\mathbb{T}^3|}\left(\mathcal{E}_0 - \frac{|B_\infty|^2}{2}|\mathbb{T}^3|\right) > 0,$$

where

$$\mathcal{E}_0 = \iint \frac{1}{2}|v|^2 f_{\text{init}}\, dv\, dx + \frac{1}{2}\int \left(|E_{\text{init}}|^2 + |B_{\text{init}}|^2\right) dx$$

is the initial total energy.

The formal LEAN statement is shown in Figure 2. It formalizes the steady-state uniqueness part of the theorem: for any fields satisfying the stationary VML equations and the stated regularity and decay assumptions, there exist $T_\infty > 0$ and a constant vector B_∞ such that the Maxwellian formula in the theorem holds. The concrete statement uses 12 hypothesis families: 2 physical parameters ($\nu > 0$, $\rho_{\text{ion}} > 0$), strict positivity of f, finite smoothness (C^3 in velocity, C^2 in space, C^2 for B), uniform Schwartz decay, a polynomial score bound, and the four steady-state equations (1)–(4). The polynomial log-growth bound needed in the entropy argument is derived internally from the score bound and smoothness.

2.3. Proof Architecture

The proof follows a classical entropy method in seven steps (Figure 3):

1. **H-theorem:** Entropy dissipation $D(f) \leq 0$; the Landau matrix is positive semi-definite.
2. **Nullspace:** $D(f) = 0$ implies f is a local Maxwellian: $\log f(x, v) = a(x) + b(x) \cdot v + c(x)|v|^2$.
3. **Transport:** Substituting into the Vlasov equation yields a polynomial in v that must vanish identically.
4. **Polynomial matching:** The $O(|v|^3)$ terms force $\nabla c = 0$ (constant temperature); the $O(|v|^2)$ terms yield Killing's equation for b.

```
theorem CoulombConcreteTheorem42
    (f : Torus3 → (Fin 3 → ℝ) → ℝ)
    (E B : Torus3 → Fin 3 → ℝ)
    (ν ρ_ion : ℝ)
    (hν : 0 < ν)
    (hρ_ion : 0 < ρ_ion)
    (hf_pos : ∀ x v, 0 < f x v)
    (hf_smooth_v : ∀ x, ContDiff ℝ 3 (f x))
    (hf_smooth_x : ∀ v, ContDiff ℝ 2
      (periodicLift (fun x => f x v)))
    (hB_smooth : ∀ i, ContDiff ℝ 2
      (periodicLift (fun x => B x i)))
    (hSchwartz : UniformSchwartzDecay f)
    (hGradBound : ∃ (Cg : ℝ) (Kg : ℕ),
      ∀ (x : Torus3) (v : Fin 3 → ℝ) (i : Fin 3),
      |fderiv ℝ (f x) v (Pi.single i 1)|
        ≤ Cg * (1 + ‖v‖) ^ Kg * f x v)
    (hVlasov : ∀ x v,
      dotProduct v (torusGradX (fun y => f y v) x) +
      dotProduct (E x + cross v (B x)) (vGrad (f x) v)
      =
      ν * LandauOperator coulombKernel (f x) v)
    (hAmpere : ∀ x,
      torusCurlX B x = fun i => ∫ v, v i * f x v)
    (hGauss : ∀ x,
      torusDivX E x = (∫ v, f x v) - ρ_ion)
    (hDivB : ∀ x, torusDivX B x = 0)
    :
    ∃ (T_eq : ℝ) (B₀ : Fin 3 → ℝ), 0 < T_eq ∧
    (∀ x v, f x v = equilibriumMaxwellian ρ_ion T_eq v
     ) ∧
    (∀ x, E x = 0) ∧
    (∀ x, B x = B₀)
```

Figure 2. The formal LEAN 4 statement of the main theorem (`CoulombConcreteTheorem42`). The proof (omitted) is 80 lines of tactic invocations that assemble the seven-step argument described in Section 2.3. Here `Torus3` is `Fin 3 → AddCircle 1`, `periodicLift` lifts torus functions to $\mathbb{R}^3$ for differentiation, and `equilibriumMaxwellian` is the Maxwellian $\rho_{\text{ion}}/(2\pi T)^{3/2} \exp(-|v|^2/2T)$.

5. **Killing's equation:** On the flat torus, the only smooth Killing fields are constants, so $b = b_0$.
6. **Maximum principle:** Ampère's law forces $b_0 = 0$; Gauss's law forces $E = 0$ and uniform density.
7. **Harmonic analysis:** With $J = 0$, curl $B = 0$ and div $B = 0$ imply B is harmonic on $\mathbb{T}^3$, hence constant.

3. Tools

The project used four AI systems, none of which required any LEAN code to be written by hand.

Claude Code (Anthropic). An agentic coding tool that operates in the terminal, with access to file reading/editing, shell commands, and MCP (Model Context Protocol) integrations. Claude Code was the primary workhorse: it generated all LEAN code, managed the project structure, ran diagnostics via the LEAN LSP, and orchestrated submissions to Aristotle. The subscription costs \$200/month (Claude Max plan).

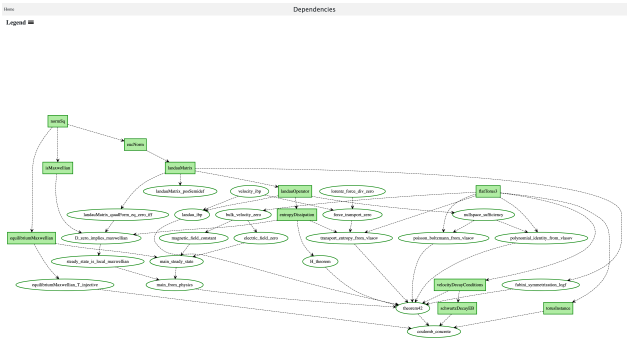

Figure 3. Proof dependency graph, generated with LeanBlueprint (Massot, 2020). All nodes are green (fully proved). The graph flows from primitive definitions (`normSq`, `eucNorm`, `landauMatrix`) at the top through intermediate results (`H_theorem`, `D_zero_implies_maxwellian`, the field and conservation lemmas) down to `theorem42` and its concrete Coulomb instantiation `coulomb_concrete` at the bottom. An interactive version is available in the repository.

