©Copyright 2026
Vasily Ilin

Neural Methods for Plasma Simulation and for Mathematical Formalization

Vasily Ilin

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2026

Reading Committee:

Jingwei Hu, Chair

Jarod Alper, Chair

Bamdad Hosseini

Soumik Pal

Kevin Jamieson

Program Authorized to Offer Degree:
Mathematics

University of Washington

Abstract

Neural Methods for Plasma Simulation
and for Mathematical Formalization

Vasily Ilin

Co-Chairs of the Supervisory Committee:
Professor Jingwei Hu
Department of Applied Mathematics

Professor Jarod Alper
Department of Mathematics

This dissertation consists of several applications of neural networks to plasma simulation and to mathematical formalization. It synthesizes eight papers into seven thematic chapters across three parts.

While nuclear fusion promises an abundance of clean energy, achieving it requires fast and accurate numerical methods to simulate plasma, whose collisional dynamics are governed by the Landau and Vlasov–Maxwell–Landau (VML) equations. I present a family of deterministic particle methods that exploit the fact that these equations can be written as continuity equations whose velocity field depends on the density only through the score function $\nabla \log f$. Replacing the kernel density estimate used in the classical blob method with a neural network trained online via implicit score matching produces a particle scheme that scales linearly rather than quadratically in the number of particles, conserves mass, momentum, and energy by construction, and avoids the unphysical truncation of velocity tails to which kernel methods are prone. The resulting solver is both faster and more accurate than the previous state of the art. The numerical advances are accompanied by theoretical guarantees: stability of the spatially homogeneous Coulomb-case Landau equation in relative entropy, an error bound on the score-based approximation in terms of the score-matching

loss, and a characterization of the global equilibria of the VML and Vlasov–Poisson–Landau systems on the torus.

The same score-based viewpoint extends beyond plasma to general distributions. I apply online score learning to a strictly harder problem than diffusion generative modeling – deterministic sampling from an unnormalized target density given only its score and no samples – and prove the optimal exponential entropy dissipation rate through a neural tangent kernel analysis of the coupled particle–network gradient flow. I then turn score estimation itself into the object of study and develop DiScoFormer, a permutation- and affine-equivariant Transformer that amortizes density and score estimation across distributions, sample sizes, and dimensions, with the property that a single self-attention head exactly reproduces Gaussian kernel density estimation – making precise the relationship between attention and classical nonparametric statistics.

While neural network-based AI promises an abundance of cheap intelligence, it comes at the cost of hallucinations. On the other hand, formal languages like Lean can guarantee complete correctness but are tedious to program in. I present several applications of AI to formalizing mathematics in Lean, organized as the discovery, repair, and formalization workflow of working mathematicians. We built a semantic search engine over theorem statements extracted from arXiv and the Stacks Project, which outperforms LLM-based search. We constructed APRIL, a dataset of broken-then-repaired Lean 4 proofs paired with compiler diagnostics, on which a fine-tuned small language model matches the state-of-the-art model at single-shot proof repair. Finally, I present a Lean 4 formalization of an equilibrium theorem for the Vlasov–Maxwell–Landau system, proved informally in Part I. The experiment illustrates how AI and formal verification become complementary: verification eliminates the hallucinations of AI, while AI makes formalization practical at research scale.

TABLE OF CONTENTS

Page

List of Figures iv

List of Tables xii

Chapter 1: Introduction 1
1.1 Kinetic equations for collisional plasma 3
1.2 Score-based transport modeling 5
1.3 AI for mathematics 6
1.4 Contributions and outline 8

Chapter 2: Score-Based Particle Methods for Collisional Plasma 11
2.1 Homogeneous Landau equation 11
2.2 Vlasov–Maxwell–Landau system 26

Chapter 3: Theory for Collisional Plasma and Score-Based Numerics 44
3.1 Stability of the homogeneous Landau equation 45
3.2 Error control for score-based approximations 53
3.3 Equilibrium of the Vlasov–Maxwell–Landau system 63

Chapter 4: Score-Based Deterministic Density Sampling 67
4.1 Introduction 68
4.2 Deterministic sampling algorithm 70
4.3 Convergence analysis 74
4.4 Numerical experiments 81
4.5 Conclusion 89

Chapter 5: DiScoFormer: Plug-In Density and Score Estimation with Transformers 91
5.1 Introduction 92

5.2 Related work . . . 94
5.3 Methodology . . . 96
5.4 Experiments . . . 101
5.5 Conclusion . . . 108

Chapter 6: Semantic Search over 9 Million Mathematical Theorems . . . 123
6.1 Introduction . . . 124
6.2 Related work . . . 125
6.3 Data collection . . . 126
6.4 Experiments . . . 133
6.5 Search tool . . . 140
6.6 Conclusion . . . 144
6.7 User feedback . . . 144
6.8 Claude experiment . . . 146

Chapter 7: Learning to Repair Lean Proofs from Compiler Feedback . . . 151
7.1 Introduction . . . 152
7.2 Related work . . . 153
7.3 Methodology . . . 155
7.4 APRIL . . . 159
7.5 Results . . . 161
7.6 Conclusion . . . 164

Chapter 8: Semi-Autonomous Formalization of the Vlasov–Maxwell–Landau Equilibrium . . . 170
8.1 Introduction . . . 171
8.2 The Theorem . . . 174
8.3 Tools . . . 176
8.4 Process . . . 177
8.5 Statistics . . . 180
8.6 Lessons Learned . . . 181
8.7 Related Work . . . 186
8.8 Limitations . . . 188
8.9 Conclusion . . . 189

Bibliography 203

LIST OF FIGURES

Figure Number Page

2.1 Effect of underfitting in the score-matching loss. Top row: non-adaptive schedule (fixed K). Bottom row: adaptive schedule. Left column: $\eta = 4 \times 10^{-4}$ (good). Right column: $\eta = 10^{-4}$ (too small). The black curve is the true weighted score-matching loss (2.9), the orange curve is the unweighted implicit loss (2.12), and the blue curve is the denoising loss (2.14). 19

2.2 BKW solution to the Maxwellian Landau equation in $d = 3$. Both blob and SBTM recover the analytic solution accurately; the conserved-mean error is at the level of floating-point round-off. 21

2.3 Anisotropic Maxwellian Landau equation in $d = 3$. SBTM tracks the analytic covariance trajectory and the entropy decay rate at $n = 100$; the blob method requires more particles for comparable accuracy. 22

2.4 Anisotropic Maxwellian Landau equation in $d = 10$. Even at $n = 25{,}600$ the blob method does not track the analytic covariance trajectory, whereas SBTM remains accurate. 23

2.5 Anisotropic Coulomb Landau equation in $d = 3$. SBTM at $n = 100$ matches the $n = 12{,}800$ reference trajectories more accurately than the blob method. The dashed line is the Maxwellian steady state $\Sigma_\infty = I_3$. 24

2.6 Wall-clock time vs. number of particles n on CPU (left, 10 threads) and GPU (right) on log-log axes. SBTM scales linearly (sublinearly on GPU); the blob method scales quadratically. The GPU timing is for the heat equation. . . . 25

2.7 Landau damping at $\nu = 0.4$. SBTM dissipates entropy consistently and maintains near-constant total energy across all particle counts; the blob method overestimates entropy production and exhibits larger energy drift. 33

2.8 Landau damping: L^2 norm of the electric field at $\nu = 0.4$ across particle counts. The blob damping rate varies with n (panels (a), (c), (e)) and converges to the SBTM rate; the SBTM rate is consistent across all n. 38

2.9 Landau damping: velocity-space distributions at $n = 10^6$, $\nu = 0.4$. SBTM (right column) maintains smooth Gaussian tails; the blob method (left column) shows unphysical sharp cutoffs at $|v_2| \approx 3$. 39

2.10 Two-stream instability at $\nu = 0.32$. Blob (blue) vs. SBTM (red) at $n = 5\times10^5$ (solid), 10^6 (dashed), 3×10^6 (dotted). SBTM curves nearly coincide across n. 40

2.11 Two-stream instability: (x, v_1) phase space at $\nu = 0.32$. SBTM achieves full vortex dissipation at every particle count; the blob method requires $n = 3\times10^6$ to approach the same level. 41

2.12 Two-stream instability: score quiver plots at $\nu = 0.24$, $n = 10^6$. Top: initial time (true score in red). Bottom: final time. SBTM produces smooth, physically consistent scores at both times. 42

2.13 Weibel instability: v_2-marginal density evolution at $\nu = 8 \times 10^{-4}$. Top: log scale. Bottom: linear scale. Dashed line: target Maxwellian $\mathcal{N}(0, T_\infty)$. SBTM converges to it with smooth Gaussian tails; the blob method exhibits unphysical sharp cutoffs at $v_2 \approx \pm0.55$. 43

4.1 Left: Langevin dynamics (stochastic). Right: ours (deterministic). The deterministic algorithm has the same marginal distributions as the stochastic one but with smooth trajectories. In this plot both algorithms interpolate between the unit Gaussian and a mixture of two Gaussians. 70

4.2 Visualization of (GF ODE). Particles are pulled towards target π and away from particle density f_t. Brownian motion acts randomly. 71

4.3 Experiment 4.4.1, log-concave target. Top: relative entropy dissipation rate of SBTM (ours) and SDE (stochastic). SBTM approximates entropy decay rate well, while SDE is noisy. Bottom left: relative entropy of SBTM, SDE and the ground truth. SBTM approximates the ground truth well. Bottom right: L2 error to the true ground truth solution. SBTM produces lower error with smoother trajectory. 83

4.4 Experiment 4.4.2, 1D Gaussian mixture. Left: KL divergence of SBTM (ours) and SDE (stochastic) over time. SBTM exhibits smoother convergence. Right: entropy dissipation of SBTM and SDE. SBTM approximates entropy decay rate perfectly with the computable quantity $\mathrm{F}(f_t \,\|\, \pi)$, while SDE is noisy. . . 84

4.5 Experiment 4.4.3, well-separated 1D Gaussian mixture. Left: reconstructed density of SBTM. It approximates the solution well despite the non-log-concavity. Right: entropy dissipation of SBTM (ours) and SDE (stochastic). SBTM approximates entropy decay rate perfectly even in annealed dynamics. 86

4.6 Experiment 4.4.4, noisy circle. SBTM (ours, top) leaves the vacuum region empty, while SDE (bottom) fills it, demonstrating the effect of determinism. 87

4.7 Experiment 4.4.5, well-separated 2D Gaussian mixture. Scatter plot over time. Top: SBTM (ours) with the dilation annealing. Bottom: SDE with the dilation annealing [29]. SBTM separates into modes early on, compared to the SDE. 88
4.8 Experiment 4.4.5, well-separated 2D Gaussian mixture. The estimate in (4.4) holds empirically, indicating small score-matching loss and good score approximation in annealed dynamics. 88
4.9 Experiment 4.4.6, high-dimensional. Sampled MNIST digits using $f_0 = \mathcal{N}(0, 1)$, sample size 64, time step 10^{-3}, 30 steps, variable amount of intermediate training. Digits generated by SBTM are competitive in visual quality. 89
4.10 Experiment 4.4.6, high-dimensional. Starting from the same initial point, SBTM produces distinct sample trajectories depending on the training schedule. The amount of training controls the strength of interaction between particles. Top to bottom: SBTM without training (equivalent to gradient ascent on $\nabla \log \pi$), SBTM with small amount training, SBTM with large amount training, and finally the SDE (Langevin). 89
4.11 Experiment 4.4.6, high-dimensional. Cosine similarity between the true score $\nabla \log \pi$ and learned score $\nabla \log f_t$ over simulation time. Even with very little training the model learns the score well. Different lines use different numbers of epochs per time-step, effectively changing η in (4.10). 90

5.1 Attention visualization of the transformer model. The top panel shows the average attention scores in layer 0 with respect to the chosen point x. The heatmaps show the attention matrix and the normalized KDE matrix $D_{i,j} \propto e^{-\|x_i - x_j\|_2^2}$. The scatter plots show very high agreement between attention scores and D. The bottom four panels visualize the attention scores of individual heads, demonstrating emergent head specialization. 109
5.2 The forward pass implementing affine equivariance. 110
5.3 Score estimation comparison between Silverman KDE [156] and our transformer model. The transformer is more accurate, especially in the sparse regions. We plot the negated score for easier viewing. 111
5.4 MSE of score estimation on the Laplace distribution. Test-time training (TTT) improves the out-of-distribution generalization. 112

5.5 Left: MSE of score estimation in dimensions 1 and 10 on a 3-modal GMM using Transformer and KDE. The Transformer has excellent scaling in both dimension d and the sample size n. Right: MSE of score estimation using the Transformer on GMMs with different numbers of modes. Despite being trained only on GMMs with 1–10 modes, and $n = 2048$, the model exhibits excellent generalization. 113
5.6 Computing relative Fisher information. Our model predicts $\nabla \log g(x_i)$ at query points x_i via cross-attention with the samples $y_i \sim g$. These predicted scores can be used to estimate the relative Fisher information. 114
5.7 Qualitative comparison: true density, Silverman KDE, Emp-SD-KDE, and SD-KDE with our learned score. Learned-score SD-KDE is visibly less biased. 115
5.8 1D bimodal GMM ($n = 2048$): SD-KDE with the learned score best approximates the density by MSE. 115
5.9 MSE of density estimation in 1D (left) and 2D (right). SD-KDE with our learned score and the Transformer model show the best scaling. The Transformer was trained only at $n = 2048$. 116
5.10 Comparison between the transformer model (learned), Silverman KDE, and score-debiased KDE for estimation of differential entropy $H(f)$ and Fisher Information $I(f)$. The Transformer's MSE is lower than that of the KDE approximation, even in dimension 1. 117
5.11 Comparison between the trained Transformer model and Silverman KDE on the task of numerically solving the homogeneous Landau equation. We plot the first entry of the covariance matrix $\Sigma_{1,1}(t)$ of the numerical solutions and ground truth, when known. The left two panels use Maxwell collisions, while the last panel shows Coulomb collisions. The Transformer outperforms KDE, and is comparable in quality to SBTM in [91]. 118
5.12 Comparison of sliced score matching and our transformer model. The transformer model can be used without retraining and does not suffer from overfitting. We plot the negated score for ease of visualization. 119
5.13 Runtime comparison between KDE and the Transformer model in 2 and 10 dimensions. Both are $O(n^2)$ asymptotically, but empirically the Transformer scales better and has improved memory efficiency. KDE encounters an OOM error at $n = 2^{15}$. 120

5.14 Attention of the eight individual heads of layer 0 as a heatmap. Yellow color denotes higher attention weight and blue is lower. We choose the particle ordering so that nearby particles are close in the ordering. Heads 0, 2, and 5 specialize on nearby points, heads 1 and 6 specialize on far-away points, whereas heads 3, 4, and 7 attend in specific directions. 121

5.15 Attention of the eight individual heads of layer 0 as scatter plots. We choose a random query point, marked with a red x. Yellow color denotes higher attention weight and blue is lower. Heads 0, 2, and 5 specialize on nearby points, head 1 specializes on far-away points, whereas heads 3, 4, 6, and 7 attend in specific directions. 122

6.1 Overview of the creation of our database and search engine. Items with a "*" are variable and we tested different methods during our experiments (Section 6.4). 127

6.2 Sunburst plot of the distribution of arXiv tags across theorems in our dataset. 128

6.3 PCA visualizations of theorems in our dataset using slogan embeddings. Left: Gemma embedding. Right: Qwen3 8B embedding. 141

6.4 UMAP visualizations of 10,000 theorem-slogan embeddings across the ten most common arXiv categories. Gemma 0.3B (left) and Qwen3 8B (right). Qwen3 8B produces tighter, better-separated clusters. 142

6.5 The interface of the Theorem Search Tool. The main window displays the top results: paper name, paper authors, theorem statement, and link to the paper. In the left pane users can filter by paper metadata: type of result (theorem/proposition/lemma/corollary), author(s), arXiv tag, specific paper, year, and publication status. The thumbs up/down allow for user feedback, used to further improve the search results. 143

7.1 Overview of our dataset collection pipeline that collects paired correct and incorrect proof and their contextual information. We collected correct proofs from public datasets (Herald, Lean Workbook, NuminaMath-Lean). The paired error proofs that maintain the same proof sketch are generated by mutating tactics, lines of code, or theorems. Then, we used the Lean compiler to filter out the error proofs and extract their error messages, error lines, and goal states. Finally, we prompted an LLM to generate error explanations and fix suggestions based on the paired proofs and Lean Infoview. The resulting dataset pairs erroneous proofs and error information with structured labels containing the error interpretation from LLM and the corresponding correct proofs. 166

7.2 Examples of the four mutation error types with illustrations of their generation models. 167
7.3 A template of our finetuning data format. 168
7.4 Statistics based on mutation type. For each dataset, we report the total number of erroneous proofs generated and detail the distribution of specific mutation types: Line Mutation Error (LME), Multi-line Mutation Error (MLME), Tactic Mutation Error (TME), and Theorem Mutation Error (THME). The percentages denote the proportion of each mutation type within its respective dataset. 169
8.1 ArXiv papers using Lean for formalization, by year. Data from the arXiv metadata snapshot (March 2026), filtered for references to the Lean theorem prover. 172
8.2 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 8.2.3. Here `Torus3` is `Fin 3 → AddCircle 1`, `periodicLift` lifts torus functions to $\mathbb{R}^3$ for differentiation, and `equilibriumMaxwellian` is the Maxwellian $\rho_{\text{ion}}/(2\pi T)^{3/2}\exp(-|v|^2/2T)$. 191
8.3 Proof dependency graph, generated with LeanBlueprint [135]. 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. 192
8.4 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). 194
8.5 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. 195

8.6 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. 196

8.7 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. 197

8.8 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. 198

8.9 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). . . . 199

8.10 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. 200

8.11 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 (∼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. 201

8.12 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 ∼3K lines of dead code, redundant lemmas, and unnecessary heartbeat overrides). Total: +21,783 added, −13,478 deleted, net +8,305 lines. 202

LIST OF TABLES

Table Number Page

2.1 Weibel instability: L^2 distance of the v_2-marginal to $\mathcal{N}(0, T_\infty)$ at $t_{\text{final}} = 125$ ($n = 10^6$, $d_v = 3$). SBTM relaxes closer to the Maxwellian steady state across all collision frequencies. 36

2.2 Runtime and memory for the two-stream benchmark ($M = 100$, $\Delta t = 0.05$, $d_v = 3$, $t_{\text{final}} = 50$). 36

4.1 KL divergence (↓) between the sample and the target $\pi = \mathcal{N}(0, 1)$, using time step 0.002 and final time 2.5. SBTM exhibits better sample efficiency, likely due to determinism. 82

4.2 KL divergence (↓) between the sample and the target $\pi = \frac{1}{4}\mathcal{N}(-2, 1) + \frac{3}{4}\mathcal{N}(2, 1)$, using time step 0.01, and final time 10. Top: non-annealed. Bottom: 100 particles. 85

5.1 Relative MSE of affine equivariance error, averaged over 50 trials. 96

5.2 MSE of score estimation on the 2D Laplace distribution. 103

5.3 MSE of score estimation on the 2D Student-t distribution ($\nu = 3$). 104

6.1 Distribution of theorem types in our dataset. 126

6.2 Results on the validation set, comparing embedder performance against existing literature-search tools. Values are reported as theorem-level / paper-level. Google Search and arXiv search cannot return specific theorem statements, so only paper-level results are reported. Our corpus was restricted to arXiv papers for evaluation. 134

6.3 Comparing context-window size in retrieval performance. Embedded with prompt using Qwen3 8B [212]. Slogans generated with DeepSeek V3 [114]. . 137

6.4 Comparing LLM slogans in retrieval performance. Embedded with prompt using Qwen3 8B [212]. Body + Abstract. 138

6.5 Comparing prompting performance. Slogans generated by DeepSeek V3.1 [114] on the Body+Abstract context window. 138

6.6 Extended results comparing context-window size in retrieval performance across six embedders. Embedded without task instructions. 140

6.7 Extended results comparing LLM slogans in retrieval performance across six embedders. Embedded without task instructions. Body+Abstract. 150

7.1 Optimization and training hyperparameters used in all experiments. 158

7.2 Statistics of correct proofs after filtering. We report the number of raw and filtered proofs for three source datasets, along with the average proof length, average number of `have` statements per proof, and number of proofs containing at least one `have` statement. Across all datasets, we observe significant variation in proof length and `have` statement usage, with proofs sourced from human annotators in Numina's dataset demonstrating the most complexity in terms of length and `have` statements. 159

7.3 Number of erroneous proofs by split. 161

7.4 Basic dataset statistics by split. 161

7.5 Statement source distribution by split. 162

7.6 Proof repair accuracy by model and error type when training jointly for repair and explanation. Results outside the parentheses correspond to models jointly finetuned on all error types and evaluated in each category. Parenthesized values denote models finetuned separately on a single error type and evaluated on that same type. Section 7.5.3 analyzes the effect of removing explanations. 163

7.7 Pass@1 on error types matching the training regime. Δ denotes the change from w/o exp (color-coded). 165

8.1 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. . 171

8.2 Project timeline and sorry count evolution. 177

8.3 Project statistics. The 229 human prompts exclude bare slash commands like `/babysit`; all are archived with timestamps. The mathematician estimates ∼50 hours of active supervision over 10 days. 193

ACKNOWLEDGMENTS

I am thankful to

- Jingwei Hu for teaching me everything I know about plasma simulation and for the many hours of weekly meetings.

- Jarod Alper for learning and teaching Lean with me since 2022, for advocating for me and for believing that I can lead the UW Math AI Lab when I did not believe in myself.

- Julia Pevtsova for talking with me on my PhD visit day, teaching my algebra sequence, leading my reading group on Homological Algebra, and giving me advice and advocating for me throughout my PhD journey.

- Tatiana Toro for teaching my analysis sequence, advocating for me throughout my PhD journey and supporting me with RAship despite not being my advisor.

- Max Lieblich for supporting the UW Math AI lab and supporting me with RAship despite not being my advisor.

- Leo Mayer, Giovanni Inchiostro, Michael Zeng, Dhruv Bhatia and Eric Klavins and many grdauate and undergraduate students for making the Math AI Lab a success.

DEDICATION

For my mother Nadya and my wife Eva – my two biggest supporters and most trusted advisors.

Chapter 1
INTRODUCTION

For thousands of years mathematical intelligence has been prohibitively expensive for most of humanity. With the rise of Artificial Intelligence (AI), intelligence is becoming commodified, similar to electricity, at the price of a few dollars per million tokens. Even tasks that once required years of specialized training, such as numerical simulation of physical systems and the discovery and verification of mathematical proofs, are now performed in part by general-purpose neural networks.

Yet two structural problems shape the trajectory of this revolution. The first is *energy*. Modern AI consumes electricity at industrial scale, and demand for AI compute is projected to outpace the growth of conventional generation. Without a new source of cheap, abundant, low-carbon energy, the commodification of intelligence will collide with the physical limits of the electric grid. The most ambitious candidate for such a source is controlled nuclear fusion, whose realization depends on our ability to simulate, predict, and control the behavior of plasmas: hot, dense, electromagnetically coupled gases of charged particles whose dynamics span many orders of magnitude in space, time, and energy.

The second is *verification*. Even the most capable large language models (LLMs) remain fundamentally stochastic: they produce fluent, confident reasoning that may be subtly or grossly wrong. In open-ended creative work this is a tolerable defect; in mathematics, science, and engineering it is not. The natural antidote is formal verification, in which definitions, theorem statements, and proofs are written in a language whose correctness is checked by a small trusted kernel. Modern proof assistants such as Lean 4 provide such a foundation, but formalization has historically been slow, brittle, and expertise-intensive.

Fortunately, AI can itself be used to tackle both of these challenges. In this dissertation I present several approaches to using machine learning for plasma simulation, which promises to unlock a path to abundant clean energy via nuclear fusion. I also present several applications of AI to mathematical formalization, which may unlock a path to verified AI, in the mathematical and coding domains.

This dissertation is based on the following eight papers:

- Vasily Ilin, Jingwei Hu, and Zhenfu Wang, "Transport based particle methods for the Fokker-Planck-Landau equation" [91]. Published in *Communications in Mathematical Sciences*.

- Vasily Ilin, "Stability of the spatially homogeneous Landau equation in relative entropy and applications to score-based numerical methods" [87]. Preprint; submitted.

- Vasily Ilin and Jingwei Hu, "A Neural Score-Based Particle Method for the Vlasov-Maxwell-Landau System" [89]. Presented at an ICLR Workshop, submitted to Journal.

- Vasily Ilin, Peter Sushko, and Jingwei Hu, "Score-based deterministic density sampling" [93]. Published in *Communications on Pure and Applied Analysis*.

- Vasily Ilin and Peter Sushko, "DiScoFormer: Plug-In Density and Score Estimation with Transformers" [92]. Published at ICML 2026, spotlight (top 2.2%).

- Luke Alexander, Eric Leonen, Sophie Szeto, Artemii Remizov, Ignacio Tejeda, Jarod Alper, Giovanni Inchiostro, and Vasily Ilin, "Semantic Search over 9 Million Mathematical Theorems" [5]. Workshop paper at the *LLM Reasoning Workshop* at ICLR 2026.

- Evan Wang, Simon Chess, Daniel Lee, Siyuan Ge, Ajit Mallavarapu, Jarod Alper, and Vasily Ilin, "Learning to Repair Lean Proofs from Compiler Feedback" [188]. Workshop

paper at *VerifAI-2: The Second Workshop on AI Verification in the Wild* at ICLR 2026.

- Vasily Ilin, "Semi-Autonomous Formalization of the Vlasov-Maxwell-Landau Equilibrium" [88]. Workshop paper at *AI, Proof and Verification (AIPV 2026)*.

1.1 Kinetic equations for collisional plasma

A plasma is a gas of charged particles—electrons, ions, and possibly neutral species—whose collective electromagnetic interactions give rise to rich and complex dynamics [35]. Understanding and predicting the behavior of plasmas is essential for applications ranging from nuclear fusion energy to space weather forecasting. Achieving thermonuclear fusion requires heating the plasma until collisions are sufficiently frequent and energetic to overcome the Coulomb barrier, so collisional effects are essential in this setting. The kinetic description of a plasma tracks the evolution of a particle distribution function $f(t, x, v)$, representing the density of particles at position $x \in \mathbb{R}^3$ with velocity $v \in \mathbb{R}^3$ at time t.

A first-principles model for collisional plasma is the *Vlasov-Maxwell-Landau* (VML) system:

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = \nu\, Q(f, f), \tag{1.1}$$

$$\frac{\partial E}{\partial t} = \nabla_x \times B - J, \qquad \frac{\partial B}{\partial t} = -\nabla_x \times E,$$

$$\nabla_x \cdot E = \rho - \rho_{\text{ion}}, \qquad \nabla_x \cdot B = 0. \tag{1.2}$$

where $\rho(t, x) = \int_{\mathbb{R}^3} f(t, x, v)\, dv$ and $J(t, x) = \int_{\mathbb{R}^3} v f(t, x, v)\, dv$ are the charge and current densities, ρ_{ion} is a neutralizing ion background, and $\nu \geq 0$ is the collision frequency. The collision term is the *Landau collision operator* [106], which models the cumulative effect of many small-angle Coulomb scattering events:

$$Q(f, f)(v) = \nabla_v \cdot \int_{\mathbb{R}^3} A(v - w)\big[f(w)\nabla_v f(v) - f(v)\nabla_w f(w)\big]\, dw, \tag{1.3}$$

where $A(z) = |z|^{\gamma+2}(|z|^2 I - z \otimes z)$ is the collision kernel. The exponent γ parameterizes the strength of the interaction: $\gamma = 0$ corresponds to Maxwellian molecules, while the physically relevant Coulomb case has $\gamma = -3$.

Two important reductions of the VML system are worth highlighting. In the absence of collisions ($\nu = 0$), one recovers the *Vlasov equation*:

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = 0, \tag{1.4}$$

which captures large-scale collective phenomena such as Landau damping, two-stream instabilities, and Weibel instabilities, but neglects the relaxation toward thermodynamic equilibrium induced by collisions. If the plasma is spatially uniform, the VML system reduces to the spatially homogeneous Landau equation

$$\frac{\partial f}{\partial t} = Q(f, f) \tag{1.5}$$

which models a plasma whose spatial distribution is uniform but whose velocity distribution evolves under collisions.

The Landau collision operator possesses fundamental structural properties that numerical methods should aim to preserve. It conserves mass, momentum, and kinetic energy, and it satisfies Boltzmann's H-theorem: the entropy $H(f) = \int f \log f \, dv$ is non-increasing, with equality if and only if f is a Maxwellian distribution [184, 52]. These properties ensure that solutions relax to thermodynamic equilibrium over time, a physical requirement for any faithful simulation.

Computational challenges. The numerical solution of the Landau and VML equations presents severe computational challenges. The distribution function $f(t, x, v)$ lives in six-dimensional phase space ($d = 3$ spatial $+3$ velocity dimensions), making grid-based methods prohibitively expensive due to the curse of dimensionality. Particle methods, which represent f as a collection of n computational particles $\{X_i\}_{i=1}^n$ and evolve them according to the underlying dynamics, offer a dimension-agnostic alternative. The classical *blob method*

[23] approximates the velocity distribution and its derivatives via kernel density estimation (KDE), yielding a deterministic particle scheme. However, the blob method suffers from two limitations: it scales quadratically ($O(n^2)$) in the number of particles, and KDE-based score estimates degrade rapidly in high dimensions, producing unphysical artifacts such as sharp cutoffs in the velocity tails.