Gemini DeepThink (Google DeepMind). A mathematical reasoning model accessed via MCP. Gemini generated the initial natural-language proof of the theorem in a five-turn dialogue, identified the relevant literature, and was consulted during the project for planning decisions (e.g., choosing the torus representation). The subscription cost was $125 for the project period.

Aristotle (Harmonic). A cloud-based automated theorem prover for LEAN 4 (Achim et al., 2025). Lemma statements were extracted from the project, submitted via a Python script, and results were automatically integrated. Aristotle proved 111 lemmas, disproved 28 false conjectures, and returned 66 with `sorry`'s still present. The service was free during the project period.

Lean LSP tools. The LEAN Language Server Protocol, accessed via MCP, provided real-time compilation feedback, goal-state inspection, tactic suggestions, and library search (LeanSearch, Loogle). These tools generated 3,194 calls over the project.

4. Process

4.1. Timeline

The project unfolded over 10 days of active development (March 1 – 10, 2026), preceded by the Gemini DeepThink session on February 28. Table 2 summarizes the key milestones.

4.2. Phase 1: Natural-Language Proof (Feb 28)

The project began with a question posed to Gemini DeepThink:

> *"What is known about the equilibrium of the Vlasov-Maxwell-Landau equations with Coulomb collisions, and a single species? Does it have to be a global Maxwellian? With what covariance? How is it shown?"*

Gemini produced a detailed six-step proof, identified the relevant literature (Villani, 2002; Cercignani et al., 1994; Guo, 2003; 2012; Desvillettes & Villani, 2005), and refined the argument over five turns. Crucially, the proof required no mathematical corrections: the mathematician verified the argument by hand and found it complete and correct, needing only the identification of exact supporting theorems from the cited papers. The final natural-language proof became the blueprint for the formalization.

4.3. Phase 2: Scaffolding (Mar 1 – 2)

The formalization prompt was:

> *"Look at H-theorem-formal.tex. I want to formalize that the steady state of the VML."*

Claude Code generated a monolithic 1,000+ line LEAN file within hours, with `sorry`'s marking every gap. Claude estimated the full formalization would require "multiple months." The mathematician's role was to reject false paths and enforce design principles, particularly hypothesis discipline:

> *"You keep doing the same thing – adding hypotheses that are actually lemmas to be proven! Why do you do that?"* (Mar 2)

This became the project's central methodological principle, codified in `CLAUDE.md`: a `sorry` (an acknowledged gap) is preferable to an unnecessary hypothesis (which silently weakens the theorem).

4.4. Phase 3: Abstract Proof Chain (Mar 3 – 7)

The monolithic file was split into a `main/` directory with separate files for each section of the proof. A `FlatTorus3` typeclass abstracted the spatial domain, specifying integration by parts, curl/divergence identities, a maximum principle, and constancy of harmonic functions – without fixing a particular manifold. The mathematical argument (Sections 2 – 8) was formalized against this interface.

A concrete `TorusInstance` module proved that $\mathbb{T}^3 = (\mathbb{R}/\mathbb{Z})^3$ satisfies all 22 fields of `FlatTorus3`, using the

Date	Event	Sorry count
Feb 28	Gemini generates natural-language proof	–
Mar 1	First LEAN code committed (monolithic file)	$\sim$25
Mar 2	First Aristotle submissions; hypothesis creep begins	25
Mar 3	File split into `main/` directory; Aristotle results arrive	$25 \to 7$
Mar 5	Concrete torus representation chosen (with Gemini)	$7 + 22$
Mar 7	Abstract theorem reaches 0 `sorry`'s	0
Mar 7	`/critique` adversarial review introduced	0
Mar 8	`/babysit` automation loop created (1:09 AM)	0
Mar 8	Coulomb kernel instantiation begun; university server online	$+35$
Mar 9	Coulomb theorem reaches 0 `sorry`'s (11:15 PM)	0
Mar 10	Cleanup (-3K lines); non-vacuousness theorem	0

Table 2. Project timeline and sorry count evolution.

MATHLIB representation `Fin 3` $\to$ `AddCircle 1`. The representation was chosen in consultation with Gemini. Periodicity is automatic: functions on the torus factor through the quotient map $\mathbb{R} \to \mathbb{R}/\mathbb{Z}$.

By March 7, all `sorry`'s in the abstract proof chain were closed – the abstract theorem was fully proved.

4.5. Phase 4: Automation (Mar 7 – 8)

On March 7, an adversarial self-review process was introduced:

> *"You are a hostile reviewer trying to REJECT this formalization. Your job is to find every weakness, gap, and dishonesty."*

The output, `critique.md`, identified false hypotheses, unnecessary assumptions, and dead code. On March 8 at 1:09 AM, the automation suite was created in a 20-minute session. The initial `/babysit` loop combined four slash commands (`/critique`, `/prove`, `/check-aristotle`, `/cleanup`) and was set to run on a 10-minute timer. Over the following days, the loop was iteratively expanded to 11 steps – adding `/plan`, `/simplify`, `/strengthen`, `/submit-aristotle`, `/log`, `/commit`, and `/alert` (Telegram notifications) – and executed 122 documented cycles over the project.

4.6. Phase 5: Coulomb Kernel (Mar 8 – 9)

With the abstract theorem proved, the final challenge was instantiating it for the physical Coulomb kernel $\Psi(r) = r^{-3}$. This required proving that the Landau collision operator satisfies all 17 integrability, differentiability, and continuity conditions in the `VelocityDecayConditions` bundle.

The Coulomb singularity at $r = 0$ makes $|A_{ij}(z)| \sim \|z\|^{-1}$, which is barely integrable in $\mathbb{R}^3$. Each integrability proof required decomposing the domain into a ball around the origin (handled by local integrability of $\|z\|^{-1}$) and the complement (handled by Schwartz decay). This phase was the hardest part of the formalization, producing $\sim$4,000 lines of analytical estimates.

Metric	Value
Development period	10 days
LEAN 4 codebase	34 files / 10,445 LOC
Git commits	213
Human prompts (excl. slash commands)	229
`/babysit` cycles	122
Assistant activity	27,186 turns / 17,334 tool calls
Token use	2.8B input / 11M output
Aristotle submissions	220
Aristotle outcomes	111 proved / 28 disproved

Table 3. Key project statistics. The 229 human prompts exclude bare slash commands like `/babysit`; all are archived with timestamps. The mathematician estimates $\sim$50 hours of active supervision over 10 days.

A university server was brought online on March 8 to run `/babysit` loops concurrently with the local machine, enabling continuous progress. The sorry count dropped from 35 to 0 on March 9, with the last sorry closed at 11:15 PM.

4.7. Phase 6: Cleanup (Mar 10)

With zero `sorry`'s achieved, the `/babysit` loop shifted to code quality: removing dead code (-3K lines), eliminating unnecessary `maxHeartbeats` overrides, and re-enabling linters. A separate non-vacuousness theorem (`CoulombConcreteTheorem42_nonvacuous`) verified that the 12 hypotheses are satisfiable – the equilibrium Maxwellian with $E = 0$, $B = 0$ witnesses all conditions. This sanity check was suggested independently by expert reviewers on the Lean Zulip.

5. Statistics

Table 3 summarizes the project. The 2.8 billion input tokens are dominated by cache reads of the LEAN codebase and MATHLIB context – the 254:1 input-to-output ratio reflects the read-heavy nature of formal verification. At Opus 4 API

pricing with prompt caching, this would cost approximately \$6,300: \$4,071 in cache reads (\$1.50/M), \$1,401 in cache creation (\$18.75/M), \$826 in output (\$75/M), and \$6 in fresh input (\$15/M). Without prompt caching, the same workload would cost ∼\$42,700 – caching saves 85%. The mathematician paid \$325 in subscriptions: \$200 for Claude Max and \$125 for Gemini.

The development trajectory is shown in Figure 4: the sorry count follows a sawtooth pattern as scaffolding and proving alternate. The appendix contains the corresponding LOC trajectory and code-composition plots (Figures 5 and 11), session activity (Figure 6), token consumption (Figure 7), tool usage (Figure 8), Aristotle outcomes and turnaround times (Figures 9 and 10), git churn (Figure 12), extended lessons learned, and concluding discussion (Sections A and B).

6. Related Work

AI-assisted theorem proving. Seed-Prover (Chen et al., 2025) uses reinforcement learning with LEAN feedback for competition mathematics. Aristotle (Achim et al., 2025) combines automated proving with LEAN integration and has been used to auto-formalize GPT-5.2-generated proofs of open Erdős problems (Sothanaphan, 2026). Google DeepMind's Aletheia agent (Feng et al., 2026b;a) produces natural-language research papers, and AxiomProver generated formal LEAN proofs of Fel's conjecture, partial regularity of primes, and parity of k-differentials from natural-language statements (Chen et al., 2026c;b;a). These systems are more autonomous on short theorem-proving tasks, often starting from a relatively compact natural-language problem statement. Our contribution is different in scale and texture: the theorem required new definitions for the Landau operator and torus calculus, substantial decay infrastructure, and Coulomb singularity estimates before the final proof could even be stated.

Agentic formalization. Numina-Lean-Agent (Liu et al., 2026) combines Claude Code with MCP tooling and formalized effective Brascamp-Lieb inequalities with human experts collaborating directly on the LEAN code. Archon (FrenzyMath, 2025) uses a dual-agent architecture for First-Proof problems. Bayer et al. (2025) formalized a new theorem on Diophantine complexity bounds in Isabelle in parallel with the mathematical research; that in-situ formalization was written by a student team by hand. Our work differs in that a single mathematician wrote no proof-assistant code or supporting scripts: Gemini produced the initial mathematical blueprint and Claude Code generated the LEAN development from natural-language supervision.

Large-scale autoformalization and physics. Math.Inc's Gauss agent (Math, Inc., 2026) eliminated ∼160K lines of sorry's in the sphere-packing formalization, starting from definitions and theorem statements written by human experts. Urban (2026) formalized a large topology textbook corpus using language models and a fast proof checker. These projects are larger but target known mathematics. In mathematical physics, Tooby-Smith (2026) formalized two-Higgs-doublet-model stability in LEAN/PhysLean and found an error in a published proof. Our project is smaller than the largest corpus-scale efforts, but it targets a new VML equilibrium theorem and required both domain modeling and proof engineering inside the same short development loop.

Transparency. A distinguishing feature of this project is process transparency. Many theorem-proving systems are closed-source, and even open systems rarely publish the full interaction trace. Here the prompts, commits, Gemini dialogue, Aristotle submissions, and automation logs are part of the artifact. This matters because the methodology itself is under evaluation: the logs expose not only successes but also false conjectures, failed automation cycles, and human interventions.

7. Limitations

Mathematical novelty. The theorem does not appear in the literature as a stated and proved standalone result, but its ingredients are classical. The Landau H-theorem and nullspace characterization are standard (Villani, 2002; Cercignani, 1988); polynomial matching for local Maxwellians appears in Cercignani (1988); Desvillettes & Villani (2005); and maximum-principle arguments for Poisson–Boltzmann equations are textbook. Prior work assembles these ingredients for simpler systems or studies dynamical stability near a Maxwellian (Desvillettes & Villani, 2005; Guo, 2003; 2012). What appears to be new is the complete static characterization for the electromagnetically coupled system: the Lorentz force contributes only lower-order velocity terms for a local Maxwellian, the high-order polynomial constraints still force constant temperature and a Killing bulk velocity, Ampère's law then forces zero bulk velocity, and Gauss's law closes the argument with $E = 0$ and uniform density. These steps are individually straightforward, but their assembly into a self-contained VML equilibrium theorem is the mathematical target of the formalization.

Definition alignment. A post-completion expert review found that an early version used `ContDiff ℝ ⊤`, which in MATHLIB means analytic rather than C^∞. The fix replaced it with explicit finite smoothness and strengthened the theorem, but the episode illustrates the main risk of AI-generated formalizations: a proof can compile while the statement fails to express the intended mathematics. Similar definition-level mismatches in project-specific definitions,

such as the torus differential operators or the Landau collision operator wrapper, cannot be ruled out without expert audit.