1.2 Score-based transport modeling

A central methodology powering Parts I and II of this thesis is *score-based transport modeling* (SBTM), which replaces kernel density estimation with a neural network approximation of the *score function* $\nabla \log f$. The score is the key quantity linking the Landau equation to a transport formulation: the Landau collision operator can be rewritten as a continuity equation

$$\frac{\partial f}{\partial t} + \nabla_v \cdot (f\, U[f]) = 0, \qquad U[f](v) = -\int_{\mathbb{R}^3} A(v-w)\big[\nabla_v \log f(v) - \nabla_w \log f(w)\big] f(w)\, dw, \tag{1.6}$$

where the velocity field $U[f]$ depends on f only through its score. Once the score is known, particles can be advected deterministically:

$$\frac{dX_i}{dt} = U[f](X_i) \approx -\frac{1}{n}\sum_{j=1}^{n} A(X_i - X_j)\big[s_\theta(X_i) - s_\theta(X_j)\big], \tag{1.7}$$

where $s_\theta \approx \nabla \log f$ is a neural network trained to fit the unknown score in least squares:

$$\min_\theta \mathbb{E}_{X\sim f}\big[\|s_\theta(X) - \nabla \log f(X)\|^2\big].$$

Expanding the square and integrating by parts in the cross term,

$$\mathbb{E}_{X\sim f}\big[s_\theta(X)\cdot \nabla \log f(X)\big] = \int_{\mathbb{R}^3} s_\theta(x)\cdot \nabla f(x)\, dx = -\mathbb{E}_{X\sim f}\big[\nabla \cdot s_\theta(X)\big],$$

shows that, up to an additive constant independent of θ, this is equivalent to the *implicit score-matching loss* [85]

$$\mathcal{L}(s_\theta) = \mathbb{E}_{X\sim f}\big[\|s_\theta(X)\|^2 + 2\,\nabla \cdot s_\theta(X)\big]. \tag{1.8}$$

This loss is minimized at $s_\theta = \nabla \log f$ and can be evaluated using only particle positions, without access to the density f itself. This reduces modeling the Landau collision operator to solving the above optimization problem.

Score-based methods originated in the statistics literature [85] and rose to prominence through their role in diffusion generative models [163], where they enable sampling from complex high-dimensional distributions. The idea of using learned score functions to solve Fokker-Planck equations with deterministic particle dynamics was introduced by Boffi and Vanden-Eijnden [20] for solving the Fokker-Planck equation. This dissertation extends this approach to the Landau and Vlasov-Maxwell-Landau equations, as well as using it for sampling via the Fokker-Planck equation.

Advantages of the score-based approach. SBTM offers two key advantages over classical blob methods. First, the score network is trained via a MSE-like loss rather than pairwise kernel density estimation, so score estimation scales linearly in n rather than quadratically. Second, neural networks provide smooth, global approximations of the score that remain accurate in low-density regions where KDE fails, eliminating the unphysical tail artifacts that plague blob methods. This makes SBTM both faster and more accurate than the previous state-of-the-art.

Beyond collisional plasma, SBTM also appears in this dissertation as a deterministic sampling method: chapter 4 shows its application to approximate the Wasserstein gradient flow on relative entropy for sampling from unnormalized densities [93]. We also propose a complementary amortized approach to score estimation, DiScoFormer [92], a pretrained Transformer that predicts density and score information in a single forward pass and can replace the online score-training stage of SBTM.

1.3 AI for mathematics

The third part of this dissertation turns from scientific computing to a different application of AI: mathematical discovery, proof repair, and formalization.

A *proof assistant* is software in which mathematical definitions, theorem statements, and proofs are written in a formal language whose correctness is verified by a small trusted kernel. To say that mathematics is *formalized* means that every object, hypothesis, and logical step has been translated into machine-checkable code. Nothing is left implicit: every use of a prior theorem must match an exact formal statement, and every omitted "obvious" step in an informal proof must be made explicit. In this dissertation the formal system is Lean 4 [50], together with its large community library Mathlib [173].

This extra precision is valuable because it turns correctness into a property that can be checked mechanically rather than socially. Formalized mathematics can be verified without ambiguity, reused as dependable infrastructure for later developments, and accessed by software systems for search, proof synthesis, and verification. More broadly, formalization creates a path from informal mathematics to verified computational artifacts, including verified algorithms and code generation.

The difficulty is that formalization demands exactly the details that mathematicians usually suppress. Definitions that are interchangeable on paper may not line up definitionally in a proof assistant; routine arguments must be decomposed into many small justified steps; and simply finding the right prior theorem or library lemma can itself be a major obstacle. For this reason, formalization has traditionally been slow, brittle, and expertise-intensive.

At the same time, formalization addresses one of the biggest bottlenecks in using AI for mathematics: hallucinations. A proof assistant does not accept plausible-sounding arguments; every definition, statement, and proof step must be checked mechanically and deterministically. Conversely, AI addresses the biggest bottleneck of formalization itself: it is tedious and time-intensive. AI systems can search large libraries, translate informal reasoning into formal code, and iteratively repair failed proofs from compiler feedback. In this dissertation I show that these two technologies are complementary: formalization makes AI-generated mathematics reliable, and AI makes formalization practical at research scale.

This dissertation contributes to three aspects of using AI for mathematics and formalization. First, the problem of *discovery*: given an informal mathematical question, how can one

find relevant theorem statements across the vast body of mathematical literature? Second, the problem of *repair*: when a formalized proof breaks—due to library updates, refactoring, or initial errors—how can AI systems learn to diagnose and fix the failure using compiler feedback? Third, the problem of *formalization* itself: can the full pipeline from conjecture to machine-verified proof be carried out semi-autonomously, with AI systems handling the code generation while a mathematician provides high-level guidance?

1.4 Contributions and outline

This dissertation is organized into three parts comprising seven Chapters. It synthesizes eight papers into a thematic exposition: the plasma contributions are grouped into one Chapter on numerical methods and one on theoretical results, followed by a Chapter on score-based methods for sampling and density estimation and a Chapter on AI for mathematics. I summarize the contributions of each Chapter below.

Part I: Collisional Plasma Kinetics

Chapter 2: Score-based particle methods for collisional plasma. In this Chapter, I turn the score-based viewpoint introduced earlier into a practical numerical method for collisional plasma, first in the homogeneous Landau setting and then in the full Vlasov-Maxwell-Landau system. My goal here is to show that score-based particle methods are not just a conceptual reformulation but a viable replacement for kernel-based estimators in realistic plasma simulation. This provides the computational entry point to Part I. The numerical behavior established here motivates the analytical questions taken up in Chapter 3: why the method preserves physical structure, when it converges, and what equilibrium it should approach. This Chapter is based on two papers: [91, 89].

Chapter 3: Theory for collisional plasma and score-based numerics. After developing the numerical method in Chapter 2, I use this Chapter to explain why it should be trusted. Here I assemble the theoretical contributions: stability of the homogeneous

Landau equation in relative entropy, error control for score-based approximations, and the characterization of equilibrium for the Vlasov-Maxwell-Landau system. The equilibrium theorem formalized later in Chapter 8 originates here. This Chapter is based on three papers: [91, 87, 89].

Part II: Score Estimation and Deterministic Sampling

Chapter 4: Score-based deterministic density sampling. In this Chapter, I show that the same technique of online learning of the score function can be used for deterministic sampling, via the Fokker-Planck equation. The methodological point is that the same mechanism underlying the plasma solvers belongs to a broader family of gradient-flow algorithms rather than to one specialized kinetic model. This also sets up the natural next question addressed in Chapter 5: if score-based transport is useful across problems and distributions, what other ways of estimating the score are there? This Chapter is based on [93].

Chapter 5: DiScoFormer: Plug-in density and score estimation with Transformers. While Chapters 2 and 4 treat score estimation as an inner loop, in this Chapter I make score estimation itself the main object of study. I connect classical nonparametric statistics with modern Transformer architectures by showing that attention can be understood as a learned generalization of KDE, while also moving from per-instance training toward amortized, reusable estimators. This shows a connection between kernel-based and neural methods. This Chapter is based on [92].

Part III: AI for Mathematics and Formalization

Chapter 6: Semantic search over mathematical theorems. I open Part III from the side of discovery rather than proof construction. We built a search engine over all theorems on arXiv and demonstrated its usefulness both for research mathematicians and mathematical AI agents. The resulting search engine outperforms Google Search and LLM-based search while being fast and hallucination-free. This Chapter is based on [5].

Chapter 7: Learning to repair Lean proofs from compiler feedback. If theorem search addresses finding the right statement, in this Chapter I address what happens after a formal proof attempt fails. We study repair as a first-class task, using compiler feedback not merely as a debugging aid but as supervision for learning how to diagnose and correct formal errors. Together with Chapter 6, this supplies another core ingredient of a practical agentic workflow for mathematical formalization. This Chapter is based on [188].

Chapter 8: Semi-autonomous formalization of the Vlasov-Maxwell-Landau equilibrium. In this final Chapter, formalization meets plasma simulation. I take a theorem from the plasma theory part of the dissertation, first conjectured using the numerical evidence, and carry it through an AI-assisted formalization pipeline, producing a complete formalization in Lean 4. Its importance is not only the theorem being formalized, but the demonstration that discovery, proof generation, repair, and verification can be composed into a single research workflow. As a case study in end-to-end AI-assisted mathematics, it also exposes the concrete failure modes that must be understood if such workflows are to become reliable. This Chapter is based on [88].

Chapter 2

SCORE-BASED PARTICLE METHODS FOR COLLISIONAL PLASMA

This chapter develops the score-based particle method for collisional plasma in two stages. Section 2.1 treats the spatially homogeneous Landau equation, which is the simplest setting in which the score-based approach can replace the kernel-based blob estimator. Building on that foundation, Section 2.2 extends the method to the full Vlasov–Maxwell–Landau (VML) system, in which the velocity-space collision operator is coupled to spatial transport and to self-consistent electromagnetic fields. The relevant background on the Landau collision operator, the VML system, and score-based transport modeling has been introduced in Chapter 1; here we focus on the algorithm, its conservation properties, and its numerical behavior. The remaining theoretical guarantees – KL stability of the Landau equation and the characterization of the VML equilibrium – are collected in Chapter 3.

2.1 Homogeneous Landau equation

We begin with the spatially homogeneous Landau equation

$$\partial_t f(t,v) = \nabla_v \cdot \int_{\mathbb{R}^d} A(v-w)\big[f(w)\,\nabla_v f(v) - f(v)\,\nabla_w f(w)\big]\,\mathrm{d}w, \qquad v \in \mathbb{R}^d, \tag{2.1}$$

with the collision kernel

$$A(z) = |z|^{\gamma+2}\big(I_d - zz^\top/|z|^2\big) = |z|^\gamma\big(|z|^2 I_d - z\otimes z\big). \tag{2.2}$$

Here $f(t,v) \geq 0$ is the velocity distribution, normalized to a probability density $\int f(t,v)\,\mathrm{d}v \equiv 1$. We treat both the Maxwellian case $\gamma = 0$, which serves as the analytically tractable benchmark, and the physically relevant Coulomb case $\gamma = -3$ in dimension $d = 3$.

2.1.1 The Landau equation as a continuity equation

The starting point of the particle method is to rewrite (2.1) in continuity form

$$\partial_t f + \nabla_v \cdot (f\, U[f]) = 0, \qquad U[f](v) = -\int_{\mathbb{R}^d} A(v-w)\big[\nabla \log f(v) - \nabla \log f(w)\big] f(w)\, \mathrm{d}w. \tag{2.3}$$

This identity follows by carrying the inner divergence in (2.1) outside the integral and using $\nabla_v f(v) = f(v)\, \nabla \log f(v)$. Equation (2.3) exposes the structural feature on which the rest of this chapter rests: the velocity field $U[f]$ depends on the density only through the score $\nabla \log f$. Provided we have access to the score, particles can be advected deterministically along the flow generated by $U[f]$, and the density does not need to be reconstructed.

Remark 2.1. It is not a lucky coincidence that the Landau equation and many other PDEs in physics admit the form of a continuity equation. Indeed, as studied in the field of Optimal Transport, continuity equations describe continuous motion of mass, where no teleportation of mass is allowed. Since conservation of mass is a fundamental physical law, many PDEs describing "real-world" motion of matter admit the continuity equation form.

A weak solution of the continuity equation

$$\partial_t f + \nabla_v \cdot (f\, U) = 0 \tag{2.4}$$

can be obtained as an empirical measure

$$f(t,v) \approx \frac{1}{n} \sum_{i=1}^{n} \delta_{V_i(t)}(v), \qquad \frac{\mathrm{d}V_i}{\mathrm{d}t} = U(t, V_i(t)), \tag{2.5}$$

in the sense that for any test function $\phi \in C_c^\infty(\mathbb{R}^d)$, $\frac{\mathrm{d}}{\mathrm{d}t} \int \phi \, \mathrm{d}f_t = \int \nabla\phi \cdot U \, \mathrm{d}f_t$ holds along the particle dynamics. See [151] for the classical theory of weak solutions of continuity equations.

The numerical task is therefore to approximate the velocity field $U[f]$ in (2.3) on the empirical measure (2.5). Substituting $f \approx \frac{1}{n}\sum_j \delta_{V_j}$ yields the pairwise sum

$$U[f](V_i) \approx -\frac{1}{n} \sum_{j=1}^{n} A(V_i - V_j)\big[s(V_i) - s(V_j)\big], \tag{2.6}$$

where $s : \mathbb{R}^d \to \mathbb{R}^d$ is a placeholder for an estimator of the score $\nabla \log f$. The classical *blob method* of [24] takes s to be the gradient of a kernel density estimate; we take s to be a neural network trained by score matching. Both choices are special cases of the same particle scheme, and both share the conservation properties established in Section 2.1.2.

2.1.2 Particle method and conservation

For any time-dependent vector field $s_t : \mathbb{R}^d \to \mathbb{R}^d$, define the approximate velocity field

$$U[f, s_t](v) := -\int_{\mathbb{R}^d} A(v-w)\big[s_t(v) - s_t(w)\big] f(w)\,\mathrm{d}w \tag{2.7}$$

and the corresponding continuity equation

$$\partial_t f + \nabla_v \cdot (f\, U[f, s_t]) = 0. \tag{2.8}$$

The pleasant feature of (2.8) is that the structural properties of the Landau collision operator – conservation of mass, momentum, and kinetic energy, and dissipation of an entropy-like quantity – are preserved for *every* choice of s_t, not merely for $s_t = \nabla \log f$. This is the content of the following proposition, which makes the particle method robust to errors in the score approximation.

Proposition 2.2 (Conservation properties under any score approximation)**.** *Let $s_t(v)$ be any sufficiently regular time-dependent vector field, and let f solve* (2.8) *with initial datum f_0. Then:*

(i) The continuous solution f conserves mass, momentum, and kinetic energy:

$$\int f_t\,\mathrm{d}v = \int f_0\,\mathrm{d}v, \quad \int v\, f_t\,\mathrm{d}v = \int v\, f_0\,\mathrm{d}v, \quad \int |v|^2 f_t\,\mathrm{d}v = \int |v|^2 f_0\,\mathrm{d}v,$$

and the estimated entropy $\mathcal{E}_t := \int_0^t \int_{\mathbb{R}^d} s_\tau(v) \cdot U[f, s_\tau](v)\, f_\tau(v)\,\mathrm{d}v\,\mathrm{d}\tau$ is non-increasing.

(ii) The particle solution $f_t = \frac{1}{n}\sum_i \delta_{V_i(t)}$ with $\frac{\mathrm{d}V_i}{\mathrm{d}t} = U[f, s_t](V_i)$ conserves mass, momentum, and kinetic energy:

$$\frac{1}{n}\sum_i V_i(t) = \frac{1}{n}\sum_i V_i(0), \qquad \frac{1}{n}\sum_i |V_i(t)|^2 = \frac{1}{n}\sum_i |V_i(0)|^2,$$

and the estimated entropy is non-increasing.

(iii) The forward-Euler discretization $V_i^{k+1} = V_i^k + \Delta t\, U[f^k, s_{t_k}](V_i^k)$ *conserves mass and momentum exactly, and the discrete estimated entropy* $\mathcal{E}^k = \frac{\Delta t}{n}\sum_{j=1}^k \sum_i s(V_i^j)\cdot U(V_i^j)$ *satisfies* $\mathcal{E}^{k+1} \le \mathcal{E}^k$.

Proof. For (i) and (ii) it suffices to show that any weak solution of (2.8) conserves the relevant moments and has non-positive entropy production. Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a test function. By the definition of a weak solution,

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\int \phi \,\mathrm{d}f_t &= \int \nabla\phi(v)\cdot U[f,s_t](v)\,\mathrm{d}f_t(v)\\ &= -\iint \nabla\phi(v)^\top A(v-w)\big[s_t(v)-s_t(w)\big]\,\mathrm{d}f_t(w)\,\mathrm{d}f_t(v)\\ &= -\frac{1}{2}\iint \big[\nabla\phi(v)-\nabla\phi(w)\big]^\top A(v-w)\big[s_t(v)-s_t(w)\big]\,\mathrm{d}f_t(w)\,\mathrm{d}f_t(v),\end{aligned}$$

where the last line follows from the symmetry $A(z) = A(-z)$ by exchanging the roles of v and w. Taking $\phi(v) = 1, v_\alpha, |v|^2$ and using $A(z)z = 0$ gives conservation of mass, momentum, and energy. Taking $\nabla\phi = s_t$ gives

$$\int s_t(v)\cdot U[f,s_t](v)\,\mathrm{d}f_t(v) = -\frac{1}{2}\iint \big[s_t(v)-s_t(w)\big]^\top A(v-w)\big[s_t(v)-s_t(w)\big]\,\mathrm{d}f_t(w)\,\mathrm{d}f_t(v) \le 0,$$

since A is positive semidefinite, which gives non-positivity of the entropy production rate.

For the time-discrete particle solution in (iii), conservation of momentum follows from a direct computation,

$$\frac{1}{n}\sum_i V_i^{k+1} = \frac{1}{n}\sum_i V_i^k - \frac{\Delta t}{n^2}\sum_{i,j} A(V_i^k - V_j^k)\big[s(V_i^k) - s(V_j^k)\big] = \frac{1}{n}\sum_i V_i^k,$$

where the double sum vanishes after symmetrizing in $i \leftrightarrow j$. Mass conservation is immediate since each particle carries weight $1/n$. The discrete entropy estimate

$$\mathcal{E}^{k+1} - \mathcal{E}^k = -\frac{\Delta t}{2n^2}\sum_{i,j}\big[s(V_i^{k+1}) - s(V_j^{k+1})\big]^\top A(V_i^{k+1} - V_j^{k+1})\big[s(V_i^{k+1}) - s(V_j^{k+1})\big] \le 0$$

follows by the same symmetrization and the positive semidefiniteness of A. □

Remark 2.3. Theorem 2.2 is independent of how s_t is obtained. The blob method of [24, 23], the kernel RKHS estimator of [133], and the score-matching neural network we develop next all fall under the same umbrella: they fix particle positions and produce a vector field s_t, and the resulting particle trajectories conserve mass, momentum, and energy and dissipate estimated entropy. This robustness is essential when s_t is a learned approximation that is never exactly equal to $\nabla \log f$.

2.1.3 Score-based transport modeling

We now address the approximation of the score itself. The key observation, due to [85, 20, 133], is that the score-matching loss

$$\int_{\mathbb{R}^d} \big|s - \nabla \log f\big|^2_{A*f} \, f \, \mathrm{d}v := \int_{\mathbb{R}^d} (\nabla \log f - s)^\top (A * f)(\nabla \log f - s) \, f \, \mathrm{d}v \tag{2.9}$$

can be made tractable by integration by parts. Expanding the square and using $\nabla f = f \, \nabla \log f$, we obtain

$$\begin{aligned} \int |\nabla \log f - s|^2_{A*f} \, f \, \mathrm{d}v &= \int \big(|\nabla \log f|^2_{A*f} + |s|^2_{A*f}\big) f \, \mathrm{d}v - 2 \int s^\top (A * f)(\nabla \log f) f \, \mathrm{d}v \\ &= \int \big(|\nabla \log f|^2_{A*f} + |s|^2_{A*f}\big) f \, \mathrm{d}v + 2 \int f \, \nabla \cdot \big((A * f)s\big) \, \mathrm{d}v. \end{aligned} \tag{2.10}$$

The first term in (2.10) is independent of s, so up to an additive constant the weighted score-matching loss equals

$$\int_{\mathbb{R}^d} \Big(|s|^2_{A*f} + 2 \, \nabla \cdot \big((A * f)s\big)\Big) f \, \mathrm{d}v. \tag{2.11}$$

Crucially, (2.11) is well-defined on empirical measures $f = \frac{1}{n} \sum_j \delta_{V_j}$ – the pointwise score $\nabla \log f$ has been eliminated – whereas the original expression (2.9) is not.

Computing $A * f$ at each particle location costs $O(n^2)$, which dominates the per-step cost when many gradient steps are taken. Following [20], we drop the $A * f$ weighting and use the *unweighted* implicit score-matching loss

$$L^{\mathrm{ISM}}(s) = \frac{1}{n} \sum_{i=1}^{n} \Big[|s(V_i)|^2 + 2 \, \nabla \cdot s(V_i)\Big], \tag{2.12}$$

which is also derivable from $\int |s - \nabla \log f|^2 f \, \mathrm{d}v$ and which is minimized at $s = \nabla \log f$. We defer the analysis of the weighted loss to future work. By the law of large numbers, if $\{V_j\}$ are i.i.d. samples from f then $L^{\mathrm{ISM}}(s) \to \mathbb{E}_f\big[|s|^2 + 2\nabla \cdot s\big]$ almost surely as $n \to \infty$.

The divergence $\nabla \cdot s$ in (2.12) can be evaluated either by automatic differentiation, which costs $O(dC_s)$ per particle where C_s is the per-evaluation cost of the network, or via the *denoising trick* of [20]: for $Z \sim \mathcal{N}(0, I_d)$ and small $\alpha > 0$,

$$(2\alpha)^{-1}\, \mathbb{E}[s(v + \alpha Z) \cdot Z - s(v - \alpha Z) \cdot Z] \xrightarrow{\alpha \to 0} \nabla \cdot s(v). \tag{2.13}$$

The denoising estimator costs only two network evaluations per particle and, as observed in [20], also acts as a regularizer that prevents overfitting. We therefore minimize the surrogate loss

$$L^{\mathrm{den}}(s) = \frac{1}{n} \sum_{i=1}^{n} \Big[|s(V_i)|^2 + \alpha^{-1} \big(s(V_i + \alpha Z) - s(V_i - \alpha Z) \big) \cdot Z \Big], \tag{2.14}$$

with Z resampled at each step. The choice of α trades bias against variance: larger α reduces variance at the cost of approximation bias.

We parameterize s as a feedforward neural network s_θ with parameters θ, and minimize $L^{\mathrm{den}}(s_\theta)$ by gradient descent. Universal approximation guarantees that for sufficiently large networks the minimizer recovers $\nabla \log f$. The full algorithm interleaves K gradient steps on L^{den} with one forward-Euler step of the particle ODE.

Algorithm 1 SBTM for the homogeneous Landau equation

Require: Initial density f_0, particle count n, time horizon T, time step Δt, GD steps per time step K, denoising scale α

1: Sample $\{V_i\}_{i=1}^n \sim f_0$ i.i.d.
2: Initialize s_θ by minimizing $\frac{1}{n}\sum_i |s_\theta(V_i) - \nabla \log f_0(V_i)|^2$
3: $t \leftarrow 0$
4: **while** $t < T$ **do**
5: **for** $k = 1, \dots, K$ **do**
6: Draw $Z \sim \mathcal{N}(0, I_d)$
7: $\theta \leftarrow \theta - \eta \nabla_\theta L^{\text{den}}(s_\theta)$
8: **end for**
9: **for** $i = 1, \dots, n$ **do**
10: $V_i \leftarrow V_i - \frac{\Delta t}{n}\sum_{j=1}^n A(V_i - V_j)\big[s_\theta(V_i) - s_\theta(V_j)\big]$
11: **end for**
12: $t \leftarrow t + \Delta t$
13: **end while**

The initialization in line 2 is a least-squares fit to the analytic initial score, which is available because f_0 is a chosen initial condition (typically Gaussian or a Gaussian mixture). The collision force evaluation in line 9 is $O(n^2)$, so the overall complexity is still $O(n^2)$ but with a smaller constant. This is the source of SBTM's runtime advantage over the blob method, which we quantify in Section 2.1.4.

2.1.4 Numerical experiments

We compare SBTM against the blob method of [24] on benchmarks with both Maxwellian and Coulomb collision kernels. The simulation code is implemented in Julia and is publicly available at `https://github.com/Vilin97/GradientFlows.jl`. Experiments were run on the Doppio cluster of the Department of Applied Mathematics at the University of Wash-

ington (two 20-core Intel Xeon E5-2698v4 CPUs at 2.20 GHz, two TITAN X Pascal GPUs).

Hyperparameter choice

The blob method has a single critical hyperparameter, the kernel bandwidth ε, which sets the scale of the kernel density estimate of the score. Both $\varepsilon \to 0$ and $\varepsilon \to \infty$ degenerate the particle velocity to zero, so an interior optimum exists. [24] sets ε to the mesh size; since we use random initialization rather than a grid, we follow Silverman's plug-in rule [41]:

$$\varepsilon = n^{-1/(d+6)} \big(\sigma_1 \sigma_2 \cdots \sigma_d\big)^{1/d}, \tag{2.15}$$

where σ_i are the eigenvalues of the empirical covariance. This choice is asymptotically optimal for estimating ∇f in mean integrated squared error; whether it is optimal for estimating $\nabla \log f = \nabla f / f$ is less clear, but we have not found a better practical alternative.

For SBTM the principal hyperparameters are network architecture, learning rate η, optimizer, activation, GD steps per time step K, and the denoising scale α. After tuning, we use $\eta = 4 \times 10^{-4}$, the Adam optimizer, the `softsign` activation, $K = 25$, and $\alpha = 0.4$. The neural network has one hidden layer of 100 neurons, except for the BKW benchmark which uses two hidden layers because the initial score is nearly singular at the origin.

A poorly chosen learning rate can cause severe underfitting in the early iterations, when the score must change rapidly. Figure 2.1 illustrates the failure mode: with $\eta = 10^{-4}$ and a fixed budget of K GD steps, the implicit score-matching loss never recovers between particle updates, and the score estimate stays far from $\nabla \log f$. An adaptive strategy – continuing GD steps until the implicit loss stops decreasing rather than capping at K – is much more forgiving across learning rates, at the cost of needing the implicit (rather than the cheaper denoising) loss to detect convergence. We use the fixed-K schedule throughout the experiments and defer the adaptive variant to future work.

For density reconstruction (used only for the L^2 error metric, never in the algorithm itself) we use Scott's rule of thumb [153] $H^* = n^{-1/(d+4)}\Sigma$, where Σ is the empirical covariance.

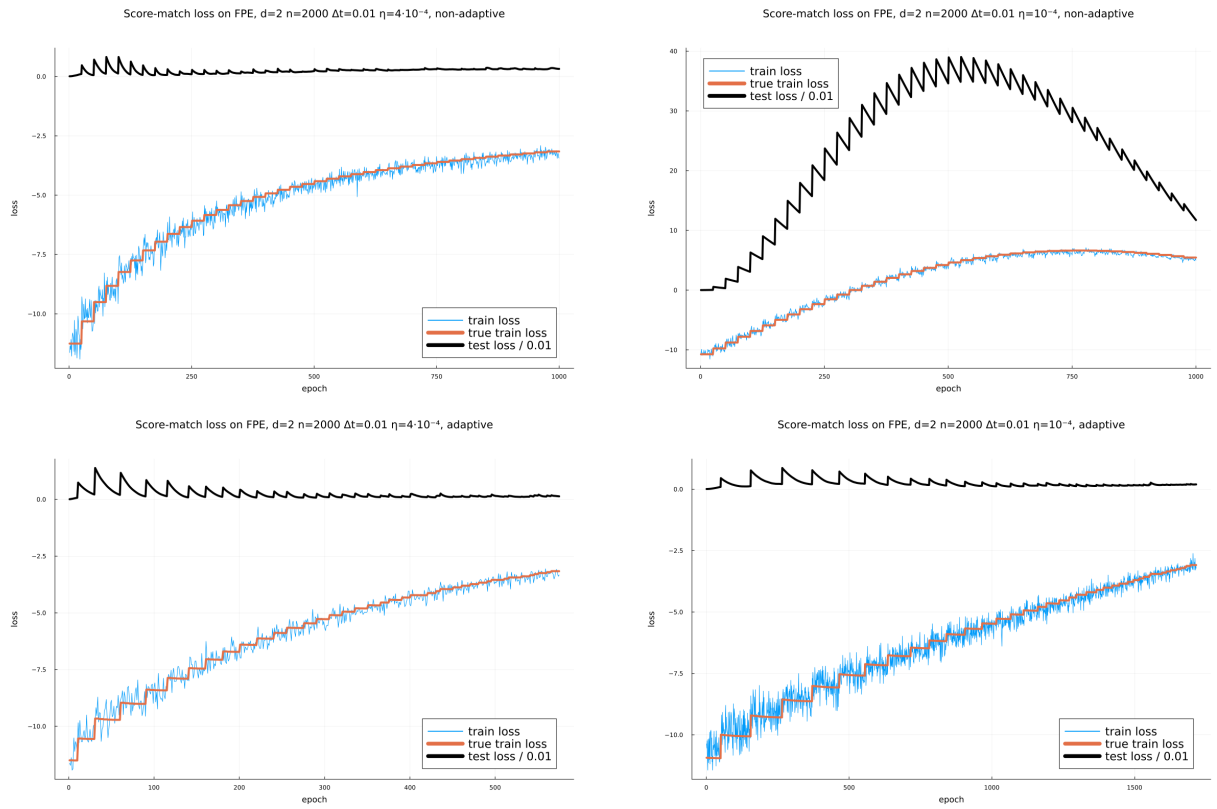


Figure 2.1: Effect of underfitting in the score-matching loss. Top row: non-adaptive schedule (fixed K). Bottom row: adaptive schedule. Left column: $\eta = 4\times10^{-4}$ (good). Right column: $\eta = 10^{-4}$ (too small). The black curve is the true weighted score-matching loss (2.9), the orange curve is the unweighted implicit loss (2.12), and the blue curve is the denoising loss (2.14).

Maxwellian benchmarks

We test the method on three Maxwellian-kernel ($\gamma = 0$) benchmarks: the analytic BKW solution in $d = 3$, an anisotropic Gaussian initial condition in $d = 3$, and the same anisotropic problem in $d = 10$.

BKW solution in dimension $d = 3$. The BKW solution is an isotropic exact solution of the Maxwellian Landau equation derived in [24]. In dimension d with $A(z) = B(|z|^2 I - z \otimes z)$, it reads

$$f_t^*(v) = (2\pi K)^{-d/2} \exp\left(-\frac{|v|^2}{2K}\right) \big(P + Q|v|^2\big), \tag{2.16}$$

with $K = 1 - \exp(-2B(d-1)t)$, $P = \frac{(d+2)K-d}{2K}$, $Q = \frac{1-K}{2K^2}$, and $B = 1/24$. We start at $t_0 = 5.5$, slightly past the singular time $t_0^* = \log(d/2+1)/(2B(d-1)) = 5.497$ for $d = 3$, so that $P(t_0) > 0$. We integrate to $t_{\text{end}} = 9.5$ with $\Delta t = 0.01$.

Figure 2.2 reports six diagnostics: the $(1,1)$ entry of the covariance trajectory, the Frobenius error of the second-moment matrix at t_{end}, the relative score error $\sum_i |\nabla \log f^*(V_i) - s(V_i)|^2 / \sum_i |\nabla \log f^*(V_i)|^2$, the L^2 density error, the conserved-mean error, and the conserved-energy error. Both methods recover the BKW solution well, as expected from the isotropy of the initial data: when f_0 is isotropic the second moments match automatically up to sampling error, and the simpler analysis of Chapter 3 applies. The conserved-mean error is at the level of floating-point round-off, in agreement with Theorem 2.2 (iii); the conserved-energy error scales with Δt from forward-Euler discretization.

Anisotropic Gaussian in $d = 3$ and $d = 10$. The isotropic BKW benchmark masks the issue at the heart of the comparison: the second-moment matrix is preserved trivially under isotropic flow. For a genuine test we initialize from an anisotropic Gaussian

$$\Sigma_{i,j}(0) = \delta_{i,j}\sigma_i, \qquad \sigma_1 = 1.8,\ \sigma_2 = 0.2,\ \sigma_i = 1 \text{ for } i \geq 3, \tag{2.17}$$

chosen so that the trace is d as in the BKW case. The exact second-moment trajectory is known in closed form,

$$\Sigma_{i,j}^*(t) = \Sigma_{i,j}^*(\infty) - \big(\Sigma_{i,j}^*(\infty) - \Sigma_{i,j}^*(0)\big)e^{-4dt}, \qquad \Sigma_{i,j}^*(\infty) = \tfrac{2E}{d}\delta_{i,j}, \tag{2.18}$$

which we use as ground truth. We integrate to $t_{\text{end}} = 4$ with $\Delta t = 0.01$.

Figure 2.3 ($d = 3$) shows that SBTM matches (2.18) accurately with $n = 100$ particles, while the blob method requires substantially more particles to do the same. Using the

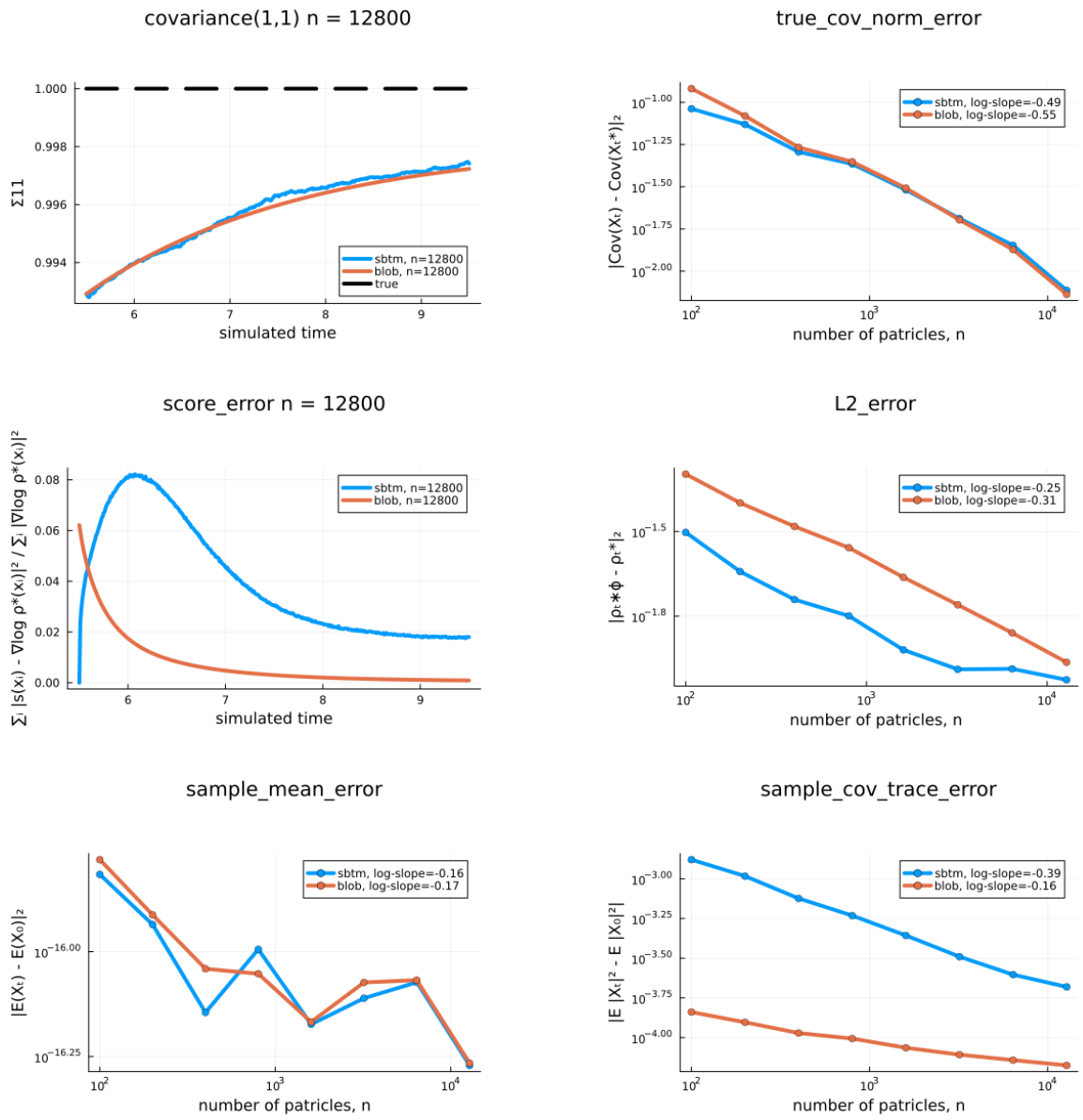


Figure 2.2: BKW solution to the Maxwellian Landau equation in $d = 3$. Both blob and SBTM recover the analytic solution accurately; the conserved-mean error is at the level of floating-point round-off.

$n = 12{,}800$ trajectory of either method as a reference, SBTM with $n = 100$ already tracks the entropy decay rate; the blob method with $n = 100$ shows visible deviations.

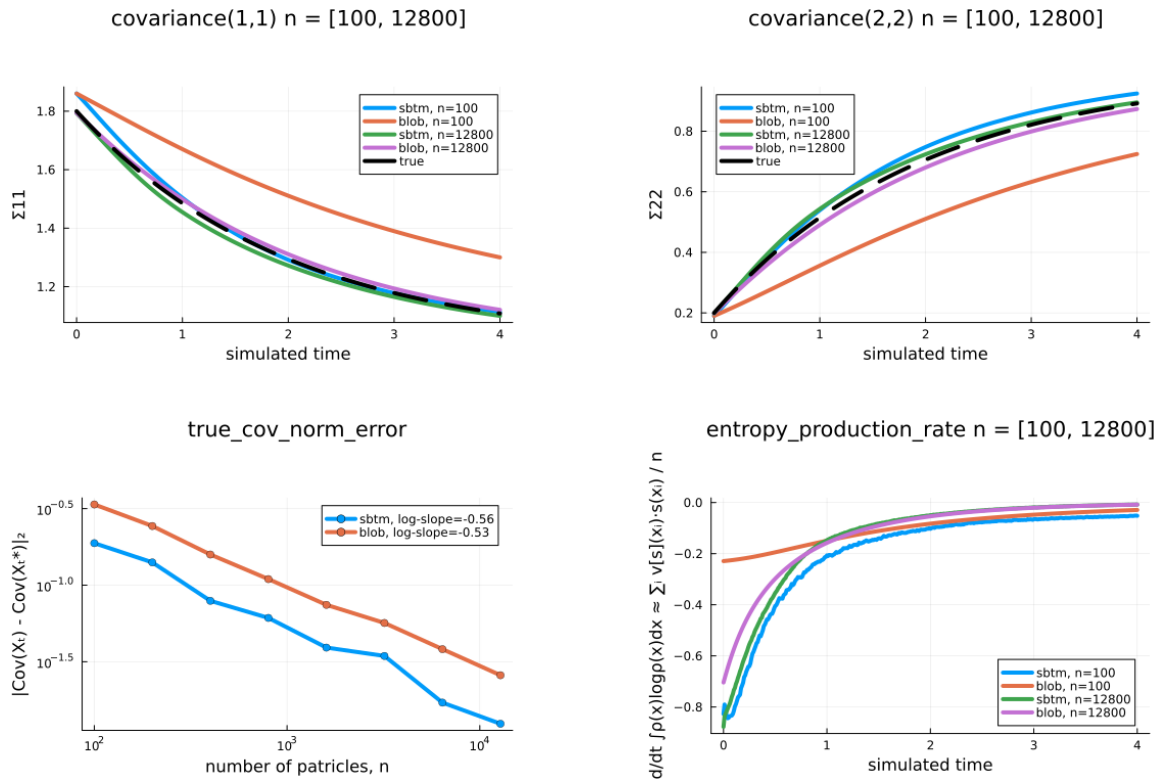


Figure 2.3: Anisotropic Maxwellian Landau equation in $d = 3$. SBTM tracks the analytic covariance trajectory and the entropy decay rate at $n = 100$; the blob method requires more particles for comparable accuracy.

The anisotropic test in $d = 10$ tests robustness to the curse of dimensionality (Figure 2.4). Even at $n = 25{,}600$ the blob method fails to track the analytic covariance trajectory, while SBTM remains accurate. The blob method's KDE-based score estimator degrades in high dimension because the bandwidth required for variance control becomes incompatible with the bandwidth needed to resolve the score; SBTM avoids the issue by using a parametric estimator with a regularization tied to the network architecture rather than to a kernel

bandwidth.

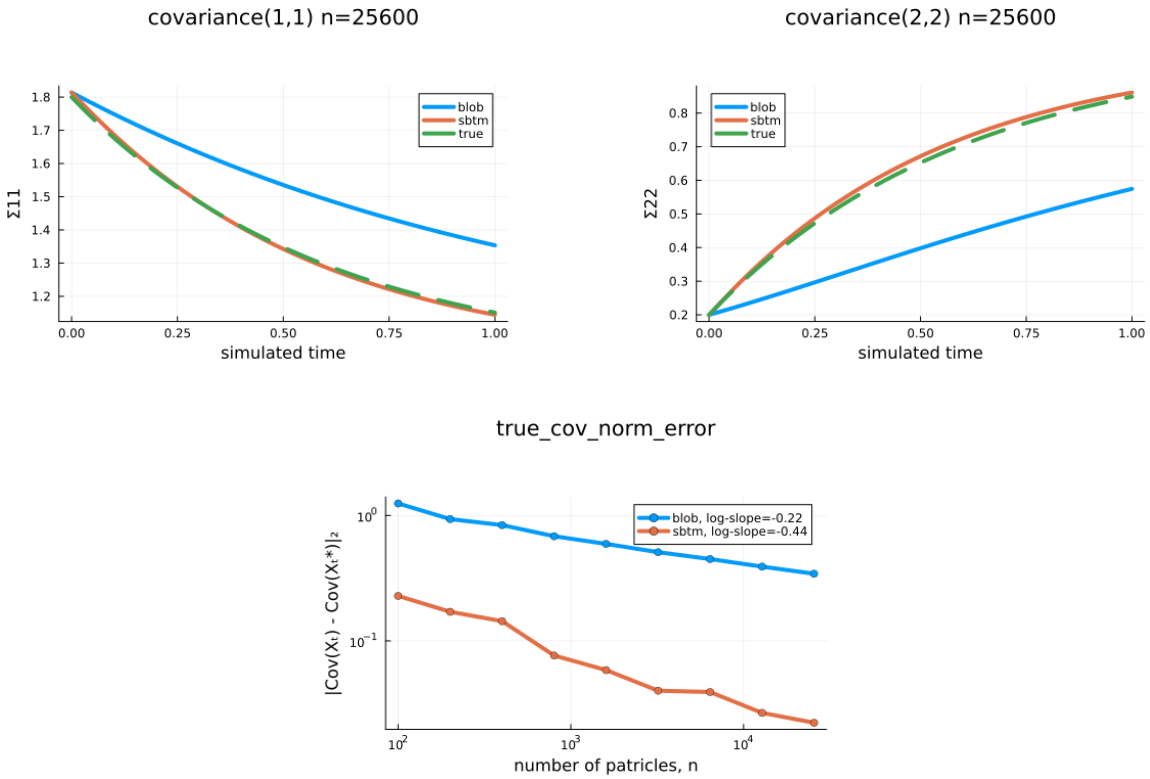


Figure 2.4: Anisotropic Maxwellian Landau equation in $d = 10$. Even at $n = 25{,}600$ the blob method does not track the analytic covariance trajectory, whereas SBTM remains accurate.

Coulomb benchmark

For the Coulomb kernel ($\gamma = -3$, $d = 3$) no closed-form solution is available. We use the same anisotropic Gaussian initial data as in the $d = 3$ Maxwellian case, integrate to $t_{\text{end}} = 300$ with $\Delta t = 1$, and use the high-particle-count trajectories of both methods as a mutual reference: where the two methods agree at large n, the trajectory is reliable. Figure 2.5 shows that SBTM at $n = 100$ matches the $n = 12{,}800$ trajectories in covariance and entropy decay rate much more closely than the blob method does. The dashed line indicates the Maxwellian

steady state, $\Sigma_\infty = I_3$, which both methods approach but neither reaches in finite time. No instabilities were observed even at the relatively large step $\Delta t = 1$.

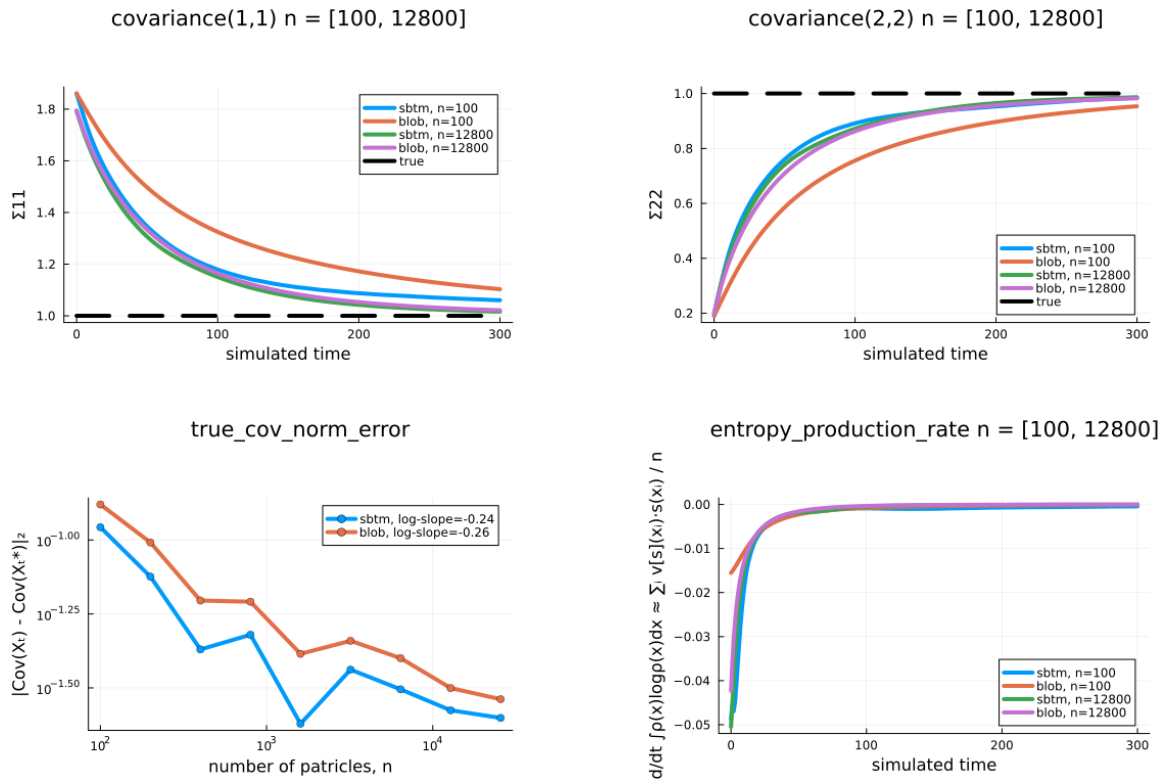


Figure 2.5: Anisotropic Coulomb Landau equation in $d = 3$. SBTM at $n = 100$ matches the $n = 12{,}800$ reference trajectories more accurately than the blob method. The dashed line is the Maxwellian steady state $\Sigma_\infty = I_3$.

Runtime

The dominant cost of SBTM per time step is the score-matching update, which is $O(n)$ in the number of particles and parallelizes well on the GPU. The collision-force evaluation (line 9 of Algorithm 1) is $O(n^2)$ but is performed only once per time step, whereas the score-matching loop runs for K steps; in practice the score-matching update dominates wall time at small to

moderate n. The blob method, by contrast, recomputes a kernel density estimate at every particle position at every time step, costing $O(n^2)$ per step regardless of any inner loops.

Figure 2.6 shows wall-clock times as a function of n for both methods on CPU and on GPU. On the GPU the heat-equation implementation of SBTM scales empirically as roughly $O(n^{0.6})$, owing to parallelism in the network forward and backward passes; the blob method's scaling remains close to the asymptotic $O(n^2)$.

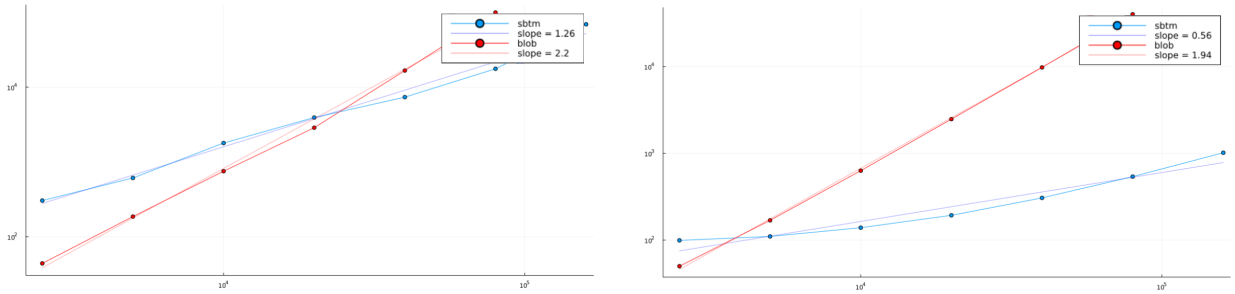


Figure 2.6: Wall-clock time vs. number of particles n on CPU (left, 10 threads) and GPU (right) on log-log axes. SBTM scales linearly (sublinearly on GPU); the blob method scales quadratically. The GPU timing is for the heat equation.

2.1.5 Summary

In the homogeneous setting SBTM offers four concrete advantages over the blob method: (i) better accuracy for anisotropic initial data, (ii) better accuracy in the sparse-particle regime including high dimension, (iii) linear-in-n runtime for the score-estimation step, and (iv) straightforward GPU parallelization through standard ML libraries. The main drawback is the proliferation of hyperparameters – network width and depth, learning rate, GD-step budget K, denoising scale α – and the need for some tuning per problem class. The adaptive training strategy mentioned above mitigates this and is a promising direction for future work.

These results motivate the next step: extending SBTM from the spatially homogeneous Landau equation to the full Vlasov–Maxwell–Landau system, in which the velocity-space

collision operator is coupled to spatial transport and to self-consistent electromagnetic fields.

2.2 Vlasov–Maxwell–Landau system

We now turn to the full Vlasov–Maxwell–Landau (VML) system, in which the collision operator of Section 2.1 is coupled to spatial transport and to self-consistent electromagnetic fields. The system reads

$$\partial_t f + v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = \nu\, Q[f, f], \tag{2.19}$$

$$\partial_t E = \nabla_x \times B - J, \qquad \partial_t B = -\nabla_x \times E, \qquad \nabla_x \cdot E = \rho - \rho_{\text{ion}}, \qquad \nabla_x \cdot B = 0, \tag{2.20}$$

with $f(t, x, v)$ the electron distribution function on $\Omega_x \times \mathbb{R}^{d_v}$, $E(t, x)$ and $B(t, x)$ the electromagnetic fields, $\rho = \int f \,\mathrm{d}v$ and $J = \int v f \,\mathrm{d}v$ the charge and current densities, ρ_{ion} a uniform neutralizing ion background with $\int_{\Omega_x} (\rho - \rho_{\text{ion}}) \,\mathrm{d}x = 0$, $\nu \geq 0$ the collision frequency, and $Q[f, f]$ the Landau collision operator (2.1) acting only in the velocity variable.

In the electrostatic case $B \equiv 0$ the magnetic field decouples and the system reduces to the Vlasov–Poisson–Landau (VPL) system,

$$\partial_t f + v \cdot \nabla_x f + E \cdot \nabla_v f = \nu Q[f, f], \qquad \partial_t E = -J, \qquad E = -\nabla_x \phi, \qquad -\Delta_x \phi = \rho - \rho_{\text{ion}}. \tag{2.21}$$

The structural properties of the VML and VPL systems – conservation of mass, energy, and (in the electrostatic case) momentum, the H-theorem, and the characterization of the equilibrium state – are reviewed in Chapter 3. In this section we focus on the algorithm.

For concreteness we assume from this point on a one-dimensional periodic spatial domain $\Omega_x = [0, L]$ with periodic boundary conditions, and $d_v \in \{2, 3\}$ velocity dimensions; the discussion extends to higher spatial dimensions with cosmetic changes. Following standard usage in the PIC literature [79, 17, 14] we write $\eta = L/M$ for the spatial cell width and let ψ_η denote a symmetric spatial kernel of compact support, used both to localize the collision operator and to deposit charge and current onto the grid. The score function is now spatially dependent: $s(x, v) = \nabla_v \log f(x, v)$.

2.2.1 Regularized Landau operator and particle method

The score-based rewriting of the Landau operator [183, 24] now reads

$$Q[f,f](t,x,v) = \nabla_v \cdot \Big\{ f(t,x,v)\, U(t,x,v) \Big\}, \tag{2.22}$$

$$U(t,x,v) := \int_{\mathbb{R}^{d_v}} A(v-w) \left[\nabla \log f(t,x,v) - \nabla \log f(t,x,w) \right] f(t,x,w)\, \mathrm{d}w, \tag{2.23}$$

exposing the same structural property as in the homogeneous case: the collision operator is a divergence in v of f times a velocity field U, which depends on f only through the score. Unlike the homogeneous case, however, $U(t,x,v)$ depends pointwise on the spatial coordinate x, which does not admit a direct particle approximation. Following [14] we introduce a spatially regularized collision force

$$U^\eta(t,x,v) := \iint_{\Omega_x \times \mathbb{R}^{d_v}} \psi_\eta(x-y)\, A(v-w) \left[\nabla \log f(t,x,v) - \nabla \log f(t,y,w) \right] f(t,y,w)\, \mathrm{d}w\, \mathrm{d}y, \tag{2.24}$$

and the regularized collision operator $Q^\eta[f,f] := \nabla_v \cdot (f U^\eta)$. A short calculation shows that Q^η retains conservation of mass, momentum, and kinetic energy and the spatially-integrated H-theorem $\iint Q^\eta[f,f] \log f \, \mathrm{d}v \, \mathrm{d}x \le 0$.

With this regularization, the VML equation (2.19) becomes a conservation law,

$$\partial_t f + \nabla_x \cdot (vf) + \nabla_v \cdot \big((E + v \times B) f \big) = \nu \nabla_v \cdot (U^\eta f). \tag{2.25}$$

We represent f as a weighted empirical measure,

$$f(t,x,v) \approx \sum_{p=1}^{n} w_p\, \delta\big(x - x_p(t)\big)\, \delta\big(v - v_p(t)\big). \tag{2.26}$$

This is a weak solution of (2.25) provided the particle positions and velocities evolve according to the coupled ODE system

$$\frac{\mathrm{d}x_p}{\mathrm{d}t} = v_{1,p} \pmod{L}, \tag{2.27}$$

$$\frac{\mathrm{d}v_p}{\mathrm{d}t} = \big(E(x_p) + v_p \times B(x_p) \big) - \nu\, U^\eta(x_p, v_p), \tag{2.28}$$

where the regularized collision force on particle p is

$$U^{\eta}(x_p, v_p) = \sum_{q=1}^{n} w_q \, \psi_\eta(x_p - x_q) \, A(v_p - v_q) \left[s(x_p, v_p) - s(x_q, v_q) \right], \tag{2.29}$$

and $s(x, v)$ is an approximation of the velocity score $\nabla \log f(x, v)$.

To couple the particle dynamics (2.27)–(2.28) to Maxwell's equations (2.20) we use a standard PIC procedure [79, 17]. The spatial domain is partitioned into M cells of size $\eta = L/M$ centered at $x_j = (j - \frac{1}{2})\eta$, and we take ψ_η to be the degree-1 B-spline (hat function)

$$\psi_\eta(x) = \frac{1}{\eta}\big(1 - |x|/\eta\big)_+, \qquad (z)_+ := \max(z, 0). \tag{2.30}$$

The kernel has compact support $[-\eta, \eta]$, so only particles in the same or adjacent spatial cells interact through the collision operator. The same kernel is used to deposit charge and current onto the grid,

$$\rho_j = \sum_p w_p \, \psi_\eta(x_j - x_p), \qquad J_{i,j} = \sum_p w_p \, v_{i,p} \, \psi_\eta(x_j - x_p), \quad i = 1, \ldots, d_v, \tag{2.31}$$

and to interpolate the grid fields back to particle positions,

$$E(x_p) = \eta \sum_j \psi_\eta(x_p - x_j) \, E_j, \qquad B(x_p) = \eta \sum_j \psi_\eta(x_p - x_j) \, B_j. \tag{2.32}$$

Using ψ_η for both deposition and interpolation is a standard choice that ensures semi-discrete energy conservation [14]. Maxwell's equations are advanced with centered second-order finite differences [205].

The conservation properties of Theorem 2.2 carry over to the VML setting, with one structural change: in the presence of electromagnetic fields, the *collision* contribution to the particle dynamics still conserves momentum and kinetic energy and dissipates estimated entropy, but momentum at the level of the full system is no longer conserved (it is exchanged with electromagnetic momentum). The collision operator's contribution can be expressed in

weak form: for any test function $\phi(x, v)$,

$$\begin{aligned}
&\nu \sum_p w_p \, \phi(x_p, v_p) \cdot U^\eta(x_p, v_p) \\
&\quad = \nu \sum_{p,q} w_p w_q \, \psi_\eta(x_p - x_q) \, \phi(x_p, v_p)^\top A(v_p - v_q) \big[s(x_p, v_p) - s(x_q, v_q) \big] \\
&\quad = \frac{\nu}{2} \sum_{p,q} w_p w_q \, \psi_\eta(x_p - x_q) \, \big[\phi(x_p, v_p) - \phi(x_q, v_q) \big]^\top \\
&\qquad\qquad \times A(v_p - v_q) \big[s(x_p, v_p) - s(x_q, v_q) \big].
\end{aligned} \tag{2.33}$$

Theorem 2.4 (Conservation under the regularized PIC particle method)**.** *For any score approximation $s(x, v)$ and the regularized collision force U^η defined by* (2.29)*, the following hold:*

(i) ***Conservation of momentum and kinetic energy by the collision force:***

$$\sum_p w_p \, U^\eta(x_p, v_p) = 0, \qquad \sum_p w_p \, v_p \cdot U^\eta(x_p, v_p) = 0.$$

(ii) ***Estimated entropy dissipation:***

$$-\nu \sum_p w_p \, s(x_p, v_p) \cdot U^\eta(x_p, v_p) \le 0.$$

Proof. Conservation of momentum follows from (2.33) with $\phi(x, v) = e_i$, since the bracket $[\phi(x_p, v_p) - \phi(x_q, v_q)]$ vanishes. Conservation of kinetic energy follows by taking $\phi(x, v) = \frac{1}{2}|v|^2 e_i$ and noting $(v - w)^\top A(v - w) = 0$, since the kernel $A(z) = |z|^{\gamma+2}\Pi(z)$ with $\Pi(z) = I - zz^\top/|z|^2$ satisfies $z^\top A(z) = 0$ for all $z \in \mathbb{R}^{d_v}$. Estimated entropy dissipation follows by taking $\phi(x, v) = s(x, v)$ in (2.33): the resulting double sum is non-negative because $A(z)$ is positive semidefinite. □

Remark 2.5. Theorem 2.4 controls only the collision contribution. Provided Maxwell's equations are solved with an energy-conserving discretization (e.g., Yee's scheme [205]), the semi-discrete particle method (2.27)–(2.28) preserves total energy (kinetic + electromagnetic). An

energy-conserving second-order explicit time integrator was recently proposed in [207] and can be combined with the present method for fully discrete energy conservation, at the cost of two or three collision-force evaluations per step. We use forward Euler in the experiments below for simplicity.