Reproducibility and cost. The workflow depends on the capabilities of Claude Opus 4, Gemini DeepThink, Aristotle, and the LEAN tooling available during the project period. These systems change quickly, so exact reproduction with future versions may not follow the same path, and Aristotle's search behavior is not a deterministic local proof script. The subscription cost was $325, but the API-equivalent cost would have been roughly $6,300 with prompt caching and roughly $42,700 without; at API pricing without caching, the workflow would be much less accessible.

Human effort. No code was written by the mathematician, but the process still required about 50 hours of expert supervision. The supervisor had to understand the theorem well enough to reject false assumptions, catch weakened statements, and decide when formal definitions matched the intended objects. Current systems can perform much of the proof-assistant engineering, but they do not remove the need for mathematical judgment.

8. Lessons Learned

The central methodological lesson is that an AI formalization agent must be supervised for theorem integrity, not merely for compilation. The most persistent failure mode was *hypothesis creep*: when a lemma was hard, the agent often tried to add it to the theorem assumptions. This preserves a green build while silently weakening the mathematical claim. The project therefore treated a visible `sorry` as preferable to an unnecessary hypothesis, because a `sorry` records an honest proof obligation while an assumption can hide the gap. The final theorem has 12 physically meaningful hypothesis families; at one intermediate point the abstract theorem had accumulated 42 assumptions, many of which were actually lemmas that had to be proved.

Automated proving changed the working rhythm. Aristotle was useful not only because it proved 111 lemmas, but because it disproved 28 false statements. These negative results caught missing measurability assumptions, pathological counterexamples, and incorrect Coulomb decay estimates before they became embedded in the architecture. The practical pattern was to decompose aggressively, submit small statements, and treat both proved and disproved responses as feedback on the formal design.

Definition alignment remained the hardest correctness risk. After completion, an expert reviewer noticed that an early version used `ContDiff ℝ ⊤`, which in MATHLIB means analytic rather than C^∞. The fix strengthened the theorem by replacing the assumption with finite smoothness, but the episode is a warning: proof checking guarantees that the formal statement follows from the formal hypotheses; it does not guarantee that the definitions express the intended mathematics. The non-vacuousness theorem and expert review of theorem statements are therefore part of the scientific artifact, not optional polish.

The automation loop helped with maintenance, polling, and overnight progress, but the decisive choices were interactive: choosing the torus representation, separating the abstract proof from the concrete instance, and deciding which assumptions were mathematically legitimate. Gemini was most useful at the beginning, producing the natural-language proof blueprint; Claude Code was most useful inside the repository, where it could read goals, edit files, and react to LEAN diagnostics. The resulting workflow is semi-autonomous in a precise sense: the AI systems wrote the proof-assistant code, but the human remained responsible for posing the question, protecting the statement, and judging mathematical alignment.

Agent incentives. Claude Code often behaved like a programmer optimizing for a passing build: it would add assumptions, raise heartbeat limits, leave dead helper lemmas in place, or retry a failing proof strategy too long. The behavior looked like laziness, but it is better understood as local optimization for an easy success criterion. A compiled theorem with extra assumptions, large heartbeat budgets, stale helpers, or unnecessary abstraction layers is cheaper for the agent than a clean proof. The project therefore used adversarial self-review prompts, linter cleanup, and repeated human reminders that proof quality mattered more than a low sorry count. Current coding agents need explicit pressure against compilation-oriented shortcuts.

Architecture mattered. The abstract/concrete split was the most important software-engineering decision. The main mathematical argument was proved against a `FlatTorus3` interface, while the concrete torus instance and the Coulomb kernel estimates were discharged separately. This isolated changes to analytic infrastructure from the proof of the theorem itself. When the torus representation, smoothness hypotheses, and decay bundles changed, the abstract theorem did not have to be rewritten. The agent did not naturally find this modular structure at first; it had to be steered toward it. Once established, however, it made the final sprint tractable.

No proof-assistant code by hand. The mathematician wrote no LEAN code and no supporting scripts. The human inputs were natural-language prompts, design constraints, reviews of theorem statements, and mathematical judgments about whether definitions matched the intended objects. This does not mean the work required no expertise:

the supervisor had to understand the VML argument well enough to reject false assumptions and recognize definition-level mistakes. It does show that the barrier has shifted. A mathematician can now contribute primarily as a reviewer and architect while the agent performs the proof-assistant engineering.

Transparency is part of the claim. Because semi-autonomous formalization is easy to oversell, the process artifacts matter. The commits, prompts, Aristotle submissions, and automation logs document not only the successful path but also the false starts: disproved lemmas, hypothesis creep, definition bugs, and cleanup passes. These records make the claim falsifiable in a way that a polished final theorem alone would not. For research-level AI formalization, we view the interaction log as analogous to experimental provenance: it is evidence about how the result was obtained and where human judgment entered the loop.

9. Conclusion

This project shows that a research-level mathematical result can now pass through the full loop from conjecture to informal proof to machine-checked formalization in days rather than months. The result is not full autonomy: the human role moved from writing code to supervising claims, definitions, and proof architecture. But that shift is substantial. For mathematicians without deep LEAN expertise, agentic formalization plus automated proving can now produce a complete checked artifact, provided the human supervisor can still evaluate whether the artifact states the right theorem.

The reversal of the usual order is also notable. Here the formalization was completed before the polished mathematical manuscript. That suggests that, for some projects, machine-checked development may no longer be a delayed verification step after publication. It can instead become part of the research process itself: conjecture, generate a proof, formalize, audit definitions, and only then write the traditional exposition. The proof assistant kernel supplies the final logical check, while the mathematician supplies the semantic check that the formal theorem is the intended one.