2.2.2 Score estimation in the inhomogeneous setting

To estimate the score $s(x, v) = \nabla \log f(x, v)$ from the weighted particle ensemble, we apply the same SBTM strategy as in Section 2.1.3, now with the score depending on both spatial and velocity coordinates. The neural network $s_\theta : \mathbb{R}^{d_x+d_v} \to \mathbb{R}^{d_v}$ is trained to minimize the expected squared error in the velocity gradient, $\mathbb{E}_f\big[\, \|s_\theta - \nabla \log f\|^2 \,\big]$, which after integration by parts on the cross-term reduces to the implicit score-matching loss

$$\mathcal{L}(\theta) = \sum_{p=1}^{n} w_p \Big[\, |s_\theta(x_p, v_p)|^2 + 2\, \nabla_v \cdot s_\theta(x_p, v_p) \,\Big]. \tag{2.34}$$

The factor of w_p in (2.34) reflects the weighted empirical measure (2.26); otherwise the loss is identical to (2.12).

The divergence $\nabla_v \cdot s_\theta$ is computed via Hutchinson's stochastic trace estimator [84]: for $z \sim \text{Rademacher}(\pm 1)^{d_v}$,

$$\nabla_v \cdot s_\theta(x, v) \approx z^\top \big(\nabla_v s_\theta(x, v)\big) z, \tag{2.35}$$

which can be evaluated with one forward-mode automatic differentiation pass per particle. We use a two-layer feedforward network with the `softsign` activation, hidden dimension 256 for the Landau damping and two-stream benchmarks, and 512 for the more demanding Weibel benchmark. At $t = 0$ we pretrain the network on the analytic score of the chosen initial condition; at each subsequent time step we take K steps of AdamW with learning rate 2×10^{-4} on $\mathcal{L}(\theta)$.

The full method is summarized in Algorithm 2. We assume here, in line with the experiments below, that $E = (E_1, E_2, 0)$ and $B = (0, 0, B_3)$, in which case Maxwell's equations reduce to

$$\partial_t E_1 = -J_1, \quad \partial_t E_2 = -\partial_x B_3 - J_2, \quad \partial_t B_3 = -\partial_x E_2, \tag{2.36}$$

with all fields stored at cell centers x_j and updated by centered second-order finite differences.

Algorithm 2 SBTM-PIC for the Vlasov–Maxwell–Landau system

Require: Particles $\{x_p, v_p, w_p\}_{p=1}^n$, grid fields E_j, B_j, time step Δt, final time t_{final}, collision frequency ν, ISM steps K, kernels ψ_η and A

1: Pretrain s_θ on the analytic initial score
2: $t \leftarrow 0$
3: **while** $t < t_{\text{final}}$ **do**
4: $\quad E_p \leftarrow \eta \sum_j \psi_\eta(x_p - x_j)\, E_j; \quad B_p \leftarrow \eta \sum_j \psi_\eta(x_p - x_j)\, B_j$ *(grid → particle)*
5: $\quad v_p \leftarrow v_p + \Delta t\,(E_p + v_p \times B_p)$ *(Lorentz push)*
6: $\quad x_p \leftarrow x_p + \Delta t\, v_{1,p} \pmod L$ *(position advance)*
7: $\quad \rho_j \leftarrow \sum_p w_p\, \psi_\eta(x_j - x_p); \quad J_{i,j} \leftarrow \sum_p w_p\, v_{i,p}\, \psi_\eta(x_j - x_p)$ *(particle → grid)*
8: $\quad$ Advance E_j and B_j by one step of (2.36) *(field update)*
9: $\quad$ **for** $k = 1, \dots, K$ **do**
10: $\quad\quad$ Draw $z \sim \text{Rademacher}(\pm 1)^{d_v}$
11: $\quad\quad \theta \leftarrow \theta - \alpha\, \nabla_\theta \Big[\sum_p w_p \big(|s_\theta(x_p, v_p)|^2 + 2\, z^\top (\nabla_v s_\theta(x_p, v_p))\, z \big) \Big]$ *(ISM)*
12: $\quad$ **end for**
13: $\quad v_p \leftarrow v_p - \Delta t\, \nu\, U^\eta(x_p, v_p)$ with U^η from (2.29) and $s = s_\theta$ *(collision)*
14: $\quad t \leftarrow t + \Delta t$
15: **end while**

Comparison with the blob method. The blob method of [23, 24, 14] replaces the score-matching update (lines 9–12 of Algorithm 2) with a kernel density estimate of the score,

$$s_{\text{KDE}}(x, v) = \nabla_v \log\Big(\sum_q w_q\, K_h(v - v_q)\, \psi_\eta(x - x_q) \Big), \tag{2.37}$$

where K_h is a Gaussian kernel with bandwidth h chosen by Silverman's rule [157]. Evaluating (2.37) requires $O(n^2)$ pairwise computations within each spatial cell and $O(n^2)$ memory. The variant of [14] adds a small entropy-dissipation correction; in the experiments below the

correction is roughly four orders of magnitude smaller than the KDE term, with negligible effect on particle trajectories, and we omit it.

2.2.3 Numerical experiments

We evaluate SBTM-PIC on three canonical benchmarks: Landau damping (a small perturbation of a Maxwellian equilibrium), the two-stream instability (counter-streaming Maxwellian beams), and the Weibel instability (an anisotropic bi-Maxwellian with a magnetic-field perturbation). All experiments use the 1D-d_vV geometry with periodic boundary conditions in space and $d_v \in \{2, 3\}$. The particles are initialized by sampling positions from the spatial marginal of f_0 and velocities from the velocity marginal, with equal weights $w_p = L/n$. The first two examples are electrostatic (the VPL system (2.21)); the Weibel example is fully electromagnetic (the VML system (2.19)–(2.20)). The implementation is in JAX [22] with double precision and is publicly available at `https://github.com/Vilin97/Vlasov-Landau-SBTM`. Experiments were run on NVIDIA L40S and H200 GPUs on the University of Washington Hyak and Tillicum clusters.

We compare two score-estimation methods within the same PIC framework: the blob (KDE) method (2.37) and SBTM (2.34). All other components of the algorithm are identical. We monitor discrete analogues of the conserved quantities – kinetic, electric, magnetic, and total energy – and the estimated entropy production $\dot{\mathcal{H}} = \sum_p w_p \, s(x_p, v_p) \cdot U^\eta(x_p, v_p)$, which is non-negative by Theorem 2.4 (ii) and should vanish at equilibrium.

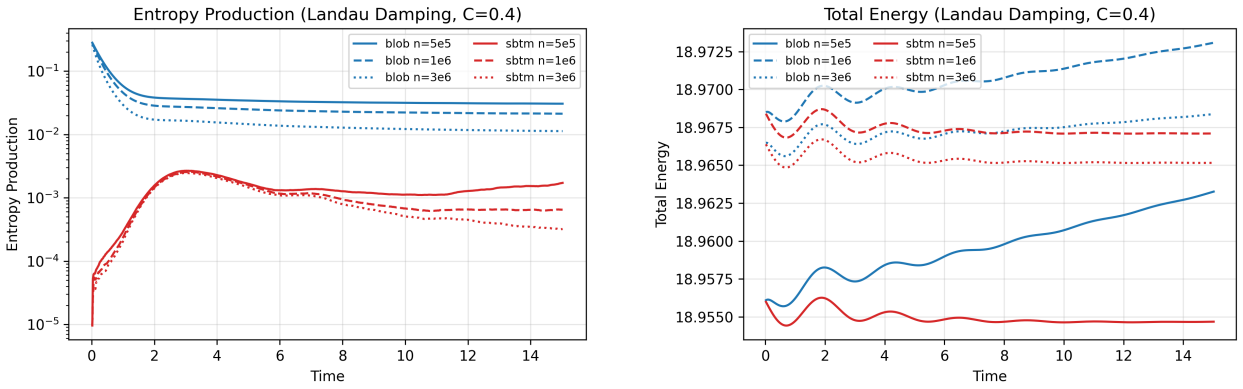
Landau damping

We consider a small spatial perturbation of a Maxwellian,

$$f_0(x, v) = \frac{1 + \alpha \cos(kx)}{(2\pi)^{d_v/2}} \exp\left(-\frac{|v|^2}{2}\right), \tag{2.38}$$

with wavenumber $k = 0.5$, domain $L = 2\pi/k$, and amplitude $\alpha = 0.1$. We use $d_v = 3$, $M = 100$ spatial cells, $\Delta t = 0.02$, $\nu = 0.4$, $t_{\text{final}} = 15$, and $K = 100$ ISM steps per time step, sweeping $n \in \{5 \times 10^5, 10^6, 3 \times 10^6\}$.

Estimated entropy and energy. Figure 2.7 shows the time evolution of the estimated entropy production and the total energy. By Chapter 3, the system relaxes to a Maxwellian, so the true entropy production must vanish at long times. SBTM's estimated entropy decays toward zero consistently across all particle counts. The blob method substantially overestimates entropy production at lower n and remains elevated at long times, reflecting persistent score-estimation errors in low-density regions. Although neither method conserves total energy exactly at the fully discrete level, the blob method exhibits visibly larger drift than SBTM.



(a) Estimated entropy production (log scale)

(b) Total energy (kinetic + electromagnetic)

Figure 2.7: Landau damping at $\nu = 0.4$. SBTM dissipates entropy consistently and maintains near-constant total energy across all particle counts; the blob method overestimates entropy production and exhibits larger energy drift.

Electric field decay. Figure 2.8 shows the L^2 norm of the electric field over time. At $\nu = 0.4$ – well outside the small-collisionality regime where linear theory is accurate – no exact damping rate is known. We use self-consistency across n as a proxy: the SBTM rate is essentially unchanged as n increases from 5×10^5 to 3×10^6, while the blob rate varies with n and converges toward the SBTM rate only at $n = 3 \times 10^6$. This is consistent with

SBTM achieving better score-estimation accuracy at fixed n.

Velocity-space relaxation. Figure 2.9 shows (v_1, v_2) phase-space heatmaps and the marginal density in v_2 on a logarithmic scale, at $n = 10^6$ for $\nu \in \{0.4, 1.0\}$. By the equilibrium theorem of Chapter 3, the system must relax to a Maxwellian whose temperature is determined by total-energy conservation; the linear-scale v_2-marginal accordingly approaches a standard Gaussian (the electric-field contribution to total energy is approximately 0.7%, so $T_\infty \approx 1.007$). On the log scale a clear difference emerges: SBTM produces smooth Gaussian tails, while the blob method exhibits unphysical sharp cutoffs in the low-density region $|v_2| \approx 3$. This is the signature failure mode of KDE in low-density tails.

Two-stream instability

The two-stream initial condition consists of two counter-streaming Maxwellian beams with a small spatial perturbation,

$$f_0(x, v) = \frac{1 + \alpha \cos(kx)}{2} \Big[\mathcal{N}(v_1; c, 1) + \mathcal{N}(v_1; -c, 1)\Big] \prod_{j \geq 2} \mathcal{N}(v_j; 0, 1), \tag{2.39}$$

with beam speed $c = 2.4$, $k = 1/5$, $\alpha = 1/200$, $L = 2\pi/k$, $M = 100$, $\Delta t = 0.05$, $d_v = 3$, $\nu = 0.32$, $t_{\text{final}} = 50$, and $K = 100$, sweeping $n \in \{5 \times 10^5, 10^6, 3 \times 10^6\}$.

Energy and entropy diagnostics. Figure 2.10 shows the energy partition and the estimated entropy production over time. The SBTM curves nearly coincide across n, indicating that $n = 5 \times 10^5$ is already in the converged regime; the blob method overestimates entropy production at low n and exhibits visibly larger total-energy drift.

Phase-space dynamics. A two-stream instability develops a phase-space vortex that should dissipate by $t = 50$ at this collision frequency (Figure 2.11). SBTM achieves full vortex dissipation at every particle count tested. The blob method requires $n = 3 \times 10^6$ to approach the same level of dissipation, again reflecting the slower convergence of the KDE-based score estimator.

Score quality. Figure 2.12 shows score quiver plots at $n = 10^6$ and $\nu = 0.24$. At $t = 0$ the true score is known analytically; SBTM matches it with low MSE, while the blob method has substantially higher error. At $t = t_{\text{final}}$ the SBTM score is smooth and physically consistent (vectors point toward the origin with magnitude increasing radially, as for a near-Maxwellian), whereas the blob score is noisy and incoherent in low-density tail regions.

Weibel instability

The Weibel instability is a fully electromagnetic test driven by velocity anisotropy. We initialize an anisotropic bi-Maxwellian with a small magnetic-field perturbation,

$$f_0(x,v) \propto \exp\left(-\frac{v_1^2}{\beta}\right)\left[\exp\left(-\frac{(v_2-c)^2}{\beta}\right)+\exp\left(-\frac{(v_2+c)^2}{\beta}\right)\right]\prod_{j\geq 3}\exp\left(-\frac{v_j^2}{\beta}\right), \qquad (2.40)$$

with $c = 0.3$, $\beta = 0.01$, $B_3(0,x) = \alpha_B \sin(kx)$, $\alpha_B = 10^{-3}$, $k = 1/5$, $L = 10\pi$, $M = 100$, $\Delta t = 0.1$, $d_v = 3$, $n = 10^6$, $t_{\text{final}} = 125$, $K = 100$, sweeping $\nu \in \{10^{-4}, 2\times10^{-4}, 4\times10^{-4}, 8\times10^{-4}\}$.

By the VML equilibrium theorem (Chapter 3), the system must relax to a Maxwellian $f_\infty \propto \exp(-|v|^2/2T_\infty)$ with $T_\infty = 0.035$ determined by total-energy conservation. Figure 2.13 shows the v_2-marginal evolution at $\nu = 8\times10^{-4}$ on logarithmic and linear scales. SBTM converges to the predicted Maxwellian with smooth Gaussian tails; the blob method produces sharp unphysical cutoffs at $v_2 \approx \pm 0.55$. On the linear scale the two methods look similar, but the log scale exposes the failure of the KDE-based score estimator to capture the tails. Table 2.1 quantifies the relaxation by the L^2 distance of the v_2-marginal to $\mathcal{N}(0, T_\infty)$ at t_{final}. SBTM is between 1.2× and 3.5× closer to the Maxwellian across the collision frequencies tested.

2.2.4 Computational efficiency

Table 2.2 compares wall-clock time and peak GPU memory between blob and SBTM on the two-stream benchmark. Both methods share an $O(n^2)$ collision-force evaluation per spatial cell, which dominates the total runtime. The score-estimation step is where the methods

ν	Blob	SBTM
2×10^{-4}	0.325	**0.265**
4×10^{-4}	0.134	**0.053**
8×10^{-4}	0.070	**0.020**

Table 2.1: Weibel instability: L^2 distance of the v_2-marginal to $\mathcal{N}(0, T_\infty)$ at $t_{\text{final}} = 125$ ($n = 10^6$, $d_v = 3$). SBTM relaxes closer to the Maxwellian steady state across all collision frequencies.

differ: the blob method forms and stores pairwise kernel evaluations, costing $O(n^2)$ memory per cell, while SBTM amortizes the score through a neural network whose parameter count is independent of n. At $n = 10^6$, SBTM is 1.57× faster than blob and uses 4× less memory; at $n = 3 \times 10^6$ it uses 3.8× less memory.

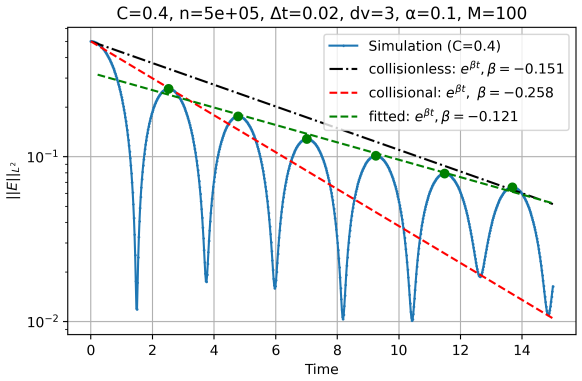
	Runtime ($n = 10^6$)		Peak memory (GB)	
Method	Time	Speedup	$n = 10^6$	$n = 3 \times 10^6$
Blob (KDE)	11h 27m	—	16.4	65.1
SBTM	7h 17m	1.57×	4.4	17.2

Table 2.2: Runtime and memory for the two-stream benchmark ($M = 100$, $\Delta t = 0.05$, $d_v = 3$, $t_{\text{final}} = 50$).

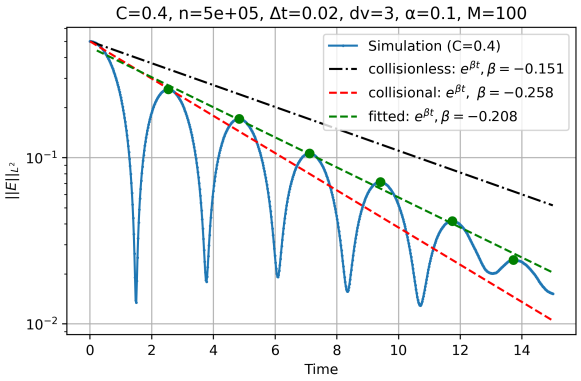
2.2.5 Summary

The score-based particle method extends from the homogeneous Landau equation to the full VML system without losing its structural advantages. Conservation of momentum and

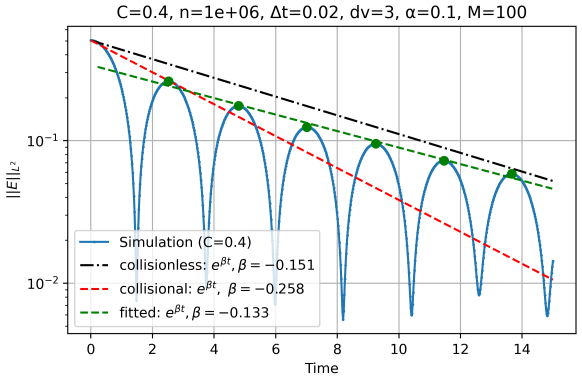
kinetic energy by the collision force (Theorem 2.4) and dissipation of estimated entropy hold for any score approximation. Across the three benchmarks – Landau damping, two-stream, Weibel – SBTM converges faster in n, produces smooth Gaussian tails where the blob method exhibits sharp cutoffs, and runs with substantially lower memory. The numerical behavior reported here motivates the analytical questions of Chapter 3: why the method is faithful to the conservation laws of the underlying PDE, in what norm the approximation error is controlled, and what equilibrium it must approach.



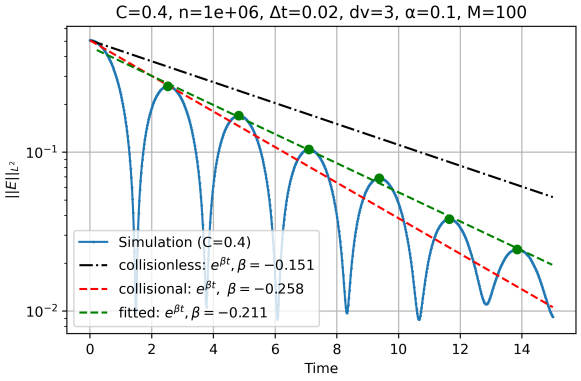
(a) Blob, $n = 5 \times 10^5$



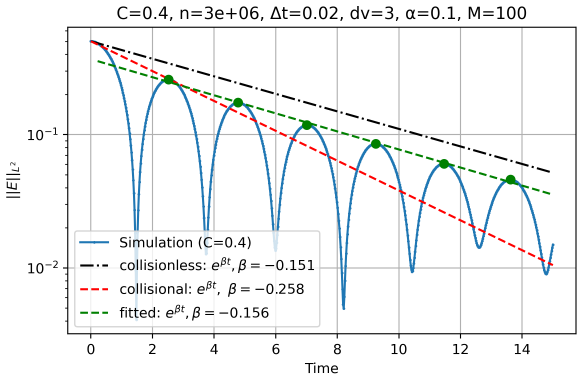
(b) SBTM, $n = 5 \times 10^5$



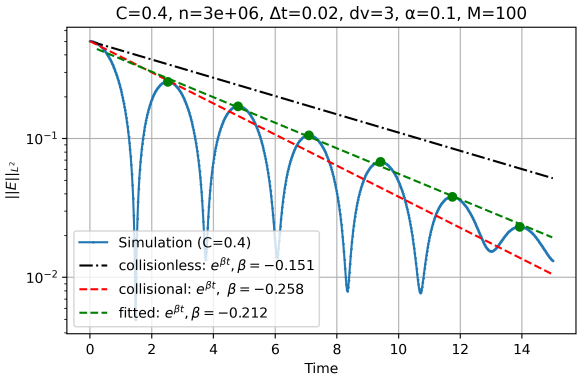
(c) Blob, $n = 10^6$



(d) SBTM, $n = 10^6$

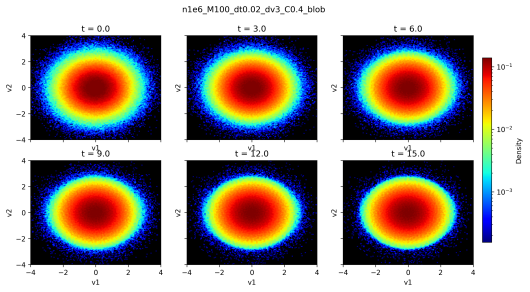


(e) Blob, $n = 3 \times 10^6$

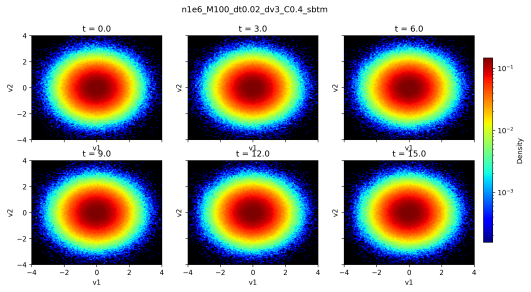


(f) SBTM, $n = 3 \times 10^6$

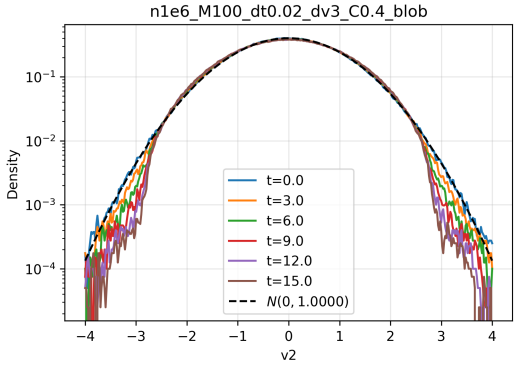
Figure 2.8: Landau damping: L^2 norm of the electric field at $\nu = 0.4$ across particle counts. The blob damping rate varies with n (panels (a), (c), (e)) and converges to the SBTM rate; the SBTM rate is consistent across all n.



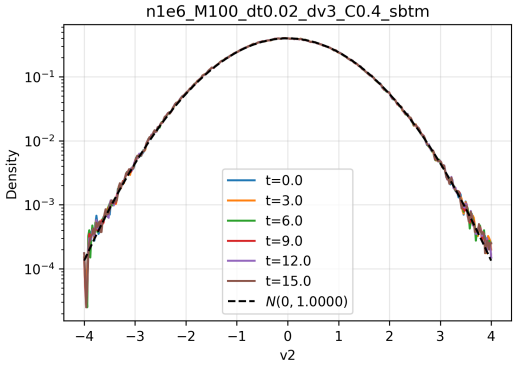
(a) Blob, $\nu = 0.4$



(b) SBTM, $\nu = 0.4$

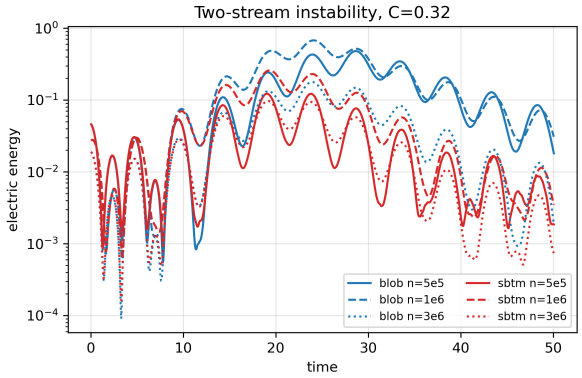


(c) Blob v_2-marginal, $\nu = 0.4$

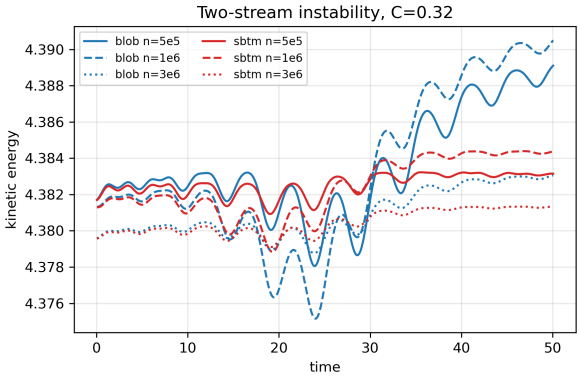


(d) SBTM v_2-marginal, $\nu = 0.4$

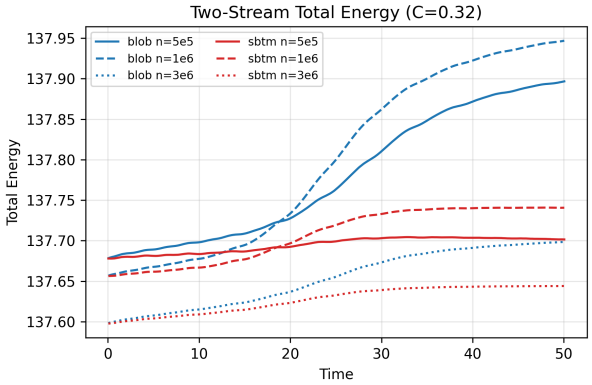
Figure 2.9: Landau damping: velocity-space distributions at $n = 10^6$, $\nu = 0.4$. SBTM (right column) maintains smooth Gaussian tails; the blob method (left column) shows unphysical sharp cutoffs at $|v_2| \approx 3$.



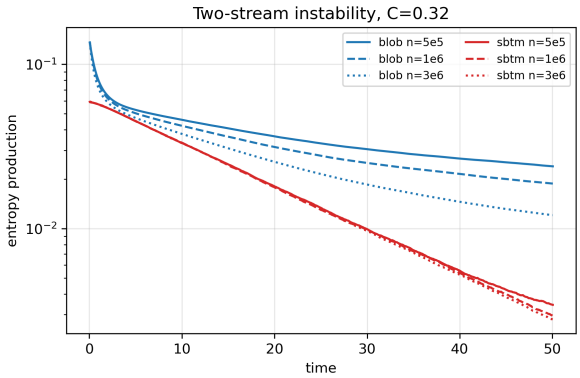
(a) Electric field energy



(b) Kinetic energy

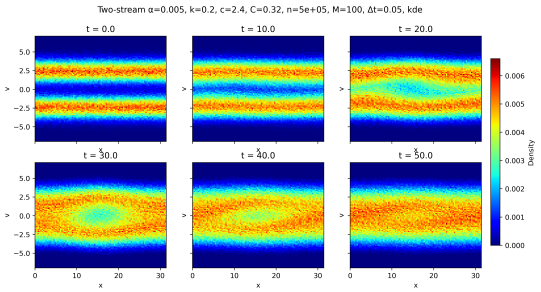


(c) Total energy

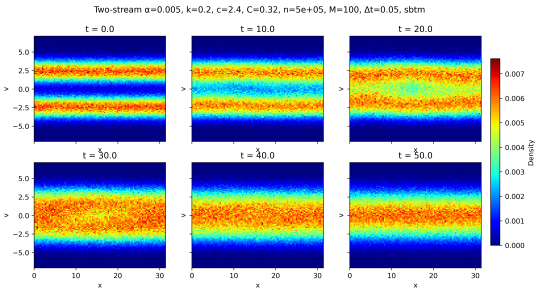


(d) Estimated entropy production

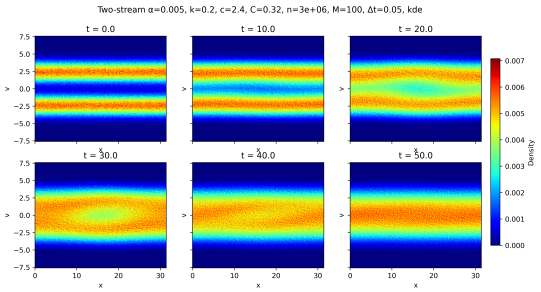
Figure 2.10: Two-stream instability at $\nu = 0.32$. Blob (blue) vs. SBTM (red) at $n = 5 \times 10^5$ (solid), 10^6 (dashed), 3×10^6 (dotted). SBTM curves nearly coincide across n.