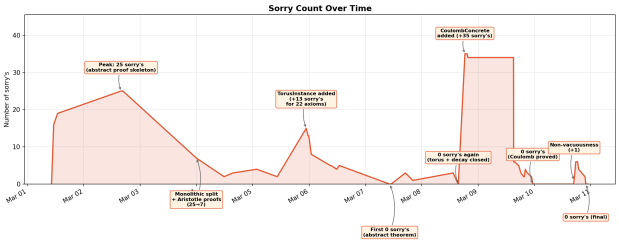

Figure 4. Sorry count in `main/` over time (85 commits). The sawtooth pattern reflects alternating phases of scaffolding (sorry's accumulate) and proving campaigns (sorry's eliminated). Key events: peak 25 on Mar 2 (abstract proof chain stated); Aristotle drops count from 25→7 on Mar 3; first 0 sorry's on Mar 7 (abstract theorem proved); +35 sorry's on Mar 8 (Coulomb kernel instantiation begun); 35→0 sprint on Mar 9; brief +1 on Mar 10 (non-vacuousness theorem, immediately resolved).

References

Achim, T., Best, A., Bietti, A., Der, K., Fédérico, M., Gukov, S., Halpern-Leistner, D., Henningsgard, K., Kudryashov, Y., Meiburg, A., Michelsen, M., Patterson, R., Rodriguez, E., Scharff, L., Shanker, V., Sicca, V., Sowrirajan, H., Swope, A., Tamas, M., Tenev, V., Thomm, J., Williams, H., and Wu, L. Aristotle: IMO-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025.

Bayer, J., David, M., Hassler, M., Matiyasevich, Y., and Schleicher, D. Diophantine equations over $\mathbb{Z}$: Universal bounds and parallel formalization. *arXiv preprint arXiv:2506.20909*, 2025.

Cercignani, C. *The Boltzmann Equation and Its Applications*, volume 67 of *Applied Mathematical Sciences*. Springer, 1988. doi: 10.1007/978-1-4612-1039-9.

Cercignani, C., Illner, R., and Pulvirenti, M. *The Mathematical Theory of Dilute Gases*, volume 106 of *Applied Mathematical Sciences*. Springer, 1994.

Chen, D., Chen, E., Lau, K., Ono, K., and Zhang, J. Parity of k-differentials in genus zero and one. *arXiv preprint arXiv:2602.03722*, 2026a.

Chen, E., Cummins, C., Eltschig, B., Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Markosyan, A., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Almost all primes are partially regular. *arXiv preprint arXiv:2602.05090*, 2026b.

Chen, E., Cummins, C., GSM, Grubisic, D., Haller, L., Hong, L., Kurghinyan, A., Lau, K., Leather, H., Lee, S., Markosyan, A., Ono, K., Patel, M., Pendharkar, G., Rathi, V., Schneidman, A., Seeker, V., Sengupta, S., Sinha, I., Xin, J., and Zhang, J. Fel's conjecture on syzygies of numerical semigroups. *arXiv preprint arXiv:2602.03716*, 2026c.

Chen, L., Gu, J., Huang, L., Huang, W., Jiang, Z., Jie, A., Jin, X., Jin, X., Li, C., Ma, K., Ren, C., Shen, J., Shi, W., Sun, T., Sun, H., Wang, J., Wang, S., Wang, Z., Wei, C., Wei, S., Wu, Y., Wu, Y., Xia, Y., Xin, H., and Yang, F. Seed-prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025.

de Moura, L. and Ullrich, S. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*. Springer, 2021. doi: 10.1007/978-3-030-79876-5_37.

Desvillettes, L. and Villani, C. On the trend to global equilibrium for spatially inhomogeneous kinetic systems: The Boltzmann equation. *Inventiones mathematicae*, 159:245–316, 2005. doi: 10.1007/s00222-004-0389-9.

Feng, T., Trinh, T., Bingham, G., et al. Semi-autonomous mathematics discovery with Gemini: A case study on the Erdős problems. *arXiv preprint arXiv:2601.22401*, 2026a.

Feng, T., Trinh, T. H., Bingham, G., et al. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026b.

FrenzyMath. Archon: AI-assisted mathematical formalization. https://frenzymath.com/blog/archon-firstproof/, 2025. Blog post.

Guo, Y. The Vlasov–Maxwell–Boltzmann system near Maxwellians. *Inventiones mathematicae*, 153:593–630, 2003. doi: 10.1007/s00222-003-0301-z.

Guo, Y. The Vlasov–Poisson–Landau system in a periodic box. *Journal of the American Mathematical Society*, 25(3):759–812, 2012. doi: 10.1090/S0894-0347-2011-00722-4.

Liu, J., Zhou, Z., Zhu, Z., Dos Santos, M., He, W., Liu, J., Wang, R., Xie, Y., Zhao, J., Wang, Q., Zhi, L., Li, J., and Li, W. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics. *arXiv preprint arXiv:2601.14027*, 2026.

Massot, P. LeanBlueprint: A plasTeX plugin to build formalization blueprints. https://github.com/PatrickMassot/leanblueprint, 2020. Software.

Math, Inc. Completing the formal proof of higher-dimensional sphere packing. https://www.math.inc/sphere-packing, 2026. Blog post.

Sothanaphan, N. Resolution of Erdős problem #728: a writeup of Aristotle's Lean proof. *arXiv preprint arXiv:2601.07421*, 2026.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2020)*, 2020. doi: 10.1145/3372885.3373824.

Tooby-Smith, J. Formalizing the stability of the two Higgs doublet model potential into Lean: Identifying an error in the literature. *arXiv preprint arXiv:2603.08139*, 2026.

Urban, J. 130k lines of formal topology in two weeks: Simple and cheap autoformalization for everyone? *arXiv preprint arXiv:2601.03298*, 2026.

Villani, C. A review of mathematical topics in collisional kinetic theory. In Friedlander, S. and Serre, D. (eds.), *Handbook of Mathematical Fluid Dynamics*, volume 1, pp. 71–305. North-Holland, 2002.