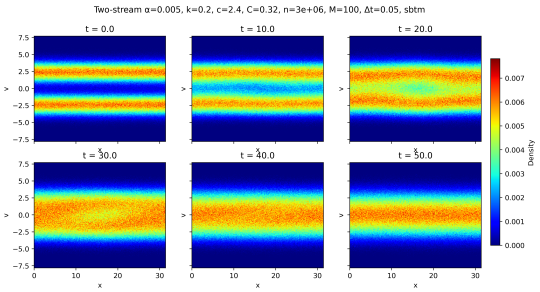
(a) Blob, $n = 5 \times 10^5$



(b) SBTM, $n = 5 \times 10^5$

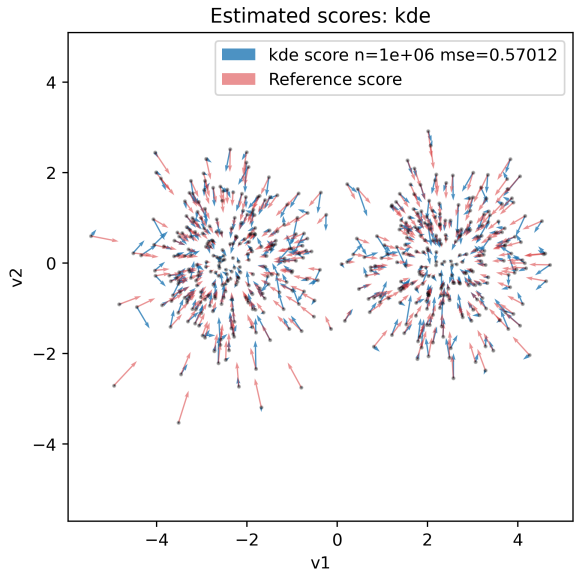


(c) Blob, $n = 3 \times 10^6$

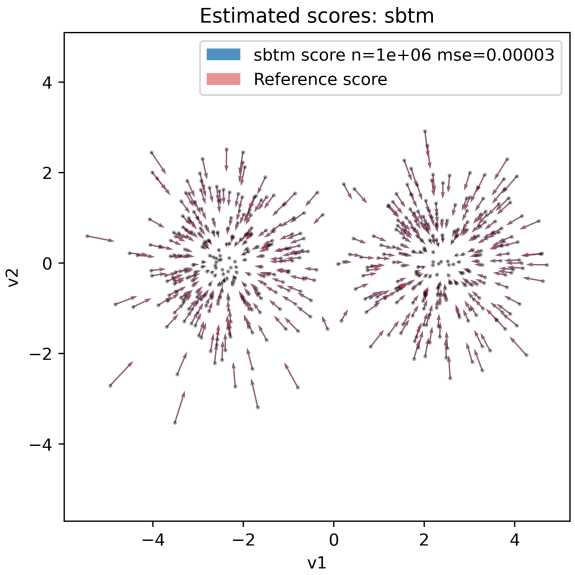


(d) SBTM, $n = 3 \times 10^6$

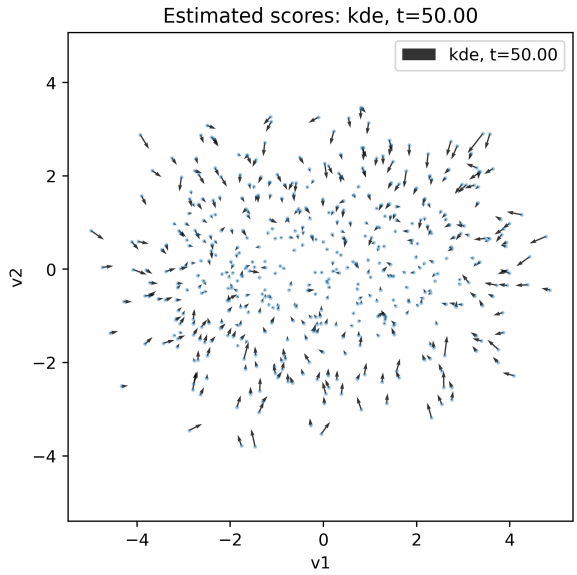
Figure 2.11: Two-stream instability: (x, v_1) phase space at $\nu = 0.32$. SBTM achieves full vortex dissipation at every particle count; the blob method requires $n = 3 \times 10^6$ to approach the same level.



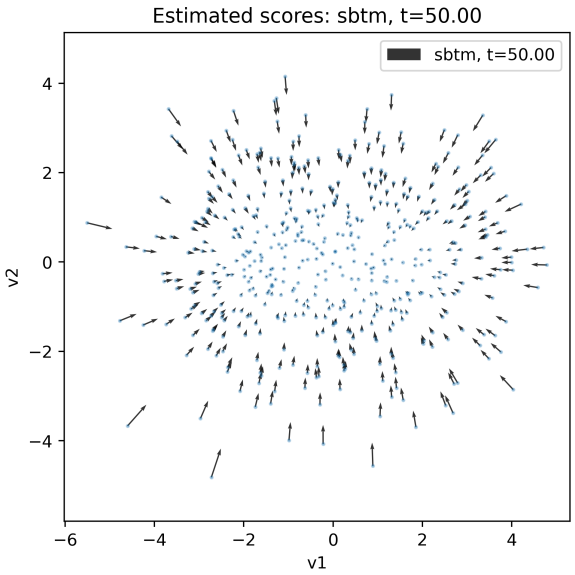
(a) Blob, $t = 0$



(b) SBTM, $t = 0$

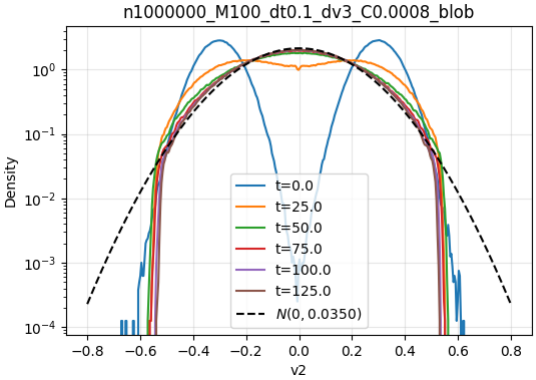


(c) Blob, $t = t_{\text{final}}$

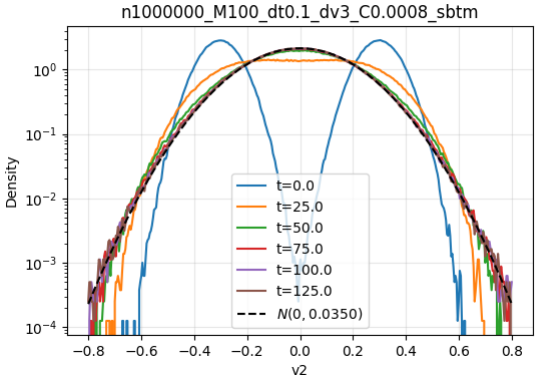


(d) SBTM, $t = t_{\text{final}}$

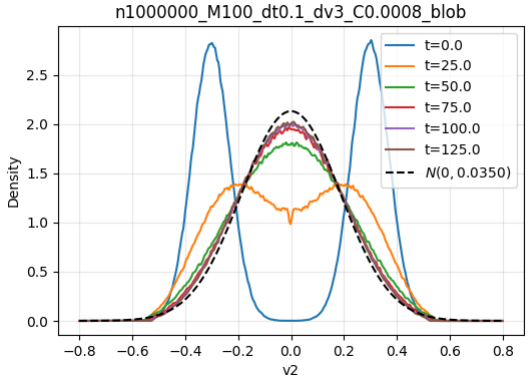
Figure 2.12: Two-stream instability: score quiver plots at $\nu = 0.24$, $n = 10^6$. Top: initial time (true score in red). Bottom: final time. SBTM produces smooth, physically consistent scores at both times.



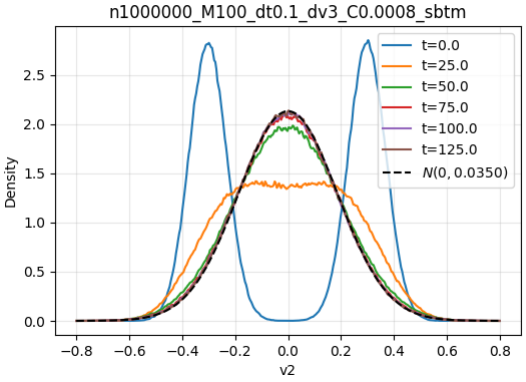
(a) Blob, log scale



(b) SBTM, log scale



(c) Blob, linear scale



(d) SBTM, linear scale

Figure 2.13: Weibel instability: v_2-marginal density evolution at $\nu = 8\times10^{-4}$. Top: log scale. Bottom: linear scale. Dashed line: target Maxwellian $\mathcal{N}(0, T_\infty)$. SBTM converges to it with smooth Gaussian tails; the blob method exhibits unphysical sharp cutoffs at $v_2 \approx \pm 0.55$.

Chapter 3

THEORY FOR COLLISIONAL PLASMA AND SCORE-BASED NUMERICS

This chapter collects the three theoretical pillars on which the score-based particle method of Chapter 2 rests. All three results bear on the same question – how faithfully a deterministic transport model can represent a collisional plasma – but address it from different angles. We prove that two solutions of the homogeneous Landau equation cannot drift apart faster than a controlled rate in relative entropy; we bound the relative entropy between the true solution and the score-based approximation by the score-matching loss; and we characterize the global equilibrium of the full Vlasov–Maxwell–Landau system.

For ease of reference, the headline statements of this chapter are gathered below. In each, $A(z) = |z|^{\gamma+2}(I_d - zz^\top/|z|^2)$ is the Landau collision kernel, $\nabla \log f = \nabla_v \log f$ is the velocity score, and $U[f,s](v) := -\int_{\mathbb{R}^d} A(v-w)\big[s(v)-s(w)\big] f(w)\,\mathrm{d}w$ is the score-driven velocity field of the homogeneous Landau equation in continuity form, as introduced in Section 2.1.1.

Theorem 3.8 (KL stability of the homogeneous Landau equation). If f, g solve the homogeneous Landau equation with Coulomb kernel ($d = 3$, $\gamma = -3$) and sufficient initial regularity, then

$$\mathrm{KL}(f_t \,\|\, g_t) \le C\, e^{t^2}\, \mathrm{KL}(f_0 \,\|\, g_0).$$

Theorem 3.17 (Maxwellian KL error bound). If f solves the homogeneous Landau equation with Maxwellian kernel ($\gamma = 0$) and g evolves according to the score-based continuity equation $\partial_t g + \nabla_v \cdot (g\, U[g, s_t]) = 0$ for some time-dependent vector field s_t, then for t small enough that $\mathrm{KL}(g_t \,\|\, f_t) \le 4$,

$$\mathrm{KL}(g_t \,\|\, f_t) \le 2e^{Mt} \int_0^t \int_{\mathbb{R}^d} \big|\nabla \log g_\tau - s_\tau\big|^2_{A*g_\tau} g_\tau \,\mathrm{d}v\,\mathrm{d}\tau,$$

with $M = M(d, T, f_0)$.

Theorem 3.18 (Coulomb KL error bound). If f solves the homogeneous Landau equation with Coulomb kernel ($d = 3$, $\gamma = -3$) and g evolves according to $\partial_t g + \nabla_v \cdot (g\, U[g, s_t]) = 0$, with $f_0 = g_0$ and sufficient initial regularity, then

$$\mathrm{KL}(f_T \,\|\, g_T) \le C\, e^{T^2} \int_0^T \left\| \tfrac{f_t}{g_t} \right\|_\infty \int_{\mathbb{R}^d} \left| \nabla \log g_t - s_t \right|^2_{A * g_t} g_t \,\mathrm{d}v \,\mathrm{d}t.$$

Theorem 3.20 (VML equilibrium). Every sufficiently smooth steady state of the Vlasov–Maxwell–Landau system on $\mathbb{T}^3 \times \mathbb{R}^3$ with $\nu > 0$ and a uniform neutralizing ion background is a spatially uniform, zero-drift global Maxwellian.

Theorem 3.21 (VPL equilibrium). Every sufficiently smooth steady state of the Vlasov–Poisson–Landau system on $\mathbb{T}^3 \times \mathbb{R}^3$ with $\nu > 0$ and a uniform neutralizing ion background is a spatially uniform, drifting global Maxwellian.

Theorem 3.20 is also the source of the formalization case study in Chapter 8: the Lean 4 verification carried out there is a machine-checkable rendering of the proof given here.

The remainder of the chapter is organized accordingly. Section 3.1 proves Theorem 3.8, building up the necessary regularity, coercivity, and weighted Fisher information estimates. Section 3.2 addresses error control for score-based approximations, treating the Maxwellian case (Theorem 3.17) as a warm-up before the harder Coulomb case (Theorem 3.18). Section 3.3 proves the equilibrium theorems for the full VML system and its electrostatic reduction.

3.1 Stability of the homogeneous Landau equation

We work throughout this section with the homogeneous Landau equation (2.1) written in continuity form

$$\partial_t f + \nabla_v \cdot \big(f\, U[f]\big) = 0, \qquad U[f](v) = -\int_{\mathbb{R}^d} A(v - w)\big[\nabla \log f(v) - \nabla \log f(w)\big] f(w)\,\mathrm{d}w, \tag{3.1}$$

with $A(z) = |z|^\gamma\big(|z|^2 I_d - z \otimes z\big)$ and the conventions of Section 2.1.1. We focus on the physically relevant Coulomb case $d = 3$, $\gamma = -3$. An auxiliary quantity that recurs throughout this section is the drift kernel

$$b(z) := \nabla \cdot A(z) = -2\, z\, |z|^\gamma, \tag{3.2}$$

so that $U[f] = -(A * f)\, \nabla \log f + b * f$.

3.1.1 Background estimates

We rely on three standard estimates from the homogeneous Landau literature, which we state without proof. The regularity result is from Section 23 of [185] combined with Theorem 0.1 of [71].

Lemma 3.1 (Regularity of solutions)**.** *Let f solve the homogeneous Landau equation* (3.1) *with $-4 < \gamma < -2$ and $\gamma \geq -d$, and assume the initial datum f_0 has finite energy, entropy, Fisher information, and sufficiently many moments. Then f is unique and smooth, and there exist $\alpha > 0$ and $s > 0$ such that*

$$\int_{\mathbb{R}^d} \langle v\rangle^s f_t(v)\, \mathrm{d}v \leq C(s, \gamma, f_0)(1 + t),$$
$$\|f_t\|_{L^p(\mathbb{R}^d)} \leq C(p, d, \gamma, f_0, \varepsilon)\big(t^{-\alpha_p(p,d,\gamma)} + t^\varepsilon\big) \quad \forall \varepsilon > 0,$$

and for any $k, \kappa > 0$ and any $\varepsilon > 0$, there exist $\alpha > 0$ and $s > 0$ such that $|\nabla^k f_t(v)| \leq C(f_0)(t^{-\alpha} + (1 + t)^\varepsilon)\, \langle v\rangle^{-\kappa}$. If additionally $\|f_0 \langle v\rangle^{9/(2p)}\|_{L^p} < \infty$ for some $p \geq 3/2$, then $\|f_t\|_{L^\infty} \leq C(f_0)\, t^{-3/(2p)}$.

The coercivity of the diffusion matrix $A * f$ in the soft-potential case is from [6, Proposition 2.1].

Lemma 3.2 (Coercivity of $A * f$)**.** *Let f be a probability density on $\mathbb{R}^d$ with finite energy $\frac{1}{2}\int |v|^2 f\, \mathrm{d}v = E$ and finite entropy $\int f \log f\, \mathrm{d}v = H$. Then there exists a constant $C_{\mathrm{coe}} = C_{\mathrm{coe}}(\gamma, E, H) > 0$ such that*

$$A * f(v) \succeq C_{\mathrm{coe}}\, \langle v\rangle^\gamma\, I_d \quad \text{for all } v \in \mathbb{R}^d.$$

The third estimate is the L^2 Pinsker inequality from [192, Lemma 2.2].

Lemma 3.3 (L^2 Pinsker inequality)**.** *Let f, g be probability densities in $L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d) \cap L^\infty(\mathbb{R}^d)$. Then*

$$\int_{\mathbb{R}^d} |f-g|^2 \,\mathrm{d}v \;\le\; 2\big(\|f\|_\infty + \|g\|_\infty\big)\,\mathrm{KL}(f \,\|\, g).$$

Proof. With $\varphi(x) = x\log x$, Taylor expansion gives $\varphi(f)-\varphi(g) = (1+\log g)(f-g)+\frac{1}{2}\zeta^{-1}(f-g)^2$ for some ζ between f and g. Integrating both sides over $\mathbb{R}^d$ and using $\int g\,\mathrm{d}v = \int f\,\mathrm{d}v = 1$ to drop the linear term yields $\int f\log(f/g)\,\mathrm{d}v \;\ge\; \frac{1}{2}(\|f\|_\infty + \|g\|_\infty)^{-1}\int (f-g)^2\,\mathrm{d}v$, which rearranges to the claim. □

3.1.2 Growth of the weighted Fisher information

The Coulomb estimates require control of weighted Fisher-information-type quantities along the flow. We derive these from the regularity lemma above together with the functional inequality of [178].

Theorem 3.4 (Weighted Fisher information growth)**.** *Let f solve the Coulomb Landau equation ($d = 3$, $\gamma = -3$) with f_0 as in Theorem 3.1. For any $s > 2$ there exist $C = C(\gamma, s, f_0) > 0$ and $\alpha = \alpha(\gamma, s, f_0) > 0$ such that*

$$\int_{\mathbb{R}^3} f_t(v)\,\langle v\rangle^s\,|\nabla \log f_t|^2\,\mathrm{d}v \;\le\; C\big(t^{-\alpha} + 1 + t\big).$$

If additionally $\|f_0\|_{H^2_5}^2 < \infty$, where

$$\|f\|_{H^2_5}^2 := \int_{\mathbb{R}^3} \big(|\nabla^2 f|^2 + |\nabla f|^2 + |f|^2\big)\langle v\rangle^{10}\,\mathrm{d}v,$$

then the same integral is bounded by $C(1+t)$.

Proof. By the estimates shown in [178, p. 1299, Proposition 5], one has the functional inequality

$$\int_{\mathbb{R}^3} f(v)\,\langle v\rangle^s\,|\nabla \log f|^2\,\mathrm{d}v \lesssim \|f\|_{H^2_{s+2}} + \int_{\mathbb{R}^3}\langle v\rangle^{s-2} f(v)\,\mathrm{d}v,$$

where

$$\|f\|_{H^2_{s+2}}^2 := \left\||\nabla^2 f(v)|\,\langle v\rangle^{s+2}\right\|_{L^2}^2 + \left\||\nabla f(v)|\,\langle v\rangle^{s+2}\right\|_{L^2}^2 + \left\||f(v)|\,\langle v\rangle^{s+2}\right\|_{L^2}^2.$$

Using the pointwise estimate $|\nabla^k f_t| \lesssim (t^{-\alpha} + (1+t))/\langle v\rangle^\kappa$ and the moment estimate $\int \langle v\rangle^{s-2} f_t(v)\,\mathrm{d}v \lesssim 1+t$ from Theorem 3.1 with κ large enough gives

$$\|f_t\|_{H^2_5}^2 \lesssim t^{-2\alpha} + (1+t)^2, \qquad \int f_t\,\langle v\rangle^s\,|\nabla \log f_t|^2\,\mathrm{d}v \lesssim t^{-\alpha} + 1 + t.$$

Further, if $\|f_0\|_{H^2_5}^2 < \infty$, then $\|f_t\|_{H^2_5}^2 \lesssim (1+t)^2$ and $\int f_t\,\langle v\rangle^s\,|\nabla \log f_t|^2\,\mathrm{d}v \lesssim 1+t$. □

The integrability conditions on f_0 used to close the stability proof are collected in the following lemma.

Lemma 3.5 (Sufficient initial-data condition)**.** *If $\|f_0\|_{H^2_5}^2 < \infty$, then the Coulomb Landau flow satisfies*

$$\|f_t\,\langle v\rangle^3\|_{L^2} \lesssim 1+t, \quad \|f_t\,\langle v\rangle^3\|_{L^1} \lesssim 1+t, \quad \|f_t\|_\infty \lesssim t^{-3/4}, \quad \|f_t\,\langle v\rangle^3 |\nabla \log f_t|^2\|_{L^1} \lesssim 1+t.$$

Proof. By Hölder's inequality and Theorem 3.1,

$$\|f\,\langle v\rangle^3\|_{L^2} \le \|f\|_{L^3}^{3/4} \cdot \|f\,\langle v\rangle^6\|_{L^1}^{1/4} \lesssim (t^{-\alpha} + t^{\varepsilon}) \cdot (1+t)^{1/4},$$
$$\|f\,\langle v\rangle^3\|_{L^1} \lesssim 1+t.$$

Assuming $\|f_0\|_{L^3} < \infty$ and choosing $\varepsilon = 3/4$ gives $\|f\,\langle v\rangle^3\|_{L^2} \lesssim 1+t$. Taking $p=2$ in the last inequality of Theorem 3.1 and assuming $\|f_0\|_{L^3} < \infty$, so that $\|f_0\,\langle v\rangle^{9/4}\|_{L^2} < \infty$, we get $\|f\|_\infty \lesssim t^{-3/4}$. Lastly, taking $s=3$ in Theorem 3.4 and assuming $\|f_0\|_{H^2_5} < \infty$, $\|f\,\langle v\rangle^3 |\nabla \log f|^2\|_{L^1} \lesssim 1+t$. By Sobolev embedding, $\|f_0\|_{L^3} \lesssim \|f_0\,|\nabla \log f_0|^2\|_{L^1} \le \|f_0\,\langle v\rangle^3\,|\nabla \log f_0|^2\|_{L^1} \lesssim \|f_0\|_{H^2_5}$. □

3.1.3 The key calculation

The stability and error-control theorems both descend from the same bilinear estimate, in which neither f nor g is assumed a priori to solve the Landau equation.

Lemma 3.6 (Key calculation)**.** *Let f, g be probability densities with finite energy and entropy, for which the integrals below are finite. Then for any d and $\gamma \le 0$,*

$$\begin{aligned}\int f\,(U[f]-U[g])\cdot\nabla\log\tfrac{f}{g}\,\mathrm{d}v \le\ & -\tfrac{1}{2}\int f\,\big|\nabla\log\tfrac{f}{g}\big|^2_{A*g}\,\mathrm{d}v \\ & + C_{\mathrm{coe}}\int f\,\langle v\rangle^{-\gamma}\,\|A*(f-g)\|^2\,|\nabla\log f|^2\,\mathrm{d}v \\ & + C_{\mathrm{coe}}\int f\,\langle v\rangle^{-\gamma}\,|b*(f-g)|^2\,\mathrm{d}v.\end{aligned} \tag{3.3}$$

In the Coulomb case $d=3$, $\gamma=-3$, this can be controlled further to give

$$\begin{aligned}\int f\,(U[f]-U[g])\cdot\nabla\log\tfrac{f}{g}\,\mathrm{d}v \le\ & -\tfrac{1}{2}\int f\,\big|\nabla\log\tfrac{f}{g}\big|^2_{A*g}\,\mathrm{d}v \\ & + C\,\mathrm{KL}(f\,\|\,g)\,\mathcal{R}(f,g),\end{aligned} \tag{3.4}$$

where

$$\mathcal{R}(f,g) := \big(\|f\|_\infty+\|g\|_\infty\big)\big(\|f\langle v\rangle^3|\nabla\log f|^2\|_{L^1}+\|f\langle v\rangle^3\|_{L^2}\big)+\|f\langle v\rangle^3\|_{L^1},$$

for some constant C depending only on the energy and entropy of f.

Proof. Note that

$$\begin{aligned}U[f](v) &= -\int A(v-w)\,\nabla\log f(v)\,f(w)\,\mathrm{d}w+\int A(v-w)\,\nabla f(w)\,\mathrm{d}w \\ &= -(A*f)\,\nabla\log f(v)+(b*f)(v),\end{aligned}$$

where $A(z)=|z|^\gamma(|z|^2 I_d - z\otimes z)$ and $b(z)=\nabla\cdot A(z)=-2z\,|z|^\gamma$. Hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f\,\|\,g) = -\int f\,\nabla\log\tfrac{f}{g}\cdot\big((A*f)\nabla\log f-(A*g)\nabla\log g\big)+\int f\,\nabla\log\tfrac{f}{g}\cdot b*(f-g).$$

By adding and subtracting the cross-term, one has

$$(A*f)\nabla\log f-(A*g)\nabla\log g=(A*g)\,\nabla\log\tfrac{f}{g}+(A*(f-g))\,\nabla\log f.$$

So far we have the exact expression

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f\,\|\,g) = -\int f\,\big|\nabla\log\tfrac{f}{g}\big|^2_{A*g}-\int f\,\nabla\log\tfrac{f}{g}\cdot(A*(f-g))\,\nabla\log f+\int f\,\nabla\log\tfrac{f}{g}\cdot b*(f-g).$$

Now we start estimating. By multiplying and dividing by $A * g$, and using Cauchy–Schwarz and Young's inequalities, one has

$$\left|\int f \nabla \log \tfrac{f}{g} \cdot (A * (f-g)) \nabla \log f\right| \leq \tfrac{1}{4} \int f \left|\nabla \log \tfrac{f}{g}\right|^2_{A*g} + \int f \left|A * (f-g) \nabla \log f\right|^2_{(A*g)^{-1}},$$
$$\int f \nabla \log \tfrac{f}{g} \cdot b * (f-g) \leq \tfrac{1}{4} \int f \left|\nabla \log \tfrac{f}{g}\right|^2_{A*g} + \int f \left|b * (f-g)\right|^2_{(A*g)^{-1}}.$$

Hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \leq -\tfrac{1}{2} \int f \left|\nabla \log \tfrac{f}{g}\right|^2_{A*g} + \int f \left|A*(f-g) \nabla \log f\right|^2_{(A*g)^{-1}} + \int f \left|b*(f-g)\right|^2_{(A*g)^{-1}}.$$

Given that g has finite energy and entropy, by Theorem 3.2, one has $A * g \geq C_{\mathrm{coe}} \langle v \rangle^{\gamma}$. Hence we obtain

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \leq\ & -\tfrac{1}{2} \int f \left|\nabla \log \tfrac{f}{g}\right|^2_{A*g} \\ & + C_{\mathrm{coe}} \int f \langle v \rangle^{-\gamma} \|A * (f-g)\|^2 |\nabla \log f|^2 + C_{\mathrm{coe}} \int f \langle v \rangle^{-\gamma} |b * (f-g)|^2.\end{aligned} \tag{3.5}$$

This is the general estimate (3.3).

We now specialize to the Coulomb case $d = 3, \gamma = -3$. Since $\|A(z)\| \lesssim 1/|z|$, the term $\|A * (f-g)\|$ can be controlled by Cauchy–Schwarz as follows:

$$\begin{aligned}\|A * (f-g)\| &\lesssim \int_{\mathbb{R}^3} \frac{1}{|v-w|} |f(w) - g(w)| \,\mathrm{d}w \\ &\leq \int_{|v-w|\leq 1} \frac{1}{|v-w|} |f(w) - g(w)| \,\mathrm{d}w + \|f-g\|_{L^1} \\ &\leq \Big(\int_{|w|\leq 1} \frac{1}{|w|^2} \,\mathrm{d}w\Big)^{1/2} \|f-g\|_{L^2} + \|f-g\|_{L^1} \\ &\lesssim \|f-g\|_{L^2} + \|f-g\|_{L^1} \\ &\leq \sqrt{2(\|f\|_\infty + \|g\|_\infty)\mathrm{KL}(f \,\|\, g)} + \sqrt{2\mathrm{KL}(f \,\|\, g)},\end{aligned}$$

where we applied both the classical and the L^2 Pinsker inequality (Theorem 3.3). Thus the first term in (3.5) becomes

$$\int_{\mathbb{R}^3} f \langle v \rangle^3 \|A * (f-g)\|^2 |\nabla \log f|^2 \lesssim (\|f\|_\infty + \|g\|_\infty) \|f \langle v \rangle^3 |\nabla \log f|^2\|_{L^1} \mathrm{KL}(f \,\|\, g). \tag{3.6}$$

Now we turn to the second term in (3.5). Since $|b(z)| \lesssim 1/|z|^2$, we obtain

$$\begin{aligned}|b * (f-g)| &\lesssim \int_{\mathbb{R}^3} \frac{1}{|v-w|^2}\,|f(w)-g(w)|\,\mathrm{d}w \\ &\le \int_{|v-w|\le 1} \frac{1}{|v-w|^2}\,|f(w)-g(w)|\,\mathrm{d}w + \|f-g\|_{L^1} \\ &\le \int_{\mathbb{R}^3} h(v-w)\,|f(w)-g(w)|\,\mathrm{d}w + \sqrt{2\mathrm{KL}(f\,\|\,g)},\end{aligned}$$

where $h(z) := |z|^{-2}\,\mathbf{1}_{|z|\le 1}$. Now, by Cauchy–Schwarz, for any $0 < a < 3/4$ (to ensure that h^{2a} is integrable),

$$\int_{\mathbb{R}^3} h(v-w)\,|f(w)-g(w)|\,\mathrm{d}w \le \|h^{2a}\|_{L^1}^{1/2}\Big(\int_{|v-w|\le 1} \frac{|f(w)-g(w)|^2}{|v-w|^{2-2a}}\,\mathrm{d}w\Big)^{1/2}.$$

Putting the two estimates above together and applying Young's convolution inequality with $1/p + 1/q = 1$,

$$\begin{aligned}\int f(v)\,\langle v\rangle^3\,|b*(f-g)|^2 &\lesssim \int f(v)\,\langle v\rangle^3 \Big(\int h(v-w)|f(w)-g(w)|\,\mathrm{d}w\Big)^2 + \mathrm{KL}(f\,\|\,g)\int f\,\langle v\rangle^3 \\ &\le \|h^{2a}\|_{L^1}\iint f(v)\,\langle v\rangle^3\,h(v-w)^{2-2a}\,|f(w)-g(w)|^2\,\mathrm{d}w\,\mathrm{d}v \\ &\quad + \mathrm{KL}(f\,\|\,g)\,\|f\,\langle v\rangle^3\|_{L^1} \\ &\le \|h^{2a}\|_{L^1}\,\|f\,\langle v\rangle^3\|_{L^p}\,\|h^{2-2a}\|_{L^q}\,\|f-g\|_{L^2}^2 \\ &\quad + \mathrm{KL}(f\,\|\,g)\,\|f\,\langle v\rangle^3\|_{L^1} \\ &\lesssim \mathrm{KL}(f\,\|\,g)\Big(\|h^{2a}\|_{L^1}\|f\langle v\rangle^3\|_{L^p}\|h^{2-2a}\|_{L^q}(\|f\|_\infty + \|g\|_\infty) \\ &\qquad + \|f\,\langle v\rangle^3\|_{L^1}\Big).\end{aligned}$$

Concretely, taking $a = 1/2$, $p = q = 2$ makes both $\|h^{2a}\|_{L^1}$ and $\|h^{2-2a}\|_{L^q}$ finite, giving

$$\int f(v)\,\langle v\rangle^3\,|b*(f-g)|^2 \lesssim \mathrm{KL}(f\,\|\,g)\,\Big(\|f\,\langle v\rangle^3\|_{L^2}(\|f\|_\infty + \|g\|_\infty) + \|f\,\langle v\rangle^3\|_{L^1}\Big). \tag{3.7}$$

Using (3.6) and (3.7) in (3.5) gives (3.4). □

Remark 3.7. Although Theorem 3.6 is stated for general densities f, g, it specializes to a differential inequality for the relative entropy when both are Landau solutions: in that case

$\partial_t f = -\nabla_v \cdot (fU[f])$ and similarly for g, and a direct calculation gives $\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, g_t) = \int f_t(U[f_t] - U[g_t]) \cdot \nabla \log(f_t/g_t)\,\mathrm{d}v$. This identity is the bridge between Theorem 3.6 and the stability theorem we now prove.