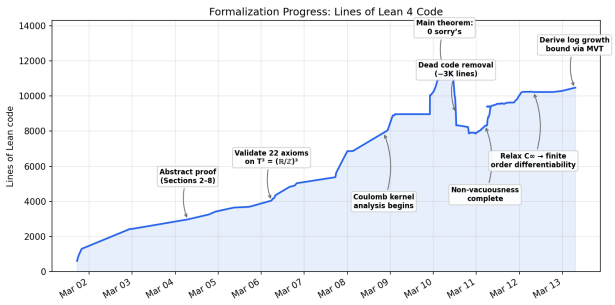

Figure 5. LEAN lines of code over time (all 213 commits). The project progressed in four phases: (1) Abstract proof chain (Mar 3 – 6), formalized against an abstract `FlatTorus3` typeclass. (2) Coulomb kernel analysis (Mar 7 – 9), handling the singularity at $r = 0$ – the sharp LOC increase reflects ∼4K lines of analytical estimates. (3) Cleanup (Mar 10, early): ∼3K lines of dead code, redundant lemmas, and unnecessary heartbeat overrides removed. (4) Non-vacuousness (Mar 10, late): satisfiability of the 12 hypotheses.

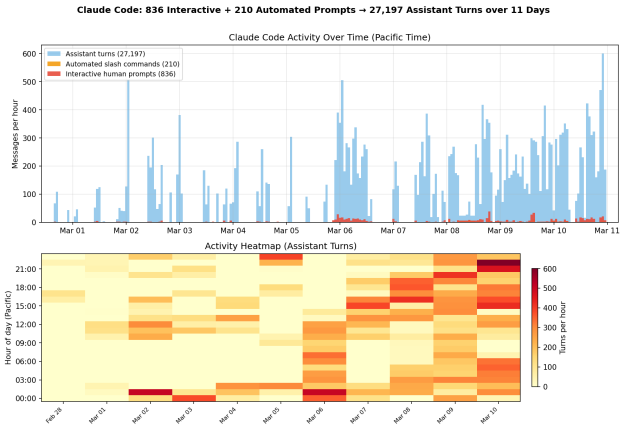

Figure 6. Claude Code activity across two machines (local + university server). Top: messages per hour (red = interactive human, orange = automated, blue = assistant). Bottom: hour-of-day heatmap. The 27,186 assistant turns versus 229 human prompts give a 119:1 ratio, reflecting the high autonomy of each `/babysit` cycle (typically 20 – 50 tool calls per invocation). The heatmap shows near-round-the-clock activity on Mar 9 – 10 when both machines ran `/babysit` loops concurrently.

A. Lessons Learned (Extended)

A.1. Hypothesis Discipline is Critical

The most persistent failure mode of Claude Code was *hypothesis creep*: when faced with a difficult lemma, the agent's default behavior was to assume it as a hypothesis of the main theorem rather than prove it. By March 4, the main theorem had accumulated 42 hypotheses – most of which should have been derived lemmas.

This failure mode is insidious because it is syntactically valid: the LEAN code compiles, the sorry count stays low, and the proof "works." But the theorem statement is progressively weakened. In the extreme case, one could "prove" any theorem by assuming its conclusion as a hypothesis.

The fix required persistent human intervention:

> *"I have told you time and again to keep the hypotheses to Theorem42 minimal. If you need a mathematical fact, it should be a lemma with a sorry, not a hypothesis!"* (Mar 3)

This was codified in `CLAUDE.md`: *"The goal is not to end up with 0 sorry's! The goal is to make an honest formalization of the main theorem, with only the genuinely needed mathematical/physical assumptions."* The 42 hypotheses were systematically reduced: velocity-space decay conditions were bundled into a single `VelocityDecayConditions` structure (replacing 15 scattered hypotheses), and regularity facts like continuity of the density ρ were derived inside the proof from domination bounds. The final concrete theorem has 12 hypotheses, all physically meaningful.

A.2. Aristotle is Transformative

The cloud-based theorem prover Aristotle (Achim et al., 2025) was the single most impactful tool after Claude Code itself. Of 220 submissions, 111 were proved outright (50%), and 28 were *disproved* – Aristotle found counterexamples or proved the negation.

The disproved statements were arguably more valuable than the proved ones. They caught false conjectures early: missing measurability hypotheses, claims that fail on pathological counterexamples (e.g., Vitali sets), and Schwartz decay estimates that are actually false for the Coulomb kernel. The fix-and-resubmit cycle served as the primary mechanism for catching bugs in the formalization.

Aristotle's overall median turnaround was 19 minutes (Figure 10). Proved lemmas returned in a median of 9 minutes; disproved statements took 29 minutes (counterexample construction); but submissions returned with `sorry` had a median of 5.6 hours, as Aristotle exhausted its time budget searching before giving up. We recommend decomposing proofs aggressively into small lemmas and submitting each to an automated prover – even lemmas that seem hard.

A.3. Definition Alignment Requires Expert Review

After the formalization was announced on the Lean Zulip, an expert reviewer immediately identified a definition-alignment bug: the original theorem used `ContDiff ℝ ⊤`, which in MATHLIB means *analytic*, not C^∞ as intended. Neither the AI agent nor the mathematician caught this during 10 days of development.

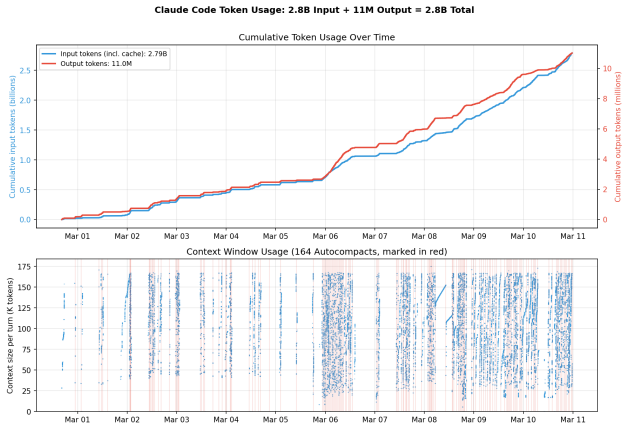

Figure 7. Token consumption across the project. Top: cumulative input tokens (blue, 2.8B) and output tokens (red, 11M). The 254:1 input-to-output ratio reflects the read-heavy nature of formal verification – Claude reads far more context than it writes. Bottom: per-turn context size, showing a sawtooth pattern as the context window fills to ∼175K tokens and resets at each of the 164 autocompacts (red vertical lines). Dense compaction clusters on Mar 6 – 10 correspond to the intensive Coulomb kernel analysis and `/babysit` loop periods.