3.1.4 Proof of KL stability

Theorem 3.8 (KL stability of the homogeneous Landau equation)**.** *Let f, g solve the Coulomb Landau equation* (3.1) *($d = 3$, $\gamma = -3$) with initial data f_0, g_0 satisfying $\|f_0\|_{H^2_5}^2 < \infty$ and $\|g_0 \langle v\rangle^{9/4}\|_{L^2} < \infty$, together with finite energy and entropy. Then there is $C = C(f_0, g_0) > 0$ such that*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, g_t) \;\le\; C\,(1+t)\,\mathrm{KL}(f_t \,\|\, g_t), \tag{3.8}$$

and consequently

$$\mathrm{KL}(f_t \,\|\, g_t) \;\le\; C\,e^{t^2}\,\mathrm{KL}(f_0 \,\|\, g_0). \tag{3.9}$$

Proof. We will write f and $\int f$ in place of $f_t(v)$ and $\int f\,\mathrm{d}v$ respectively for simplicity. The following calculation holds for any values of d, γ:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) &= -\int \frac{f}{g}\partial_t g + \int \log\frac{f}{g}\,\partial_t f \\
&= -\int U[g] \cdot \nabla\Big(\frac{f}{g}\Big) g + \int U[f] \cdot \nabla \log\Big(\frac{f}{g}\Big) f \\
&= \int f\,(U[f] - U[g]) \cdot \nabla \log\frac{f}{g}.
\end{aligned}$$

Now, dropping the strictly negative term in the conclusion of Theorem 3.6 (3.4), we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) = \int f\,(U[f] - U[g]) \cdot \nabla \log\frac{f}{g} \le C\,\mathrm{KL}(f \,\|\, g)\,\mathcal{R}(f, g).$$

Finally, by Theorem 3.5, using the assumption $\|f_0\|_{H^2_5}^2 < \infty$, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \le C\,(1+t)\,\mathrm{KL}(f \,\|\, g) \sup_{0\le s\le t}\|g_s\|_\infty$$

for some $C = C(f_0)$. The integrability assumption on g_0 is sufficient to guarantee that $\sup_{0\le s\le t}\|g_s\|_\infty$ is finite. Thus

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \le C\,(1+t)\,\mathrm{KL}(f \,\|\, g)$$

where $C = C(f_0, g_0)$. Grönwall's inequality completes the proof. □

Remark 3.9. The exponent of t^2 is sharp for the elementary computation used here, but not necessarily for the underlying flow: weak–strong uniqueness in relative entropy was recently established in [171] by a dimension-doubling argument that yields tighter time dependence. Our proof remains in $\mathbb{R}^d$ and proceeds by an energy-type computation directly on $\mathrm{KL}(f \,\|\, g)$, which is what allows the same calculation to control the score-based numerical error in Theorem 3.18.

3.2 Error control for score-based approximations

We now turn to the question of how well the score-based particle method of Section 2.1.3 approximates the true Landau solution. We first treat the Maxwellian case ($\gamma = 0$), where a Fokker–Planck reformulation simplifies the analysis, and then the harder Coulomb case ($\gamma = -3$, $d = 3$), which uses the key calculation of Section 3.1.3 directly.

Throughout, f denotes the true Landau solution and g the score-based approximation, governed by

$$\partial_t g + \nabla_v \cdot \big(g\, U[g, s_t]\big) = 0, \qquad U[g, s_t](v) = -\int_{\mathbb{R}^d} A(v-w)\big[s_t(v) - s_t(w)\big] g(w)\,\mathrm{d}w, \quad (3.10)$$

for some time-dependent vector field $s_t : \mathbb{R}^d \to \mathbb{R}^d$ – the score estimate. In the SBTM scheme of Chapter 2, s_t is the output of a neural network trained by score matching; the bounds below hold for any sufficiently regular s_t.

3.2.1 Fokker–Planck reformulation in the Maxwellian case

The Maxwellian Landau equation admits a Fokker–Planck reformulation that makes the diffusion structure explicit. Conservation of mass, momentum, and kinetic energy along the flow gives $\int f\,\mathrm{d}v \equiv 1$, $V := \int v\, f\,\mathrm{d}v$ const., $2E := \int |v|^2 f\,\mathrm{d}v$ const. The (time-dependent) covariance matrix $\Sigma(t) := \int v \otimes v\, f_t(v)\,\mathrm{d}v$ is then the only non-trivial second-order moment; its evolution can be computed in closed form.

Proposition 3.10 (Fokker–Planck form of Maxwellian Landau)**.** *The Maxwellian Landau equation ($\gamma = 0$) can be written*

$$\partial_t f + \nabla_v \cdot \big(f\, V_t\big) = 0, \qquad V_t(v) = b(v) - D_t(v)\,\nabla \log f(v), \tag{3.11}$$

with drift $b(v) = A * \nabla f = -(d-1)(v - V)$, and diffusion matrix

$$\begin{aligned} D_t(v) &= A * f_t(v), \\ D_{ij}(t,v) &= \delta_{ij}\big(|v|^2 + 2E - 2V\cdot v\big) + V_i v_j + V_j v_i - v_i v_j - \Sigma_{ij}(t), \\ \Sigma_{ij}(t) &= \Sigma_{ij}(\infty) - \big(\Sigma_{ij}(\infty) - \Sigma_{ij}(0)\big)e^{-4dt}, \\ \Sigma_{ij}(\infty) &= \tfrac{1}{d}\big(\delta_{ij}(2E - |V|^2) + d\, V_i V_j\big). \end{aligned}$$

Proof. We must show that the velocity field V_t given by $U[f]$ in (3.1) can be written in the form $b - D\nabla \log f$. First, split V_t into two terms:

$$\begin{aligned} V_t &= -\int_{\mathbb{R}^d} A(v-w)\big[\nabla \log f(v) - \nabla \log f(w)\big] f(w)\,\mathrm{d}w \\ &= (A * \nabla f) - (A * f)\,\nabla \log f, \end{aligned}$$

where the first term is the drift and the second the diffusion. The drift simplifies to

$$(A * \nabla f)_i = \sum_j \partial_j A_{ij} * f = -(d-1)(v_i - V_i).$$

For the diffusion term, compute the convolution $A * f$:

$$\begin{aligned} (A * f)_{ij} &= \int_{\mathbb{R}^d} \delta_{ij}\,|v-w|^2\, f(w)\,\mathrm{d}w - \int_{\mathbb{R}^d} (v_i - w_i)(v_j - w_j)\, f(w)\,\mathrm{d}w \\ &= \delta_{ij}(|v|^2 + 2E - 2V\cdot v) + V_i v_j + V_j v_i - v_i v_j - \Sigma_{ij}. \end{aligned}$$

By a straightforward calculation, Σ satisfies

$$\Sigma_{ij}(t) = \Sigma_{ij}(\infty) - \big(\Sigma_{ij}(\infty) - \Sigma_{ij}(0)\big)e^{-4dt}.$$

This completes the proof. □

The proof of the Maxwellian KL bound also requires control of the score along the flow. The following sublinear estimate is from [64].

Theorem 3.11 (Linear score upper bound, Maxwellian case)**.** *Let f solve the Maxwellian Landau equation with initial datum $f_0 \in W^{2,1} \cap W^{2,\infty}$ satisfying $f_0(v) \le C_1\, e^{-|v|^2/C_1}$ and $|\nabla \log f_0(v)| \le C_2(1+|v|)$. Then for any $T > 0$, there exists a universal constant $C > 0$ such that for all $t \le T$,*

$$|\nabla \log f_t(v)| \le C(1 + |v| + \sqrt{t}).$$

A second ingredient is the explicit two-sided bound on the diffusion matrix $A * f$ in the Maxwellian case, which follows from a direct computation.

Lemma 3.12 (Boundedness of $A * f$ in the Maxwellian case)**.** *For any distribution f not concentrated on a line and Maxwellian collision kernel $A(z) = |z|^2 I_d - z \otimes z$, the matrix $A * f$ is uniformly bounded from below and from above, i.e., there exists $\varepsilon > 0$ such that for all $v \in \mathbb{R}^d$,*

$$\varepsilon \le \|A * f(v)\| \le 2E + |v - V|^2, \quad 2E = \int_{\mathbb{R}^d} |v|^2 \,\mathrm{d}f(v), \quad V = \int_{\mathbb{R}^d} v \,\mathrm{d}f(v),$$

where $\|\cdot\|$ denotes the matrix 2-norm, and ε can be taken to be

$$\varepsilon = \operatorname{tr}(\operatorname{cov}(f)) - \|\operatorname{cov}(f)\|.$$

In particular, the bounds above hold for any continuous density.

Proof. Denote $v' = v - V$ and compute

$$A * f(v) = I_d\big(|v'|^2 + 2E - |V|^2\big) - \big(v' \otimes v' + \Sigma - V \otimes V\big).$$

By the triangle inequality, and noting $\|v' \otimes v'\| = |v'|^2$,

$$\|A * f(v)\| \ge |v'|^2 + \operatorname{tr}(\Sigma - V \otimes V) - |v'|^2 - \|\Sigma - V \otimes V\| = \operatorname{tr}(\operatorname{cov}(f)) - \|\operatorname{cov}(f)\| > 0.$$

The last inequality is strict because $\operatorname{tr}(\operatorname{cov}(f))$ is the sum of the eigenvalues of $\operatorname{cov}(f)$ whereas $\|\operatorname{cov}(f)\|$ is the largest eigenvalue. The two are equal iff f is concentrated on a line.

For the upper bound, recalling that Σ is positive semi-definite,

$$\|A * f(v)\| \le |v'|^2 + 2E - |V|^2 - \inf_{|z|=1}\{z^\top \Sigma z + |v \cdot z|^2\} + |V|^2 \le 2E + |v - V|^2.$$

□

3.2.2 KL bound via the Fokker–Planck form

We first record a generic KL bound for Fokker–Planck-type equations [20, Proposition 1], the Boffi–Vanden-Eijnden estimate that underlies the SBTM analysis.

Theorem 3.13 (Generic Fokker–Planck KL bound)**.** *Fix $b_t(v)$ and a positive semidefinite $D_t(v)$. Let f, g satisfy the continuity equations*

$$\partial_t f + \nabla_v \cdot (f V_t^*) = 0, \quad V_t^* = b_t - D_t \nabla \log f, \qquad \partial_t g + \nabla_v \cdot (g V_t) = 0, \quad V_t = b_t - D_t s_t.$$

Then

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g_t \,\|\, f_t) \;\le\; \tfrac{1}{2}\int_{\mathbb{R}^d} |s_t - \nabla \log g_t|^2_{D_t}\, g_t \,\mathrm{d}v.$$

Combining Theorem 3.13 with Theorem 3.10 immediately gives a KL bound for the Maxwellian Landau equation, provided the Fokker–Planck velocity $b - D_t s_t$ is used in the approximate flow.

Corollary 3.14 (KL bound with FP velocity, isotropic data)**.** *If f solves the Maxwellian Landau equation, g satisfies $\partial_t g + \nabla_v \cdot (g(b - D_t s_t)) = 0$ with b, D_t as in Theorem 3.10, and the initial first and second moments of f_0 and g_0 match, then*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g_t \,\|\, f_t) \;\le\; \tfrac{1}{2}\int_{\mathbb{R}^d} |s_t - \nabla \log g_t|^2_{D_t}\, g_t \,\mathrm{d}v.$$

Proof. Conservation of mass, momentum, and kinetic energy along both flows ensures $D_t(v)$ in Theorem 3.10 is the same for f and g, and is positive semidefinite by Theorem 3.12. The bound follows from Theorem 3.13. □

This already gives a clean KL bound, but only when the approximate flow uses the Fokker–Planck velocity $b - D_t s_t$ – not the original collision-form velocity $U[g, s_t]$ – and only in the Maxwellian case. The next theorem removes the first restriction at the cost of a Grönwall factor.

3.2.3 Maxwellian KL bound for the collision-form approximation

Proposition 3.15 (Intermediate KL bound)**.** *Let f solve the Maxwellian Landau equation* (3.1) *and let g satisfy the score-based flow* (3.10)*, both with the same initial datum $f_0 = g_0$ of finite energy and entropy. Then*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g_t \,\|\, f_t) \le 2\int_{\mathbb{R}^d} \left|\nabla \log g_t - s_t\right|^2_{A*g_t} g_t \,\mathrm{d}v + \int_{\mathbb{R}^d} (\nabla \log f_t)^\top (\Sigma_g - \Sigma_f)(\nabla \log g_t - \nabla \log f_t)\, g_t \,\mathrm{d}v, \tag{3.12}$$

where Σ_g, Σ_f are the covariance matrices of g_t, f_t.

Proof. We omit the subscript t throughout this proof to simplify notation. Let $V_t = U[g, s_t]$ and $V_t^* = U[f]$ denote the velocity fields driving g and f respectively. Define an auxiliary vector field

$$\bar{V}(v) := -\int_{\mathbb{R}^d} A(v-w)\big[\nabla \log g(v) - \nabla \log g(w)\big] g(w) \,\mathrm{d}w,$$

and compute

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g \,\|\, f) &= -\int_{\mathbb{R}^d} \frac{g}{f}\partial_t f \,\mathrm{d}v + \int_{\mathbb{R}^d} \log\frac{g}{f}\,\partial_t g \,\mathrm{d}v \\
&= \int_{\mathbb{R}^d} \frac{g}{f}\nabla\cdot(V^* f)\,\mathrm{d}v - \int_{\mathbb{R}^d} \nabla\cdot(Vg)\log\Big(\frac{g}{f}\Big)\,\mathrm{d}v \\
&= -\int_{\mathbb{R}^d} V^*\cdot\nabla\Big(\frac{g}{f}\Big) f \,\mathrm{d}v + \int_{\mathbb{R}^d} V\cdot\nabla\log\Big(\frac{g}{f}\Big) g \,\mathrm{d}v \\
&= \int_{\mathbb{R}^d} (V - V^*)\cdot(\nabla\log g - \nabla\log f)\, g \,\mathrm{d}v \\
&= \int_{\mathbb{R}^d} (V - \bar{V})\cdot(\nabla\log g - \nabla\log f)\, g \,\mathrm{d}v + \int_{\mathbb{R}^d} (\bar{V} - V^*)\cdot(\nabla\log g - \nabla\log f)\, g \,\mathrm{d}v.
\end{aligned}$$

Bound the first term in this decomposition with Young's inequality:

$$\begin{aligned}
&\int_{\mathbb{R}^d} (V - \bar{V}) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\quad = \int_{\mathbb{R}^d} (\nabla \log g - s)^\top (A * g) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\qquad - \int_{\mathbb{R}^d} A * (g(\nabla \log g - s)) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\quad \le \int_{\mathbb{R}^d} |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + \tfrac{1}{4} \int_{\mathbb{R}^d} |\nabla \log g - \nabla \log f|^2_{A*g}\, g \,\mathrm{d}v \\
&\qquad - \int_{\mathbb{R}^d} A * (g(\nabla \log g - s)) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v,
\end{aligned}$$

where

$$\begin{aligned}
&- \int_{\mathbb{R}^d} A * (g(\nabla \log g - s)) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\quad \le \iint |\nabla \log g(w) - s(w)|^2_{A(v-w)}\, g(v)\, g(w) \,\mathrm{d}v \,\mathrm{d}w \\
&\qquad + \tfrac{1}{4} \iint |\nabla \log g(v) - \nabla \log f(v)|^2_{A(v-w)}\, g(v)\, g(w) \,\mathrm{d}v \,\mathrm{d}w \\
&\quad = \int_{\mathbb{R}^d} |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + \tfrac{1}{4} \int_{\mathbb{R}^d} |\nabla \log g - \nabla \log f|^2_{A*g}\, g \,\mathrm{d}v.
\end{aligned}$$

Together,

$$\int_{\mathbb{R}^d} (V - \bar{V}) \cdot \nabla \log\Big(\tfrac{g}{f}\Big) g \,\mathrm{d}v \le 2 \int_{\mathbb{R}^d} |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + \tfrac{1}{2} \int_{\mathbb{R}^d} \big|\nabla \log \tfrac{g}{f}\big|^2_{A*g}\, g \,\mathrm{d}v. \tag{3.13}$$

Now analyze the second term. Using that A is the Maxwellian kernel, we can write

$$\begin{aligned}
V^* &= -(A * f)\, \nabla \log f - (d-1)(v - V), \\
\bar{V} &= -(A * g)\, \nabla \log g - (d-1)(v - V),
\end{aligned}$$

so the drift term in $\bar{V} - V^*$ cancels:

$$\bar{V} - V^* = (A * f)\, \nabla \log f - (A * g)\, \nabla \log g.$$

Thus

$$\begin{aligned}
&\int_{\mathbb{R}^d} (\bar{V} - V^*) \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\quad = \int_{\mathbb{R}^d} \big[(A * f)\nabla \log f - (A * g)\nabla \log g\big] \cdot (\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v \\
&\quad = -\int_{\mathbb{R}^d} |\nabla \log g - \nabla \log f|^2_{A*g}\, g \,\mathrm{d}v + \int_{\mathbb{R}^d} (\nabla \log f)^\top (A * (f - g))(\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v.
\end{aligned} \tag{3.14}$$

Combining (3.13) and (3.14),

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g \,\|\, f) \le 2\int_{\mathbb{R}^d} |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v - \tfrac{1}{2}\int_{\mathbb{R}^d} |\nabla \log g - \nabla \log f|^2_{A*g}\, g \,\mathrm{d}v \\
+ \int_{\mathbb{R}^d} (\nabla \log f)^\top (A * (f - g))(\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v,
\end{aligned} \tag{3.15}$$

where the second term is non-positive because $A * g$ is positive semi-definite (in fact strictly positive definite by Theorem 3.12).

Finally, $A * (f - g) = \Sigma_g - \Sigma_f$. This can be seen directly by computing

$$\begin{aligned}
(A * g)_{ij}(v) &= \delta_{ij}(|v|^2 + 2E - 2V \cdot v) + (v_i V_j + v_j V_i - v_i v_j) - \Sigma_{g,ij}, \\
(A * f)_{ij}(v) &= \delta_{ij}(|v|^2 + 2E - 2V \cdot v) + (v_i V_j + v_j V_i - v_i v_j) - \Sigma_{f,ij},
\end{aligned}$$

and subtracting the two equations. This proves the Proposition. □

The residual term in (3.12) vanishes when the second moments match identically. This is automatic when the initial condition is isotropic.

Corollary 3.16 (Isotropic Maxwellian KL bound)**.** *If the initial data $f_0 = g_0$ is radially symmetric and s_t is of the form $s_t(v) = v\,\hat{s}_t(|v|)$ for some $\hat{s}_t : \mathbb{R} \to \mathbb{R}^d$, then*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g_t \,\|\, f_t) \;\le\; 2\int_{\mathbb{R}^d} \big|\nabla \log g_t - s_t\big|^2_{A*g_t} g_t \,\mathrm{d}v.$$

Proof. With f_0 isotropic, f_t stays isotropic (the collision operator preserves rotational symmetry). An induction shows that if s_t has the displayed radial form, the velocity field $U[g, s_t]$ is radial, hence g_t is also isotropic. Then Σ_g and Σ_f are scalar matrices, and since their traces both equal $2E$ (conserved), they are equal. The residual term in Theorem 3.15 vanishes. □

For general initial data the residual term is nontrivial but can be controlled in a short-time window using a weighted Csiszár–Kullback–Pinsker inequality [21].

Theorem 3.17 (Maxwellian KL error bound)**.** *Fix $T > 0$ and assume that $f_0 = g_0 \in W^{2,1} \cap W^{2,\infty}$ with $|\nabla \log f_0(v)| \le C_0(1+|v|)$, $f_0(v) \le C_0' e^{-3\beta|v|^2}$ for some $\beta > 0$. Then for any $t \le T$ small enough that $\mathrm{KL}(g_t \,\|\, f_t) \le 4$, the score-based flow (3.10) satisfies*

$$\mathrm{KL}(g_t \,\|\, f_t) \;\le\; 2\, e^{Mt} \int_0^t \int_{\mathbb{R}^d} |\nabla \log g_\tau - s_\tau|^2_{A * g_\tau}\, g_\tau \,\mathrm{d}v \,\mathrm{d}\tau, \tag{3.16}$$

where $M = \frac{16 C_1 d^2 (1+T)(1+E)}{\beta^2 \varepsilon} \big(3 + 2\log(C_2\sqrt{\pi/\beta})\big)^2$ with $\varepsilon = \inf_{t \le T}\{\mathrm{tr}\,\mathrm{cov}(g_t) - \|\mathrm{cov}(g_t)\|\} > 0$ and C_1, C_2 universal constants. In particular, as the time-integrated score-matching loss vanishes, so does $\mathrm{KL}(g_t \,\|\, f_t)$.

Proof. We omit the subscript t for convenience of notation. Following the proof of Theorem 3.15, there remains to control the term

$$-\tfrac{1}{2} \int_{\mathbb{R}^d} |\nabla \log g - \nabla \log f|^2_{A*g}\, g \,\mathrm{d}v + \int_{\mathbb{R}^d} (\nabla \log f)^\top (\Sigma_g - \Sigma_f)(\nabla \log g - \nabla \log f)\, g \,\mathrm{d}v.$$

By Theorem 3.12, $A * g - \varepsilon I \succeq 0$. Using Young's inequality, we get

$$\begin{aligned}
&-\tfrac{1}{2} \int_{\mathbb{R}^d} \big|\nabla \log \tfrac{g}{f}\big|^2_{A*g}\, g \,\mathrm{d}v + \int_{\mathbb{R}^d} (\nabla \log f)^\top (\Sigma_g - \Sigma_f)\, \nabla \log \tfrac{g}{f}\, g \,\mathrm{d}v \\
&\le -\tfrac{1}{2} \int \big|\nabla \log \tfrac{g}{f}\big|^2_{A*g}\, g \,\mathrm{d}v + \tfrac{\varepsilon}{2} \int \big|\nabla \log \tfrac{g}{f}\big|^2 g \,\mathrm{d}v + \tfrac{1}{2\varepsilon} \int \big|(\Sigma_g - \Sigma_f)\, \nabla \log f\big|^2 g \,\mathrm{d}v \\
&\le \tfrac{1}{2\varepsilon} \|\Sigma_g - \Sigma_f\|^2 \int |\nabla \log f|^2\, g \,\mathrm{d}v \\
&\le \tfrac{4C_1}{\varepsilon}(1+t)(1+E)\, \|\Sigma_g - \Sigma_f\|^2.
\end{aligned}$$

The last inequality (and constant $C_1 > 0$) follows from Theorem 3.11 and conservation of energy.

We rewrite the difference of second moments in the form of a weighted L^1 norm:

$$\|\Sigma_g - \Sigma_f\|^2 \le \sum_{i,j=1}^d (\Sigma_{g,ij} - \Sigma_{f,ij})^2 = \sum_{i,j=1}^d \Big(\int v_i v_j (g - f) \,\mathrm{d}v \Big)^2 \le d^2 \Big(\int |v|^2\, |g - f| \,\mathrm{d}v \Big)^2.$$

We are now in a position to use the weighted CKP inequality from [21],

$$\int_{\mathbb{R}^d} \phi(v)|g-f|\,\mathrm{d}v \le \Big(3+2\log\int e^{2\phi(v)}f\,\mathrm{d}v\Big)\Big(\mathrm{KL}(g\,\|\,f)^{1/2}+\tfrac{1}{2}\mathrm{KL}(g\,\|\,f)\Big),$$

with $\phi(v)=\beta|v|^2$. By Theorem 3.1, there is C_2 such that $f(v)\le C_2\,e^{-3\beta|v|^2}$, so

$$\int e^{2\beta|v|^2}f(v)\,\mathrm{d}v \le C_2\int e^{-\beta|v|^2}\,\mathrm{d}v = C_2\sqrt{\pi/\beta}.$$

We have

$$\begin{aligned}\beta\int_{\mathbb{R}^d}|v|^2\,|g-f|\,\mathrm{d}v &\le \Big(3+2\log\int e^{2\beta|v|^2}f\,\mathrm{d}v\Big)\big(\mathrm{KL}(g\,\|\,f)^{1/2}+\tfrac{1}{2}\mathrm{KL}(g\,\|\,f)\big)\\ &\le \beta C_3\,\mathrm{KL}(g\,\|\,f)^{1/2}\quad\text{for all } t \text{ such that } \mathrm{KL}(g\,\|\,f)\le 4,\\ C_3 &= \tfrac{2}{\beta}\big(3+2\log(C_2\sqrt{\pi/\beta})\big).\end{aligned}$$

Combining the above estimates, we obtain the differential inequality that holds for $t\le T$ small enough that $\mathrm{KL}(g\,\|\,f)\le 4$:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(g\,\|\,f) \le L(t)+\tfrac{4C_1}{\varepsilon}(1+E)\,d^2C_3^2(1+T)\,\mathrm{KL}(g\,\|\,f) = L(t)+M\,\mathrm{KL}(g\,\|\,f),$$

with

$$L(t)=2\int|\nabla\log g-s|^2_{A*g}\,g\,\mathrm{d}v,\quad M=\tfrac{4C_1}{\varepsilon}(1+E)\,d^2C_3^2(1+T).$$

By generalized Grönwall's inequality, $\mathrm{KL}(g_t\,\|\,f_t)\le e^{Mt}\int_0^t L(\tau)\,\mathrm{d}\tau$, which is (3.16). □

3.2.4 Coulomb KL error bound

The Maxwellian argument above relies on the Fokker–Planck rewriting and on the bilinear scaling of the covariance defect, both of which break down for soft potentials. For the Coulomb case we use the key calculation of Theorem 3.6 directly.

Theorem 3.18 (Coulomb KL error bound)**.** *Let f solve the Coulomb Landau equation ($d=3$, $\gamma=-3$) with $\|f_0\|^2_{H^2_5}<\infty$, and let g satisfy the score-based flow* (3.10) *with $g_0=f_0$. Assume that $\|f_t/g_t\|_\infty<\infty$ and $\sup_{t\ge0}\|g_t\|_\infty<\infty$. Then for any $T>0$,*

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t\,\|\,g_t) \;\le\; 2\left\|\tfrac{f_t}{g_t}\right\|_\infty\int_{\mathbb{R}^3}|\nabla\log g_t-s_t|^2_{A*g_t}\,g_t\,\mathrm{d}v + C\,(1+t)\,\mathrm{KL}(f_t\,\|\,g_t), \tag{3.17}$$

with $C = C(f_0, \sup_t \|g_t\|_\infty)$*, and consequently*

$$\mathrm{KL}(f_T \,\|\, g_T) \;\le\; C\, e^{T^2} \int_0^T \left\| \tfrac{f_t}{g_t} \right\|_\infty \int_{\mathbb{R}^3} |\nabla \log g_t - s_t|^2_{A*g_t}\, g_t \,\mathrm{d}v \,\mathrm{d}t. \tag{3.18}$$

Proof. We omit the time subscript for brevity. Similarly to the computation in Theorem 3.8, we obtain

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) &= \int (U[f] - U[g,s]) \cdot \nabla \log \tfrac{f}{g}\, f \,\mathrm{d}v \\
&= \int (U[g] - U[g,s]) \cdot \nabla \log \tfrac{f}{g}\, f \,\mathrm{d}v + \int (U[f] - U[g]) \cdot \nabla \log \tfrac{f}{g}\, f \,\mathrm{d}v.
\end{aligned}$$

The first term is estimated as follows:

$$\begin{aligned}
&\int (U[g] - U[g,s]) \cdot \nabla \log \tfrac{f}{g}\, f \,\mathrm{d}v \\
&= \iint \big[(\nabla \log g(v) - s(v)) - (\nabla \log g(w) - s(w))\big]^\top A(v-w)\, \nabla \log \tfrac{f(v)}{g(v)}\, f(v)\, g(w) \,\mathrm{d}w \,\mathrm{d}v \\
&= \iint \big[\nabla \log g(v) - s(v)\big]^\top A(v-w)\, \nabla \log \tfrac{f(v)}{g(v)}\, f(v)\, g(w) \,\mathrm{d}w \,\mathrm{d}v \\
&\quad - \iint \big[\nabla \log g(w) - s(w)\big]^\top A(v-w)\, \nabla \log \tfrac{f(v)}{g(v)}\, f(v)\, g(w) \,\mathrm{d}w \,\mathrm{d}v \\
&\le \int |\nabla \log g - s|^2_{A*g}\, f \,\mathrm{d}v + \tfrac{1}{4} \int \big|\nabla \log \tfrac{f}{g}\big|^2_{A*g}\, f \,\mathrm{d}v \\
&\quad + \int |\nabla \log g - s|^2_{A*g}\, f \,\mathrm{d}v + \tfrac{1}{4} \int \big|\nabla \log \tfrac{f}{g}\big|^2_{A*g}\, f \,\mathrm{d}v \\
&\le 2 \left\| \tfrac{f}{g} \right\|_\infty \int |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + \tfrac{1}{2} \int \big|\nabla \log \tfrac{f}{g}\big|^2_{A*g}\, f \,\mathrm{d}v.
\end{aligned}$$

Now, by Theorem 3.6, the second term satisfies

$$\int f\, (U[f] - U[g]) \cdot \nabla \log \tfrac{f}{g} \le -\tfrac{1}{2} \int f \,\big|\nabla \log \tfrac{f}{g}\big|^2_{A*g} + C\, \mathrm{KL}(f \,\|\, g)\, \mathcal{R}(f,g).$$

Finiteness of energy and entropy of g follows from the corresponding finiteness at $t = 0$ and the energy and entropy conservations of Theorem 2.2. Now one obtains

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \le 2 \left\| \tfrac{f}{g} \right\|_\infty \int |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + C\, \mathrm{KL}(f \,\|\, g)\, \mathcal{R}(f,g).$$

Finally, by Theorem 3.5, using the assumption $\|f_0\|^2_{H^2_5} < \infty$, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f \,\|\, g) \le 2 \left\| \tfrac{f}{g} \right\|_\infty \int |\nabla \log g - s|^2_{A*g}\, g \,\mathrm{d}v + C\,(1+t)\, \mathrm{KL}(f \,\|\, g)$$

for some $C = C(f_0, \sup_{0 \le s \le t} \|g\|_\infty)$. Grönwall's inequality completes the proof. □

Remark 3.19. Theorem 3.18 bounds $\mathrm{KL}(f \,\|\, g)$, whereas Theorem 3.17 bounds $\mathrm{KL}(g \,\|\, f)$ – the order of the arguments differs because the Maxwellian argument uses the Fokker–Planck representation of f, while the Coulomb argument uses the bilinear key calculation evaluated against f. In practice the score-matching loss in (3.18) can be evaluated on particles drawn from g, making the bound *a posteriori* computable and providing a quantitative criterion linking training loss to KL error.

3.3 Equilibrium of the Vlasov–Maxwell–Landau system

We now leave the homogeneous setting and characterize the long-time equilibrium of the full VML and VPL systems. These are Theorem 3.20 and Theorem 3.21 from the chapter introduction; their formal verification in Lean 4 forms the case study of Chapter 8.

Theorem 3.20 (VML equilibrium)**.** *Consider the VML system* (2.19)–(2.20) *on* $\mathbb{T}^3 \times \mathbb{R}^3$ *with collision frequency* $\nu > 0$ *and uniform neutralizing ion background* $\rho_{\mathrm{ion}} > 0$*. Let* $f(x, v) > 0$ *be a smooth steady state with finite total energy and entropy and decay sufficient for integration by parts in* v*. Then* f *is a spatially uniform, zero-drift global Maxwellian:*

$$f_\infty(v) = \frac{\rho_{\mathrm{ion}}}{(2\pi T_\infty)^{3/2}} \exp\!\left(-\frac{|v|^2}{2T_\infty}\right), \qquad E_\infty = 0, \qquad B_\infty = \frac{1}{|\mathbb{T}^3|}\int_{\mathbb{T}^3} B_{\mathrm{init}}(x)\,\mathrm{d}x, \tag{3.19}$$

with $T_\infty = \frac{2}{3\,\rho_{\mathrm{ion}}\,|\mathbb{T}^3|}\big(\mathcal{E}_0 - \frac{1}{2}|B_\infty|^2\,|\mathbb{T}^3|\big) > 0$*, where* $\mathcal{E}_0$ *is the initial total energy.*

Theorem 3.21 (VPL equilibrium)**.** *Under the same hypotheses with* $B \equiv 0$ *(so the system reduces to VPL), every steady state is a spatially uniform, drifting global Maxwellian:*

$$f_\infty(v) = \frac{\rho_{\mathrm{ion}}}{(2\pi T_\infty)^{3/2}} \exp\!\left(-\frac{|v - u_\infty|^2}{2T_\infty}\right), \qquad E_\infty = 0, \tag{3.20}$$

with bulk drift and temperature determined by the initial data:

$$u_\infty = \frac{1}{\rho_{\mathrm{ion}}|\mathbb{T}^3|}\iint v\, f_{\mathrm{init}}\,\mathrm{d}v\,\mathrm{d}x, \quad T_\infty = \frac{2}{3\rho_{\mathrm{ion}}|\mathbb{T}^3|}\big(\mathcal{E}_0 - \tfrac{1}{2}\rho_{\mathrm{ion}}|u_\infty|^2|\mathbb{T}^3|\big) > 0.$$

The proofs share the first three steps (which produce a local Maxwellian with constant temperature and uniform bulk velocity) and diverge thereafter.

Proof of Theorem 3.20. **Step 1: Entropy dissipation forces a local Maxwellian.** At equilibrium $\partial_t f = \partial_t E = \partial_t B = 0$, and the Vlasov equation reduces to $v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = \nu Q[f, f]$. Multiplying by $\log f$ and integrating over $\mathbb{T}^3 \times \mathbb{R}^3$, the spatial transport term integrates to zero by the divergence theorem on the torus, and the Lorentz term integrates by parts in v to yield a factor of $\nabla_v \cdot (E + v \times B) = 0$. We are left with

$$\int_{\mathbb{T}^3} \Big(\int_{\mathbb{R}^3} Q[f, f] \log f \, \mathrm{d}v \Big) \, \mathrm{d}x = 0.$$

By the H-theorem the inner integral is non-positive pointwise in x; since its spatial integral vanishes, it must vanish identically. The null space of the Landau operator is spanned by the collisional invariants $\{1, v, |v|^2\}$ [184], so $\log f$ is a linear combination of these: $\log f(x, v) = a(x) + b(x) \cdot v + c(x)|v|^2$. Equivalently, $f(x, v)$ is a local Maxwellian [27, Ch. II.7]

$$f(x, v) = \frac{\rho(x)}{(2\pi T(x))^{3/2}} \exp\left(-\frac{|v - u(x)|^2}{2T(x)} \right), \tag{3.21}$$

with $\rho(x) = \int f \, \mathrm{d}v > 0$, $u(x) = \rho^{-1} \int v f \, \mathrm{d}v$, and $T(x) = \frac{1}{3\rho} \int |v - u|^2 f \, \mathrm{d}v > 0$.

Step 2: Polynomial matching in v. For a local Maxwellian, $Q[f, f] = 0$, so the steady Vlasov equation reduces to $v \cdot \nabla_x \log f + (E + v \times B) \cdot \nabla_v \log f = 0$. Substituting $\log f = a + b \cdot v + c|v|^2$, using $(v \times B) \cdot v = 0$ and $(v \times B) \cdot b = v \cdot (B \times b)$, the equation becomes a polynomial in v:

$$(v \cdot \nabla_x c)|v|^2 + \sum_{i,j=1}^{3} v_i v_j \, \partial_{x_i} b_j + v \cdot (\nabla_x a + 2cE + B \times b) + E \cdot b = 0. \tag{3.22}$$

Each homogeneous degree of v must vanish identically.

Step 3: Macroscopic uniformity. *(a) Cubic in v:* $v \cdot \nabla_x c = 0$ for all v gives $\nabla_x c = 0$, so c is constant; equivalently $T(x) \equiv T_\infty$.

(b) Quadratic in v: $\sum_{i,j} v_i v_j \, \partial_{x_i} b_j = 0$ for all v yields Killing's equation $\partial_{x_i} b_j + \partial_{x_j} b_i = 0$ [27, Eq. 10.6]. Integrating $\sum_{i,j} (\partial_i b_j + \partial_j b_i)^2$ over $\mathbb{T}^3$ and integrating by parts shows $\partial_{x_i} b_j = 0$, so b, and hence $u(x) \equiv u_\infty$, are spatially constant.

(c) Linear in v: With c, b constant, substituting $c = -1/(2T_\infty)$ and $b = u_\infty / T_\infty$ gives the macroscopic force balance

$$\nabla_x \log \rho = \frac{1}{T_\infty} (E + u_\infty \times B). \tag{3.23}$$

Step 4: Ampère's law forces $u_\infty = 0$. Since u_∞ is constant, the current is $J(x) = \rho(x)\, u_\infty$. At steady state Ampère's law reads $\nabla_x \times B = J = \rho\, u_\infty$. Dotting with u_∞, $u_\infty \cdot (\nabla_x \times B) = \rho |u_\infty|^2$. Using the identity $\nabla_x \cdot (B \times u_\infty) = u_\infty \cdot (\nabla_x \times B)$ (since $\nabla_x \times u_\infty = 0$) and integrating over $\mathbb{T}^3$, the divergence integrates to zero by periodicity, leaving $0 = |u_\infty|^2 \int_{\mathbb{T}^3} \rho \, \mathrm{d}x$. Since $\rho > 0$, $|u_\infty|^2 = 0$, hence $u_\infty = 0$.

Step 5: Maximum principle forces $\rho \equiv \rho_{\mathrm{ion}}$. With $u_\infty = 0$, (3.23) reduces to $\nabla_x \log \rho = E/T_\infty$. Taking the divergence and substituting Gauss's law $\nabla_x \cdot E = \rho - \rho_{\mathrm{ion}}$ yields the nonlinear Poisson–Boltzmann equation

$$T_\infty \Delta_x (\log \rho) = \rho - \rho_{\mathrm{ion}}. \tag{3.24}$$

Compactness of $\mathbb{T}^3$ and smoothness of ρ guarantee that $\log \rho$ attains a global maximum at some $x_{\max}$ and a global minimum at $x_{\min}$. At $x_{\max}$, $\Delta_x \log \rho \le 0$, so $\rho(x_{\max}) \le \rho_{\mathrm{ion}}$; at $x_{\min}$, $\Delta_x \log \rho \ge 0$, so $\rho(x_{\min}) \ge \rho_{\mathrm{ion}}$. Combining, $\rho_{\mathrm{ion}} \le \rho(x_{\min}) \le \rho(x) \le \rho(x_{\max}) \le \rho_{\mathrm{ion}}$, forcing $\rho \equiv \rho_{\mathrm{ion}}$ and consequently $E \equiv 0$.

Step 6: Magnetic field is constant. With $u_\infty = 0$, the current vanishes and Ampère's law gives $\nabla_x \times B = 0$. Combined with $\nabla_x \cdot B = 0$, the vector Laplacian $\Delta_x B = \nabla_x (\nabla_x \cdot B) - \nabla_x \times (\nabla_x \times B) = 0$. Multiplying by B and integrating over $\mathbb{T}^3$ gives $\int |\nabla_x B|^2 \, \mathrm{d}x = 0$, so B is constant: $B(x) \equiv B_\infty$. Note that for the isotropic equilibrium (3.19), $\nabla_v f_\infty \propto v\, f_\infty$, so the Lorentz term $(v \times B_\infty) \cdot \nabla_v f_\infty \propto (v \times B_\infty) \cdot v = 0$ identically; hence any $B_\infty \in \mathbb{R}^3$ leaves the equilibrium unchanged.

Step 7: Conservation laws fix B_∞ and T_∞. Faraday's law $\partial_t B = -\nabla_x \times E$ implies that the spatial mean of B is conserved (its integral over $\mathbb{T}^3$ is zero by periodicity), so $B_\infty = |\mathbb{T}^3|^{-1} \int B_{\mathrm{init}} \, \mathrm{d}x$. Total energy conservation, $\mathcal{E}_0 = \frac{3}{2} \rho_{\mathrm{ion}} T_\infty |\mathbb{T}^3| + \frac{1}{2} |B_\infty|^2 |\mathbb{T}^3|$, fixes T_∞ as in the statement. Positivity of T_∞ follows from Jensen's inequality: $|B_\infty|^2 \le |\mathbb{T}^3|^{-1} \int |B_{\mathrm{init}}|^2 \, \mathrm{d}x$, so the magnetic energy of the uniform field cannot exceed the initial magnetic energy. □

Proof of Theorem 3.21. Steps 1–3 of the previous proof carry over verbatim with $B \equiv 0$: the magnetic field plays no role in the entropy dissipation argument (Step 1) or in the cubic and quadratic coefficients of the polynomial matching (Steps 2–3). The conclusion is again

a local Maxwellian with constant temperature and uniform bulk velocity satisfying the force balance

$$\nabla_x \log \rho = E/T_\infty. \tag{3.25}$$

Step 4 absent. In the absence of a magnetic field there is no Ampère's law to constrain u_∞. The $\mathcal{O}(|v|^0)$ coefficient of the polynomial (3.22) reduces to $E \cdot b/T_\infty = 0$, which once $E = 0$ is established imposes no constraint on u_∞.

Step 5 unchanged. From (3.25) and Gauss's law one obtains the same Poisson–Boltzmann equation (3.24), and the maximum principle forces $\rho \equiv \rho_{\text{ion}}$ and $E \equiv 0$.

Step 6 absent.

Determination of u_∞ and T_∞. The VPL system is Galilean invariant, so a uniform bulk drift produces no back-reacting field. Total kinetic momentum is therefore conserved $\frac{\mathrm{d}}{\mathrm{d}t}\iint v f \,\mathrm{d}v\,\mathrm{d}x = 0$, and equating equilibrium momentum to the initial value gives $u_\infty = (\rho_{\text{ion}}|\mathbb{T}^3|)^{-1}\iint v f_{\text{init}}\,\mathrm{d}v\,\mathrm{d}x$. Total energy conservation then determines $T_\infty = \frac{2}{3\rho_{\text{ion}}|\mathbb{T}^3|}(\mathcal{E}_0 - \frac{1}{2}\rho_{\text{ion}}|u_\infty|^2|\mathbb{T}^3|)$. □

Remark 3.22. The static characterization above is novel for the electromagnetically coupled system: each individual ingredient is classical, but their assembly is not. [53] prove a related result for the inhomogeneous Boltzmann equation *without* fields; [74, 75] study *dynamical* stability of small perturbations near an assumed Maxwellian for the VML and VPL systems, but state the macroscopic constraints inside linearized energy estimates rather than as standalone uniqueness results. The new ingredients here are the Lorentz-force structure $(v \times B)\cdot v = 0$ – which makes the polynomial matching of Step 3 carry over from the unforced case unchanged – the Ampère-law integration in Step 4 forcing $u_\infty = 0$, and the Gauss-law closure of Step 5. The full assembly is the content of Theorem 3.20 and is verified formally in Chapter 8.

Chapter 4

SCORE-BASED DETERMINISTIC DENSITY SAMPLING

This chapter opens Part II of the dissertation, which shifts the focus of score-based transport modeling from the collisional plasma setting of Chapters 2 and 3 to the problem of sampling from a probability distribution in this Chapter and density and score estimation in Chapter 5. In Chapter 2 we used SBTM to evolve a particle ensemble according to the homogeneous Landau equation and the Vlasov–Maxwell–Landau system; the unknown was the time-evolving plasma density $f(t, v)$, and the score $\nabla \log f$ was learned on the fly to drive the deterministic particle ODE. Here we apply the same machinery to a different task: given only the score $\nabla \log \pi$ of an unnormalized target density π – and *no* samples from it – produce samples whose empirical distribution converges to π. This is a strictly harder problem than the diffusion generative modeling on which modern score-based methods were originally built, since it must be solved without the data that those methods rely on. The approach we take is to run the Wasserstein gradient flow of relative entropy $\mathrm{KL}(\cdot \,\|\, \pi)$ as a deterministic interacting particle system, with the score $\nabla \log f_t$ of the evolving particle density learned online by a neural network – exactly as in Chapter 2, but with the Landau collision operator replaced by the linear $\nabla \log \pi - \nabla \log f_t$ drift of the Fokker–Planck equation associated with the target.

The chapter is mostly self-contained. The implicit score-matching loss already derived in Section 2.1.3 reappears in Section 4.2.1; we repeat the short integration-by-parts argument here so that this chapter can be read independently of Part I. The convergence analysis in Section 4.3 is new relative to Chapter 2: it is built around a coupled gradient flow on both the particles and the neural network weights, and uses neural tangent kernel theory to control the dynamically changing score-matching loss. This analysis is what allows us to prove

the optimal entropy dissipation rate for the joint particle–network system. The numerical experiments in Section 4.4 demonstrate the algorithm on log-concave and multimodal targets in low dimension and on MNIST in 784 dimensions. Chapter 5 returns to the question of *how* the score is estimated, replacing the per-problem online training used here with an amortized Transformer-based estimator.

This chapter follows [93] almost verbatim.

4.1 Introduction

Diffusion generative modeling (DGM) [160, 164] has emerged as a powerful set of techniques to generate "more of the same thing", i.e., given many samples from some unknown distribution π, train a model to generate more samples from π. While the SDE-based generation is commonly used in practice, its deterministic counterpart, termed "probability flow ODE" by [164], offers several practical advantages – higher order solvers [82], better dimension dependence [37], and interpolation in the latent space [158]. The two processes are given [39] by the reverse of the Ornstein–Uhlenbeck (OU) process and the corresponding ODE, respectively:

$$\mathrm{d}X_t = (X_t + 2\nabla \log f_t(X_t))\ \mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t \qquad \text{(DGM SDE)}$$

$$\mathrm{d}X_t = (X_t + \nabla \log f_t(X_t))\ \mathrm{d}t, \qquad \text{(DGM ODE)}$$

where B_t is the Brownian motion and f_t is the law of the OU process. The score $\nabla \log f_t$ is learned by running the OU process from π to $\mathcal{N}(0, I_d)$, which crucially depends on having many samples from π. In this work we pose and attempt to answer the following question: **How to deterministically sample an unnormalized density π in the absence of samples, using techniques of diffusion generative modeling?** We propose an analogue of (DGM ODE) when no samples from π are available. The absence of samples from π makes sampling a much harder problem than DGM. Indeed, DGM can be reduced to unnormalized density sampling by estimating the score $\nabla \log \pi$ on the samples, but the converse is not true; recently [77] proved the exponential in d query complexity lower bound for density sampling,

while DGM is solvable in polynomial time [39].

Contributions. Our main contributions are a deterministic sampling algorithm, scalable to high dimensions, and a proof of fast convergence under the log-Sobolev inequality. Unlike the classical Langevin dynamics, our algorithm produces smooth deterministic trajectories, and gives access to the otherwise intractable score $\nabla \log f_t$ allowing for convergence estimation via the relative Fisher information. Unlike other deterministic interacting particle systems, the memory and runtime scale linearly in the number of particles. The convergence analysis is based on a system of coupled gradient flows, one for the samples and one for the weights of the neural network. This is a novel framework that can be of independent interest as a general technique for analyzing convergence of NN-based approximations of gradient flows. We are not aware of any other work that uses the neural tangent kernel for analyzing dynamically changing loss. Our main theoretical result is Theorem 4.10.

Related work. Sampling from unnormalized densities traditionally relies on stochastic algorithms such as Langevin dynamics, whose fast convergence under isoperimetry assumptions is well-understood [48, 194, 181, 43]. Deterministic alternatives, notably Stein Variational Gradient Descent (SVGD) [120, 55], rely heavily on handcrafted kernels, resulting in limited theoretical guarantees and empirical success, especially in high dimensions [216]. Inspired by deterministic probability-flow ODEs from diffusion generative modeling [86, 160, 164, 37, 82], which often outperform their stochastic counterparts due to high-order integrators, better dimension dependence, and latent-space interpolation capabilities [158, 124], Boffi and Vanden-Eijnden proposed [20] Score-Based Transport Modeling (SBTM) as a general method of solving the Fokker–Planck equation with a deterministic interacting particle system, building on neural approximations of gradient flows [200, 59]. However, it has not been used for sampling. Our work fills this gap by adapting the method for the problem of sampling and by providing entropy dissipation guarantees and proving rapid convergence for the coupled particle system dynamics, integrating smoothly with recent annealing approaches like those

proposed by [138, 45, 29].

Organization. The rest of the chapter is organized as follows. In Section 4.2 we introduce the main algorithm, building on Score-Based Transport Modeling [20]. In Section 4.3 we prove that the resulting dynamics achieve the optimal convergence rate. In Section 4.4 we verify convergence in several numerical experiments. The chapter is concluded in Section 4.5.

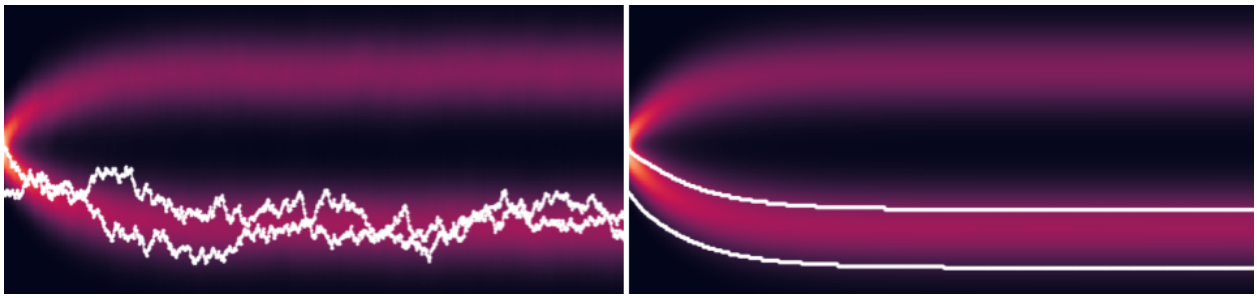


Figure 4.1: Left: Langevin dynamics (stochastic). Right: ours (deterministic). The deterministic algorithm has the same marginal distributions as the stochastic one but with smooth trajectories. In this plot both algorithms interpolate between the unit Gaussian and a mixture of two Gaussians.

4.2 Deterministic sampling algorithm

Diffusion generative modeling (DGM SDE) and (DGM ODE) converges quickly because it is the reverse of a fast process, namely the OU process. In the absence of samples from π, there is no known numerically tractable process that can be run from π to f_0 and reversed, and the dimension-exponential lower bound of [77] suggests that it cannot exist. Thus, the standard approach to classical sampling is to greedily minimize a divergence, most commonly relative entropy $\mathrm{KL}(f_t \,\|\, \pi) := \mathbb{E}_{f_t} \log \frac{f_t}{\pi}$, between f_t and π at every time step. In continuous time, this is called a Wasserstein gradient flow (GF). The Wasserstein GF on $\mathrm{KL}(\cdot \,\|\, \pi)$ can be implemented stochastically or deterministically, mimicking equations (DGM SDE) and

(DGM ODE):

$$\mathrm{d}X_t = \nabla \log \pi(X_t)\,\mathrm{d}t + \sqrt{2}\,\mathrm{d}B_t, \quad B_t := \text{Brownian motion}, \tag{GF SDE}$$

$$\mathrm{d}X_t = \nabla \log \pi(X_t)\,\mathrm{d}t - \nabla \log f_t(X_t)\,\mathrm{d}t, \quad f_t := \mathrm{law}(X_t). \tag{GF ODE}$$

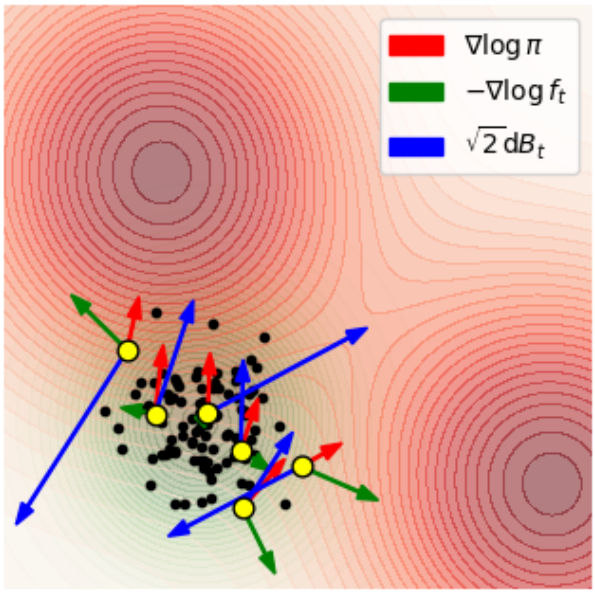


Figure 4.2: Visualization of (GF ODE). Particles are pulled towards target π and away from particle density f_t. Brownian motion acts randomly.

Remarkably, the distributions of X_t for a fixed t in equations (GF SDE) and (GF ODE) match exactly. See [97] for the rigorous derivation and connection to Optimal Transport and see Figure 4.1 for a visualization. While the former equation is well-studied under the names Langevin Dynamics or ULA, the latter can be more appealing due to determinism, for example due to smooth trajectories, higher order time integrators, and potentially better sample complexity. In this work we simulate the (GF ODE). The term $-\nabla \log f_t$ in the ODE provides deterministic exploration by making samples $X_t^1, \ldots, X_t^n$ repel from each other, since more particles appear where f_t is larger. We visualize the forces $\nabla \log \pi, -\nabla \log f_t$ and the noise $\sqrt{2}\,\mathrm{d}B_t$ in Figure 4.2.

Remark 4.1. While equations (DGM ODE) and (GF ODE) look similar, there is a big difference in how f_t is defined. In (DGM ODE), f_t is given by the OU process started from π. In (GF ODE) f_t follows the gradient flow started from f_0, usually chosen to be Gaussian $\mathcal{N}(0, 1)$.

4.2.1 Score-Based Transport Modeling

Score-based transport modeling (SBTM) was introduced in [20] as a method of solving Fokker–Planck equations, of which (GF ODE) is a special case. Additionally, SBTM was recently successfully employed to solve other Fokker–Planck-type equations [90, 129, 83].

The core idea of SBTM is to approximate $\nabla \log f_t$ with a neural network s^{Θ_t} trained on the sample $X_t^1, \dots, X_t^n$. The training is done by minimizing the score matching loss from [86, 160]. The SBTM dynamics approximate (GF ODE) and are given by the interacting particle system

$$\begin{aligned} \frac{\mathrm{d}X_t}{\mathrm{d}t} &= \nabla \log \pi(X_t) - s^{\Theta_t}(X_t), \quad X_t = (X_t^1, \dots, X_t^n), \quad X_0^i \sim f_0 \text{ iid} \\ \frac{\mathrm{d}\Theta_t}{\mathrm{d}t} &= -\eta \nabla_{\Theta_t} L(s^{\Theta_t}, f_t), \quad X_t \sim f_t, \end{aligned} \tag{SBTM}$$

where $L(s, f_t) = \mathbb{E}_{f_t} \|s - \nabla \log f_t\|^2$ is the dynamically changing score-matching loss [86] and $\eta(t)$ controls the amount of NN training relative to sampling dynamics. The resulting interacting particle system is a coupled gradient flow, where f_t approximates the Wasserstein GF on $\mathrm{KL}(f_t \,\|\, \pi)$, and Θ_t undergoes the Euclidean GF on the dynamically changing loss $L(s^{\Theta_t}, f_t)$, while η controls the relative speed of the two flows. If the loss $L(s, f)$ is identically zero, then SBTM dynamics exactly recover the (GF ODE) dynamics. Unlike most other interacting particle systems used for sampling [120, 118, 45], in SBTM the particles $X_t^1, \dots, X_t^n$ interact via the neural network s^{Θ_t}, which makes the memory and runtime scale linearly, and provides much better dimension scaling.

Remark 4.2 (Comparison with Langevin Dynamics). In equation (GF SDE), also called Langevin Dynamics, particles X_t^i are iid, whereas in SBTM dynamics the particles interact via the NN s^{Θ_t}. Additionally, while particles in Langevin Dynamics undergo Brownian

motion, resulting in continuous but non-differentiable trajectories, particles in SBTM have smooth trajectories. Thus, while the time- and particle-marginals of Langevin and SBTM dynamics coincide in the limit $L(s, f) \to 0$, the joint distributions are very different.

While the loss $L(s, f_t) = \mathbb{E}_{f_t}\|s - \nabla \log f_t\|^2$ is not tractable due to the score term $\nabla \log f_t$, for smooth f, it is equivalent to the loss $\mathbb{E}_{f_t}\|s\|^2 + 2\nabla \cdot s$ via integration by parts:

$$\begin{aligned}\mathbb{E}_f\|s - \nabla \log f\|^2 &= \mathbb{E}_f\left(\|s\|^2 - 2s \cdot \nabla \log f\right) + \mathbb{E}_f\|\nabla \log f\|^2 && \text{(expand } \|\cdot\|^2\text{)} \\ &= \mathbb{E}_f\left(\|s\|^2 + 2\nabla \cdot s\right) + \text{const}(s) && \text{(integrate by parts).}\end{aligned}$$

The latter loss is tractable, and is readily approximated by evaluating on a sample $X^1, \ldots, X^n$ from f:

$$\mathbb{E}_f\left(\|s\|^2 + 2\nabla \cdot s\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\|s(X^i)\|^2 + 2\nabla \cdot s(X^i)\right) + \text{const}(s) \quad \text{if} \quad f = \frac{1}{n}\sum_{i=1}^{n}\delta_{X^i}.$$

While we use automatic differentiation to compute the divergence $\nabla \cdot s$ in the low-dimensional experiments, we employ Hutchinson's trace estimator [84] $\mathbb{E}\left[\varepsilon^T J\varepsilon\right]$ where ε follows the Rademacher distribution. This is because computing $\nabla \cdot s$ with autodiff takes $O(dC_s)$ time, where C_s is the cost of one NN evaluation, whereas the Hutchinson's trace estimator takes $O(mC_s)$, where m is the number of random projections.