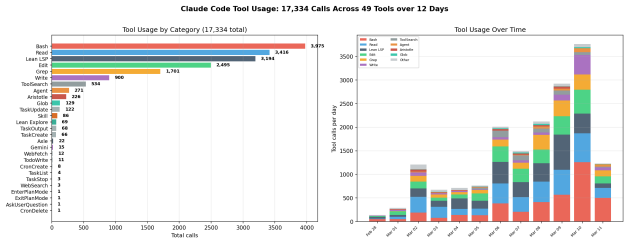

Figure 8. Tool usage across 17,334 invocations. Left: totals by category. The core editing loop dominates – Bash (3,975), Read (3,416), Edit (2,495), Grep (1,701), Write (900) – reflecting the iterative nature of formal verification. LEAN LSP tools (3,194 combined) provided real-time feedback. Right: daily breakdown showing the ramp-up from exploratory work (Feb 28 – Mar 2) through peak activity (Mar 6 – 10) when both machines ran `/babysit` loops concurrently.

The fix was straightforward – replacing $\top$ with explicit finite smoothness (C^3 in velocity, C^2 in space) – and the resulting theorem is actually *stronger*, since it assumes less. But the episode illustrates a fundamental risk of AI-generated formalizations: the code compiles and the proof checks, yet the theorem does not state what was intended. The same reviewer also suggested formalizing that the equilibrium Maxwellian satisfies all 12 hypotheses (the non-vacuousness theorem), which was implemented as an additional sanity check.

This lesson is perhaps the most important for the community: **compilation is not correctness**. Expert review of definitions and theorem statements remains essential, and cannot yet be delegated to the AI.

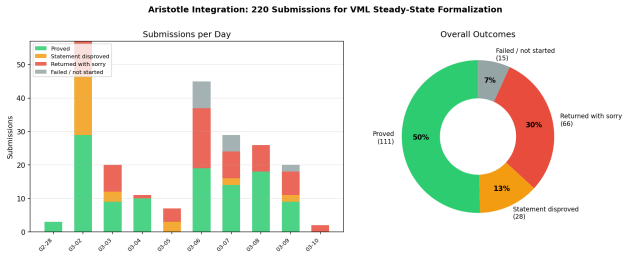

Figure 9. Outcomes of 220 Aristotle submissions. Aristotle proved 111 lemmas cleanly (50%), disproved 28 false statements (13%), returned 66 with sorry's still present (30%), and 15 failed or were never started (7%). The disproved statements were particularly valuable – they caught false conjectures early (e.g., missing measurability hypotheses, Vitali-set counterexamples, incorrect Schwartz decay estimates for the Coulomb kernel). The stacked bar chart shows peak submission activity on Mar 2 (57 jobs) and Mar 6 (45 jobs).

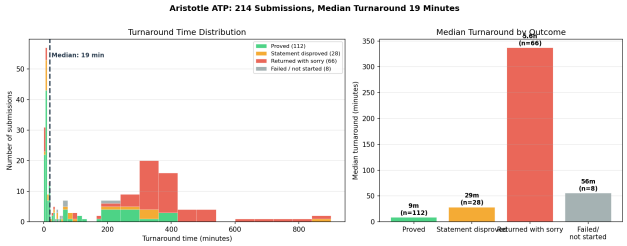

Figure 10. Aristotle turnaround times for 214 submissions, colored by outcome. Left: histogram. Proved lemmas (green) cluster in the first bins (median 9 minutes), while "returned with sorry" (red) dominates the long tail. Right: median turnaround by category. The overall median was 19 minutes. Disproved statements took 29 minutes (counterexample construction); submissions returned with sorry took a median of 5.6 hours – Aristotle exhausts its time budget before giving up.

A.4. Automation Helps but is Not Strictly Necessary

The `/babysit` loop (Section 4) ran for 122 cycles over the project. It was valuable for overnight progress and for maintaining momentum on tedious tasks (linter cleanup, dead code removal). However, the most important breakthroughs (closing the abstract theorem, proving the Coulomb integrability estimates) happened during interactive sessions with direct human guidance.

The automation was most effective for:

- **Maintaining code quality:** The adversarial `/critique` step caught issues that would otherwise accumulate (unused imports, stale comments, false docstrings).
- **Aristotle polling:** Checking for completed Aristotle jobs and integrating results.
- **Overnight progress:** Running unattended on two machines while the mathematician slept or traveled.

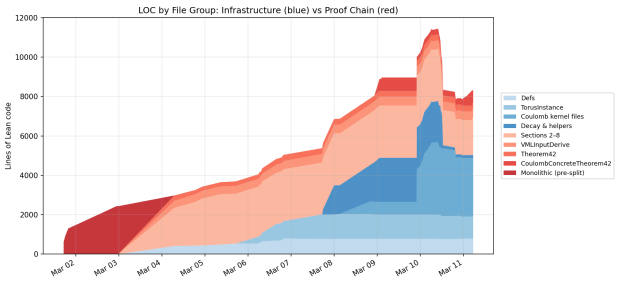

Figure 11. LOC by file group over time. Red shades are the mathematical proof chain (Sections 2–8, `Theorem42`, `VMLInputDerive`, `CoulombConcreteTheorem42`); blue shades are supporting infrastructure (definitions, torus instance, Coulomb kernel integrability files, decay helpers). At completion, infrastructure accounts for roughly half of the codebase – reflecting the substantial gap between MATHLIB's current coverage and the background material needed for research-level mathematical physics. The sharp blue growth on Mar 7–8 is the Coulomb kernel analysis ($\sim$4K lines of analytical estimates for the singularity at $r = 0$). The cleanup phase (Mar 10) primarily removed blue infrastructure code, while the red proof chain remained stable.

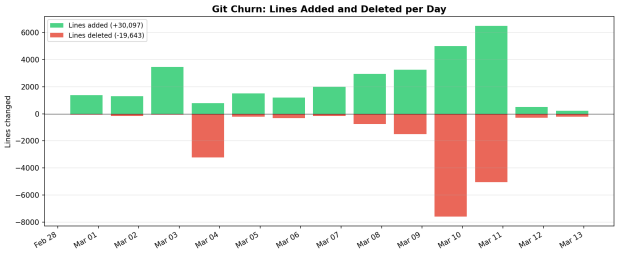

Figure 12. Lines of LEAN code added and deleted per day. Most days are net-positive (new content), with two exceptions: Mar 3 (monolithic file split into `main/` – same code reorganized) and Mar 9 – 10 (cleanup phase removed $\sim$3K lines of dead code, redundant lemmas, and unnecessary heartbeat overrides). Total: $+21{,}783$ added, $-13{,}478$ deleted, net $+8{,}305$ lines.

It was less effective for:

- **Novel proof strategies:** The agent often got stuck in loops, retrying the same failing approach rather than reconsidering the proof strategy.
- **Architectural decisions:** Choosing representations, structuring typeclasses, and deciding what to abstract required human judgment.

A.5. Gemini for Mathematical Reasoning

Gemini DeepThink served two distinct roles: (1) generating the initial proof blueprint, and (2) providing mathematical advice during the formalization. For the first role, it was excellent – the proof it produced (Section 4) served as the complete blueprint for the 10K-line formalization. For the second role, it was useful but not essential: Claude Code's own mathematical reasoning was often sufficient for tactical decisions. In the mathematician's own assessment: *"I did not find Gemini to be that terribly useful [beyond the initial proof]. It was mostly for experimental sake."*

The key insight is that the full loop of mathematical research – from conjecture to resolution to detailed proof – can now be completed by AI with the human acting as verifier rather than creator. Proof generation and proof formalization are different skills, and a model that excels at informal mathematical reasoning (Gemini) can be productively combined with a model that excels at code generation (Claude), with each handling its strength. The LEAN kernel then provides the final guarantee of correctness that neither model can offer alone.

A.6. The Agent Gets "Lazy"

Beyond hypothesis creep, Claude Code exhibited several avoidance behaviors:

- **Premature `sorry`:** Marking a goal as `sorry` and moving on, rather than attempting the proof.
- **Excessive heartbeats:** Setting `maxHeartbeats 800000` to suppress timeouts rather than simplifying the proof.
- **Dead code accumulation:** Leaving unused lemmas, commented-out code, and stale helper definitions rather than cleaning up.
- **Over-engineering:** Creating unnecessary abstraction layers, helper functions, and intermediate definitions when a direct proof would suffice.
- **No-op cycles:** During automated `/babysit` runs, the agent would sometimes complete a cycle with no meaningful changes despite open issues in `critique.md`, requiring prompt rewrites to enforce progress.

These behaviors are not bugs per se – they are rational strategies for an agent optimizing for compilation success rather than proof quality. The fix is explicit instructions (`CLAUDE.md`) and persistent human supervision. The adversarial `/critique` step in the `/babysit` loop partially automated this supervision.

A.7. The Abstract/Concrete Split Pays Off

Separating the proof into an abstract framework (`FlatTorus3` typeclass) and a concrete instantiation (`TorusInstance`) was an early design decision that paid dividends throughout. When the torus proofs required rewrites – removing false hypotheses, changing the representation of spatial differentiability – the abstract proof chain (Sections 2 – 8) was completely unaffected. This

separation of concerns is standard software engineering practice, but it is not the default behavior of the AI agent, which tends to produce monolithic proofs.

A.8. No Lean Code Required

The mathematician wrote zero lines of LEAN code and zero lines of any other programming language. All LEAN code was generated by Claude Code from natural-language prompts. Of the 229 human prompts sent to Claude Code (excluding bare slash commands and context continuations), most were short directives or confirmations. The mathematician estimates approximately 50 hours of active supervision – an average of 5 hours per day watching Claude Code work in real time.

The mathematician's contributions were:

1. Posing the conjecture.
2. Checking the final theorem statement (12 hypotheses, each evaluated for physical correctness).
3. Enforcing hypothesis discipline and architectural decisions.
4. Providing mathematical intuition when the agent was stuck.
5. Designing and iterating on the automation suite.

The mathematician did read key definitions and theorem statements in LEAN to verify alignment with the intended mathematics, but did not debug proofs or lemma statements directly.

B. Discussion

We have demonstrated that the full loop of AI-assisted mathematical research can be closed in a matter of days: starting from an open conjecture, an AI reasoning model (Gemini DeepThink) correctly resolved the question and produced a detailed proof requiring no corrections; an agentic coding tool (Claude Code) translated this proof into 10K+ lines of LEAN 4 from natural-language prompts alone; a cloud-based prover (Aristotle) closed individual lemmas and caught false conjectures; and the LEAN kernel verified the final result with guaranteed logical correctness. A single mathematician supervised the entire process at a subscription cost of $325, writing zero lines of code.

The process is not yet fully autonomous – the human's role in maintaining theorem integrity is essential and cannot currently be delegated to the AI. But the barrier to entry has dropped dramatically: from years of LEAN expertise to the ability to supervise an AI agent and evaluate theorem statements.

A striking feature of this project is that the formalization was completed *before* the traditional mathematics paper containing the informal proof. The 10-day formalization finished on March 10; the companion math paper, which requires cleaning up the natural-language arguments and obtaining co-author sign-off, is still in preparation. This reversal of the usual order – where formalization lags publication by months or years – suggests that AI-assisted formalization has reached a point where it takes comparable effort to produce a machine-checked proof and to write a polished mathematical manuscript.

We believe this workflow – conjecture, generate proof, formalize, verify, review – will become a common mode of computer-assisted mathematical research. Each stage of the pipeline is now within reach of current AI systems: reasoning models can resolve conjectures and produce detailed proofs, coding agents can translate these proofs into formal languages, automated provers can close gaps and catch errors, and proof kernels provide the final guarantee of correctness. The mathematician's role shifts from writing proofs to posing questions and evaluating claims – a supervisor of an AI-driven research pipeline rather than its sole executor.