Remark 4.3. The neural network trains on the particles that were produced following the same neural network, as opposed to the true solution of the Wasserstein GF (GF ODE). The commonsense intuition makes one suspect that local errors may accumulate exponentially. Miraculously, this does not happen, see Theorem 4.6.

For ease of reproducibility, we provide the pseudocode for SBTM: In practice we use the Adamw optimizer for the gradient descent step and use the denoised score matching loss for high-dimensional experiments. To initialize the NN $s^\Theta \approx \arg\min L(s, f_0)$ we perform gradient descent on the standard MSE loss

$$\frac{1}{n}\sum_{i=1}^{n}\|s(X_0^i) - \nabla \log f_0(X_0^i)\|^2,$$

since we have access to the initial data f_0. For example, in most experiments below the initial data is the standard Gaussian, making the initial score $\nabla \log f_0(x) = -x$.

Algorithm 3 SBTM for deterministic density sampling.

Require: Initial density f_0, target score $\nabla \log \pi$, sample size n, final time T, time step Δt, learning rate η.

Sample $\{X^i\}_{i=1}^n$ iid from f_0.

Initialize NN $s^\Theta \approx \arg\min L(s, f_0)$.

$t \leftarrow 0$.

while $t < T$ **do**

$t \leftarrow t + \Delta t$.

for $k = 1, \dots, K$ (flow Θ) **do**

$\Theta \leftarrow \Theta - \eta \cdot \nabla_\Theta L(s^\Theta, f_t)$.

end for

for $i = 1, \dots, n$ (flow f) **do**

$X^i \leftarrow X^i + \Delta t\big(\nabla \log \pi(X^i) - s^\Theta(X^i)\big)$.

end for

end while

Output particle locations $X^1, \dots, X^n$.

4.3 Convergence analysis

How quickly does f_t converge to π in (SBTM)**?** To answer this question we study the decay rate of the relative entropy $\mathrm{KL}(f_t \,\|\, \pi)$ [43]. Our strategy is to break down the change in $\mathrm{KL}(f_t \,\|\, \pi)$ into the change due to the density f_t and the change due to the neural network s^{Θ_t}, effectively splitting the coupled GF into two parts. Theorem 4.5 does the former, and Theorem 4.8 does the latter. Theorem 4.10 brings both pieces together. Theorem 4.12 extends Theorem 4.5 to the annealed setting.

4.3.1 Entropy dissipation

First, recall the following classical calculation [43] that quantifies the rate of relative entropy decay for the Wasserstein GF on relative entropy:

Theorem 4.4. *If f_t follows the Wasserstein GF on $\mathrm{KL}(\cdot \,\|\, \pi)$ then relative entropy dissipates at the rate of the relative Fisher information:*

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) = \mathrm{F}(f_t \,\|\, \pi) := \mathbb{E}_{f_t}\left\|\nabla \log \frac{f_t}{\pi}\right\|^2.$$

Additionally, if π satisfies the log-Sobolev inequality with constant α (e.g. if π is α-log-concave) then

$$\mathrm{KL}(f_t \,\|\, \pi) \le \mathrm{KL}(f_0 \,\|\, \pi) e^{-\frac{1}{\alpha}t}.$$

Intuitively, if $s_t \approx \nabla \log f_t$, SBTM should converge at the nearly optimal rate

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) = \mathrm{F}(f_t \,\|\, \pi)\,.$$

Indeed, this holds if the score matching loss is small:

Theorem 4.5 (Small loss guarantees optimal entropy dissipation)**.** *If f_t is the density of X_t, which follows*

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = \nabla \log \pi(X_t) - s_t(X_t), \quad X_0 \sim f_0,$$

for any time-dependent vector field s_t, then

$$\begin{aligned} -\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) &= \mathrm{F}(f_t \,\|\, \pi) - \mathbb{E}_{f_t}\left\langle s_t - \nabla \log f_t,\, \nabla \log \frac{f_t}{\pi}\right\rangle \\ &\ge \frac{1}{2}\,\mathrm{F}(f_t \,\|\, \pi) - \frac{1}{2}L(s_t, f_t). \end{aligned} \tag{4.1}$$

In particular, if $L(s_t, f_t) \le \frac{1}{2}\,\mathrm{F}(f_t \,\|\, \pi)$, then

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) \ge \frac{1}{4}\,\mathrm{F}(f_t \,\|\, \pi)\,.$$

While Theorem 4.5 looks simple, it elucidates a remarkable property of SBTM:

Remark 4.6. Integrating (4.1) in time, one obtains

$$\mathrm{KL}(f_T \,\|\, \pi) \le \mathrm{KL}(f_0 \,\|\, \pi) - \frac{1}{2}\int_0^T \mathrm{F}(f_t \,\|\, \pi)\, \mathrm{d}t + \frac{1}{2}\int_0^T L(s_t, f_t)\, \mathrm{d}t.$$

Since there is no exponential term e^T in the bound, local errors do not accumulate. Importantly, f_t is the law of X_t, where X_t follows (SBTM); it is not the (unknown) true solution of (GF ODE).

Proof of Theorem 4.5. If f_t is the density of X_t, where X_t satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t} X_t = \nabla \log \pi(X_t) - s_t(X_t)$$

then f_t satisfies the Fokker–Planck equation

$$\partial_t f_t + \nabla \cdot (f_t(\nabla \log \pi - s_t)) = 0.$$

Thus, we may explicitly compute the relative entropy dissipation rate as

$$\begin{aligned} -\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} f_t \log \frac{f_t}{\pi}\, \mathrm{d}x &= \int_{\mathbb{R}^d} \left\langle s_t - \nabla \log \pi,\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \, \mathrm{d}x + \int_{\mathbb{R}^d} f_t \partial_t \log f_t \, \mathrm{d}x \\ &= \int_{\mathbb{R}^d} \left\langle s_t - \nabla \log \pi,\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \, \mathrm{d}x, \end{aligned}$$

where we used the fact that $\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} f_t \, \mathrm{d}x = 0$ for the last equality.

By taking $s_t = \nabla \log f_t$ exactly, we recover the proof of the classical result Theorem 4.4. Otherwise, adding and subtracting $\nabla \log f_t$ in the integrand, we get

$$\begin{aligned} -\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathbb{R}^d} f_t \log \frac{f_t}{\pi}\, \mathrm{d}x &= \int_{\mathbb{R}^d} \|\nabla \log f_t - \nabla \log \pi\|^2 f_t \, \mathrm{d}x \\ &\quad - \int_{\mathbb{R}^d} \langle \nabla \log f_t - s_t,\, \nabla \log f_t - \nabla \log \pi \rangle \, f_t \, \mathrm{d}x \\ &\ge \frac{1}{2}\int_{\mathbb{R}^d} \|\nabla \log f_t - \nabla \log \pi\|^2 f_t \, \mathrm{d}x - \frac{1}{2}\int_{\mathbb{R}^d} \|\nabla \log f_t - s_t\|^2 f_t \, \mathrm{d}x \\ &= \frac{1}{2}\,\mathrm{F}(f_t \,\|\, \pi) - \frac{1}{2} L(s_t, f_t). \end{aligned}$$

The third line is by Young's inequality. □

Remark 4.7. The Young bound $\frac{1}{2}\,\mathrm{F}(f_t \,\|\, \pi) - \frac{1}{2} L(s_t, f_t)$ is conservative. Using Cauchy–Schwarz instead of Young gives

$$\Big| -\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) - \mathrm{F}(f_t \,\|\, \pi) \Big| \le \sqrt{L(s_t, f_t)\ \mathrm{F}(f_t \,\|\, \pi)},$$

so that $-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) \to \mathrm{F}(f_t \,\|\, \pi)$ at rate $O(\sqrt{L})$ as $L \to 0$. In numerical experiments we observe $-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) \approx \mathrm{F}(f_t \,\|\, \pi)$, indicating small loss.

Now we show how to ensure that the approximation $s_t \approx \nabla \log f_t$ remains accurate despite changing f_t. We achieve it by breaking up $\frac{\mathrm{d}}{\mathrm{d}t} L(s_t, f_t)$ into the change due to s_t training and the change due to f_t evolving. Informally speaking, the NN training needs to be faster than the evolution of f_t.

4.3.2 Bounding score matching loss

With sufficient size and amount of training a neural network can fit arbitrary data, including dynamically changing data. The next theorem shows that the score-matching loss does not increase, given sufficient training.

Theorem 4.8 (Sufficient training guarantees bounded loss)**.** *Assume that f_t is any time-dependent density, $X_t^1, \dots, X_t^n$ are distributed according to f_t, and the neural network $s_t = s^{\Theta_t}$ is trained with gradient descent*

$$\frac{\mathrm{d}\Theta_t}{\mathrm{d}t} = -\eta \nabla_{\Theta_t} L^n(s_t, f_t), \quad L^n(s_t, f_t) := \frac{1}{n}\sum_{i=1}^{n} \|s(X_t^i) - \nabla \log f(X_t^i)\|^2,$$

and is such that the Neural Tangent Kernel (NTK)

$$H_{\alpha,\beta}^{i,j}(t) = \sum_{k=1}^{N} \nabla_{\theta_k} s_t^\alpha(X_t^i) \nabla_{\theta_k} s_t^\beta(X_t^j), \quad s(x) = (s^1(x), \dots, s^d(x)),$$

is lower bounded by $\lambda = \lambda(t) > 0$, i.e. $\|Hv\|^2 \ge \lambda \|v\|^2$. Then as long as

$$\eta(t) \ge \frac{n}{\lambda} \left.\frac{\partial}{\partial \tau}\right|_{\tau=t} \log L^n(s_t, f_\tau),$$

the loss $L^n(s_t, f_t)$ is non-increasing in time.

Remark 4.9. The assumption that the NTK is lower bounded in Theorem 4.8 is non-trivial, but holds under relatively mild assumptions [98]; the most restrictive one is that the NN size is superlinear in the number of data. While the current state-of-the-art NTK theory cannot handle the infinite training data regime $n \to \infty$ with finite NN size, as better NTK theory becomes available, the assumptions in Theorem 4.8 can be weakened.

Proof of Theorem 4.8. We start with an elementary computation based on the chain rule:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}L(s_t, f_t) &= \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_\tau, f_t) + \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_t, f_\tau), \\
\left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_\tau, f_t) &= \nabla_\Theta L \cdot \frac{\mathrm{d}}{\mathrm{d}t}\Theta \\
&= -\eta \sum_{k=1}^{N} (\nabla_{\theta_k} L)^2 \\
&= -\frac{\eta}{n^2} \sum_{k=1}^{N} \left([s(X^i) - \nabla \log f(X^i)] \cdot \nabla_{\theta_k} s(X^i) \right)^2 \\
&= -\frac{\eta}{n^2} \sum_{i,j=1}^{n} \sum_{\alpha,\beta=1}^{d} [s_\alpha(X^i) - \nabla_\alpha \log f(X^i)] H^{i,j}_{\alpha,\beta} [s_\beta(X^j) - \nabla_\beta \log f(X^j)] \\
&= -\frac{\eta}{n^2} \|s - \nabla \log f\|_H^2,
\end{aligned}$$

where

$$H^{i,j}_{\alpha,\beta} = \sum_{k=1}^{N} \nabla_{\theta_k} s_\alpha(X^i) \nabla_{\theta_k} s_\beta(X^j)$$

is called the Neural Tangent Kernel. Its lowest eigenvalue determines the convergence speed of gradient descent. If $\|Hv\|^2 \geq \lambda \|v\|^2$, then

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}L(s_t, f_t) &= \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_\tau, f_t) + \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_t, f_\tau) \\
&\leq -\frac{\eta\lambda}{n} L(s_t, f_t) + \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} L(s_t, f_\tau).
\end{aligned}$$

Thus, the loss is non-increasing if

$$\eta(t) \geq \frac{n}{\lambda} \left.\frac{\partial}{\partial\tau}\right|_{\tau=t} \log L(s_t, f_\tau). \qquad \square$$

Combining Theorems 4.4 and 4.8 shows that the convergence rate of SBTM matches the convergence rate of the true GF (GF ODE). This is the main theoretical guarantee of our sampling method.

Theorem 4.10. *Suppose that X_t and Θ_t follow* (SBTM) *and η and H are as in Theorem 4.8. If the initial loss is small and the true loss L is well-approximated by the training loss L^n*

$$L^n(s_0, f_0) \leq \frac{1}{4}\varepsilon, \quad \varepsilon := \inf_{t \leq T} \mathrm{F}(f_t \,\|\, \pi)\,, \tag{4.2}$$

$$|L^n(s_t, f_t) - L(s_t, f_t)| \leq \frac{1}{4}\varepsilon, \tag{4.3}$$

then relative entropy dissipates at the optimal rate:

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) \geq \frac{1}{4}\,\mathrm{F}(f_t \,\|\, \pi)\,.$$

If π satisfies the log-Sobolev inequality with constant α then convergence is exponential:

$$\mathrm{KL}(f_t \,\|\, \pi) \leq \mathrm{KL}(f_0 \,\|\, \pi) e^{-\frac{1}{4\alpha}t}.$$

Remark 4.11. Since $s_t \approx \nabla \log f_t$, relative Fisher information may be approximated by

$$\mathrm{F}(f_t \,\|\, \pi) \approx \mathrm{F}^n(f_t \,\|\, \pi) := \frac{1}{n}\sum_{i=1}^{n} \|s(X^i) - \nabla \log \pi(X^i)\|^2.$$

One may stop the sampling when $\mathrm{F}^n(f_t \,\|\, \pi) \leq \varepsilon$ for some predefined ε. This makes condition (4.2) practical.

If particles $X^1(t), \ldots, X^n(t)$ were independent, the Law of Large Numbers would imply

$$\lim_{n\to\infty} |L^n(s_t, f_t) - L(s_t, f_t)| = 0,$$

satisfying condition (4.3) for large enough n. Further, for interacting particle systems propagation of chaos often holds, namely that in the limit $n \to \infty$ particles X_t^i become independent. In numerical experiments, we treat L^n as L, and confirm the optimal rate of relative entropy dissipation.

4.3.3 Annealed dynamics

Classical sampling can be broken down into three distinct parts: mode discovery, mode weighting, and mode approximation. While Langevin dynamics performs mode approximation very fast, both mode discovery and mode weighting are challenging. For example, in the absence of samples from π, the mere discovery of a mode with support of small width h in d dimensions requires $\Omega\left(h^{-d}\right)$ function evaluations [77]. Empirically, it helps to pick an annealing between f_0 and π to "guide" f_t, similar to how the forward OU process guides the reverse process in DGM. We use the geometric annealing $\pi_t \propto f_0^{1-t}\pi^t$ [138, 136, 30] and the dilation annealing $\pi_t(x) \propto \pi(x/t)$ [29].

One obtains a similar entropy dissipation estimate in annealed dynamics. Taking $\pi_t = \pi$ recovers Theorem 4.5.

Theorem 4.12. *If π_t is any time-dependent density, s_t any time-dependent vector field, and*

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = \nabla \log \pi_t(X_t) - s(X_t),$$

then

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) = \mathbb{E}_{f_t}\left\langle s_t - \nabla \log \pi_t,\, \nabla \log \frac{f_t}{\pi} \right\rangle.$$

In particular, as $L(s_t, f_t) \to 0$,

$$-\frac{\mathrm{d}}{\mathrm{d}t}\mathrm{KL}(f_t \,\|\, \pi) \to \mathbb{E}_{f_t}\left\langle \nabla \log \frac{f_t}{\pi_t},\, \nabla \log \frac{f_t}{\pi} \right\rangle \tag{4.4}$$

at the rate $O(\sqrt{L(s_t, f_t)})$.

Theorem 4.12 gives a way to test for $L(s_t, f_t) = 0$ by testing equality (4.4). This is important, because the true loss $L(s_t, f_t) = \mathbb{E}_f\|s - \nabla \log f\|^2$ cannot be computed from a finite sample $X_t^1, \ldots, X_t^n$ of f_t since $\nabla \log f_t$ is unknown. Empirically, we confirm that (4.4) holds, indicating small loss – see Figure 4.8.

Proof. If f_t is the density of X_t, where X_t satisfies

$$\frac{\mathrm{d}}{\mathrm{d}t}X_t = \nabla \log \pi_t(X_t) - s_t(X_t),$$

then f_t satisfies the Fokker–Planck equation

$$\partial_t f_t + \nabla \cdot (f_t(\nabla \log \pi_t - s_t)) = 0.$$

Thus, we may explicitly compute the relative entropy dissipation rate as

$$\begin{aligned} -\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{R}^d} f_t \log \frac{f_t}{\pi} \,\mathrm{d}x &= \int_{\mathbb{R}^d} \left\langle s_t - \nabla \log \pi_t,\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \,\mathrm{d}x + \int_{\mathbb{R}^d} f_t \partial_t \log f_t \,\mathrm{d}x \\ &= \int_{\mathbb{R}^d} \left\langle s_t - \nabla \log \pi_t,\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \,\mathrm{d}x, \end{aligned}$$

where we used the fact that $\int_{\mathbb{R}^d} f_t \partial_t \log f_t \,\mathrm{d}x$ is zero for the last equality. Now,

$$\begin{aligned} &\left| \int_{\mathbb{R}^d} \left\langle s_t - \nabla \log \pi_t,\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \,\mathrm{d}x - \int_{\mathbb{R}^d} \left\langle \nabla \log \frac{f_t}{\pi_t},\, \nabla \log \frac{f_t}{\pi} \right\rangle f_t \,\mathrm{d}x \right|^2 \\ &\leq \int_{\mathbb{R}^d} \left| \left\langle s_t - \nabla \log f_t,\, \nabla \log \frac{f_t}{\pi} \right\rangle \right|^2 f_t \,\mathrm{d}x \\ &\leq \int_{\mathbb{R}^d} |s_t - \nabla \log f_t|^2 \, f_t \,\mathrm{d}x \int_{\mathbb{R}^d} |\nabla \log f_t - \nabla \log \pi|^2 \, f_t \,\mathrm{d}x \\ &= L(s_t, f_t)\, \mathrm{F}(f_t \,\|\, \pi)\,. \end{aligned}$$

Thus, $-\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathbb{R}^d} f_t \log \frac{f_t}{\pi} \,\mathrm{d}x \to \mathbb{E}_{f_t} \left\langle \nabla \log \frac{f_t}{\pi_t},\, \nabla \log \frac{f_t}{\pi} \right\rangle$ as $L \to 0$, at the rate $O(\sqrt{L})$. □

4.4 Numerical experiments

We present several low- and high-dimensional experiments, comparing SBTM to its stochastic counterpart (GF SDE), and SVGD. The examples differ in the log-concavity of the target, the number of modes, the relative mode weighting, the dimension and the annealing path used. We do not extensively compare SBTM to SVGD [120] because without additional tricks such as momentum and cherry-picking the kernel bandwidth we were unable to obtain decent performance from SVGD even in low dimensions, see Tables 4.1 and 4.2. We emphasize that while the numerical experiments presented below would be trivial in the context of DGM, they are quite challenging in the absence of samples from π, only using the score $\nabla \log \pi$.

In the low-dimensional experiments we confirm that, unlike (GF SDE), the (SBTM) dynamics produce smooth trajectories of the relevant metrics: relative entropy, relative

entropy dissipation rate, L2 error. Moreover, we verify that the relative entropy dissipation rate almost exactly matches the approximated relative Fisher information, indicating good score approximation and optimal rate of convergence, empirically confirming the validity of Theorems 4.10 and 4.12 and estimate (4.4), and showcasing the usefulness of having access to the time-varying score $\nabla \log f_t$. Finally, the high-dimensional experiment confirms the scaling of SBTM to high dimensions and demonstrates that even in 784 dimensions the score approximation is accurate and SBTM retains exploration. Additionally, SBTM exhibits somewhat better sample efficiency, likely due to particle dependence, and demonstrates early mode splitting as shown in Section 4.4.5.

We use a three-layer Residual Neural Network of width 128 for 1D experiments, and increase it to five layers for 2D experiments. For the MNIST experiment we use an 8-layer U-Net. For all experiments we use mini-batch gradient descent with the Adamw optimizer. For low-dimensional experiments we use learning rate $5 \cdot 10^{-4}$, batch size 400, and 10 gradient descent steps per simulation step. For the MNIST experiment we use learning rate 10^{-3} and batch size 64. These hyperameters were found empirically. All experiments were done in JAX on a single TITAN X Pascal 12GB GPU. The runtime of the longest experiment is 90 minutes. The code repository is Vilin97/SBTM-sampling.

4.4.1 Log-concave target

Table 4.1: KL divergence (↓) between the sample and the target $\pi = \mathcal{N}(0, 1)$, using time step 0.002 and final time 2.5. SBTM exhibits better sample efficiency, likely due to determinism.

	sample size				
Method	100	300	1000	3000	10000
SBTM (ours)	**0.013**	**0.0032**	**0.0019**	**0.0020**	**0.00099**
SDE	0.020	0.022	0.0094	0.0036	0.0012
SVGD	0.33	0.23	0.16	0.20	0.29

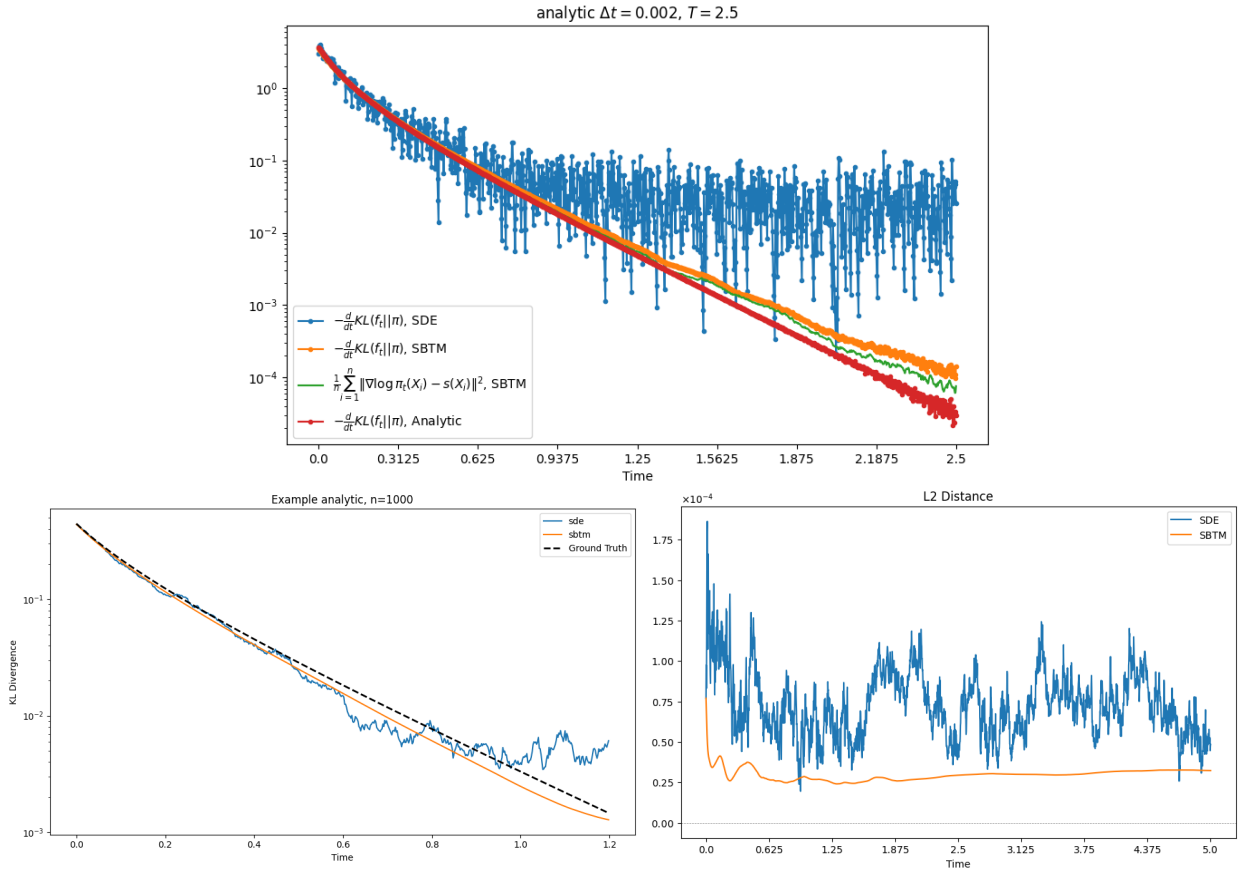


Figure 4.3: Experiment 4.4.1, log-concave target. Top: relative entropy dissipation rate of SBTM (ours) and SDE (stochastic). SBTM approximates entropy decay rate well, while SDE is noisy. Bottom left: relative entropy of SBTM, SDE and the ground truth. SBTM approximates the ground truth well. Bottom right: L2 error to the true ground truth solution. SBTM produces lower error with smoother trajectory.

First, we consider an example that admits an analytic solution $f_t = \mathcal{N}\left(0, 1 - e^{-2(t+0.1)}\right)$, which allows us to compute the true entropy dissipation and the L^2 distance to the true solution. We use sample size $n = 1000$. SBTM exhibits the optimal relative entropy dissipation rate, as evidenced by the close alignment of the orange, green, and red lines in the left panel of Figure 4.3. The deterministic nature of SBTM allows it to achieve better sample

efficiency than the noisy SDE, as shown in Table 4.1. While the marginal density of X_t^i is the same between SBTM and SDE for a fixed i and t, the whole particle ensemble $X_t^1, \ldots, X_t^n$ has a different distribution, as the particles in SBTM interact via the NN. More detailed investigation of the ensemble properties of SBTM is left as future work.

4.4.2 Gaussian mixture

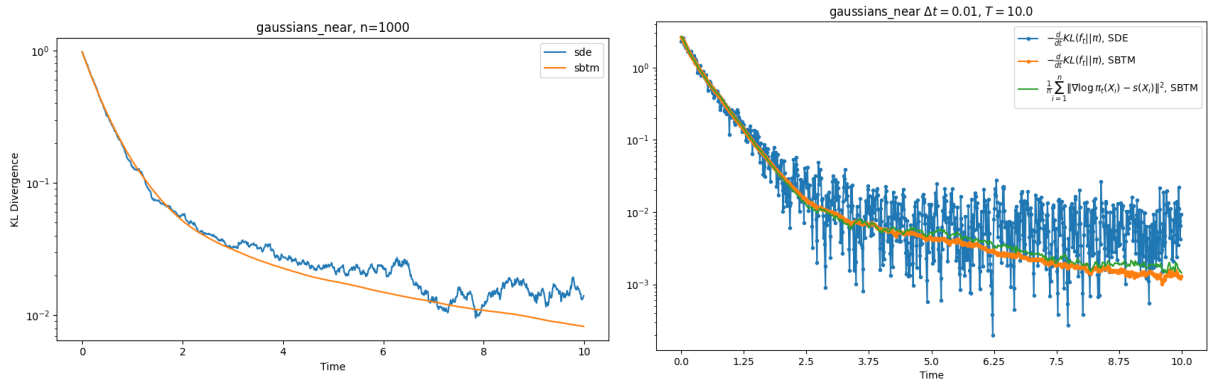


Figure 4.4: Experiment 4.4.2, 1D Gaussian mixture. Left: KL divergence of SBTM (ours) and SDE (stochastic) over time. SBTM exhibits smoother convergence. Right: entropy dissipation of SBTM and SDE. SBTM approximates entropy decay rate perfectly with the computable quantity $\mathrm{F}(f_t \,\|\, \pi)$, while SDE is noisy.

This and further examples do not admit analytic solutions, but we can still compare the quality of the final sample as well as compare the trajectories of the SDE and SBTM. We use $f_0 = \mathcal{N}(0, 1)$ here and in further experiments. Here we sample from the Gaussian mixture $\pi = \frac{1}{4}\mathcal{N}(-2, 1) + \frac{3}{4}\mathcal{N}(2, 1)$ with $n = 1000$. As evidenced by the change of slope at $t = 2.5$ in Figure 4.4, the underlying Markov chain enters metastability, which plagues the convergence to non-log-concave targets. Here the non-log-concavity is mild, so the process still converges in a reasonable time frame. Table 4.2 shows that SBTM achieves competitive KL divergence.

Table 4.2: KL divergence (↓) between the sample and the target $\pi = \frac{1}{4}\mathcal{N}(-2,1) + \frac{3}{4}\mathcal{N}(2,1)$, using time step 0.01, and final time 10. Top: non-annealed. Bottom: 100 particles.

	sample size				
	100	300	1000	3000	10000
SBTM (ours)	**0.022**	0.018	**0.0082**	0.0082	**0.0036**
SDE	0.029	**0.013**	0.014	**0.0068**	0.0043
SVGD	2.8	1.4	2.4	2.1	2.0

	annealing		
	Non-annealed	Geometric	Dilation
SBTM (ours)	**0.022**	**0.058**	**0.037**
SDE	0.029	0.060	0.062
SVGD	2.800	0.470	0.480

4.4.3 Gaussian mixture with geometric annealing

In this example we sample from the well-separated Gaussian mixture $\pi = \frac{1}{4}\mathcal{N}(-4,1) + \frac{3}{4}\mathcal{N}(4,1)$ with $n = 1000$. Because the target is extremely non-log-concave we employ geometric annealing, which linearly interpolates the score between the initial and the target densities: $\nabla \log \pi_t = (1-t)\nabla \log f_0 + t\nabla \log \pi$. Figure 4.5 shows that estimate (4.4) holds empirically almost perfectly, indicating small score-matching loss. The imperfect density reconstruction on the left panel of Figure 4.5 is due to the meta-stability issue, which is prevalent even in annealed dynamics; there is no known tractable dynamics without teleportation of mass that perform well on this example.

4.4.4 Noisy circle

In this example we compare the SDE and SBTM samples from the "noisy circle" density $\pi \propto \exp\left(-\frac{(\|x-(4,0)\|-1)^2}{0.08}\right)$. While this example is non-log-concave, unlike other non-log-concave

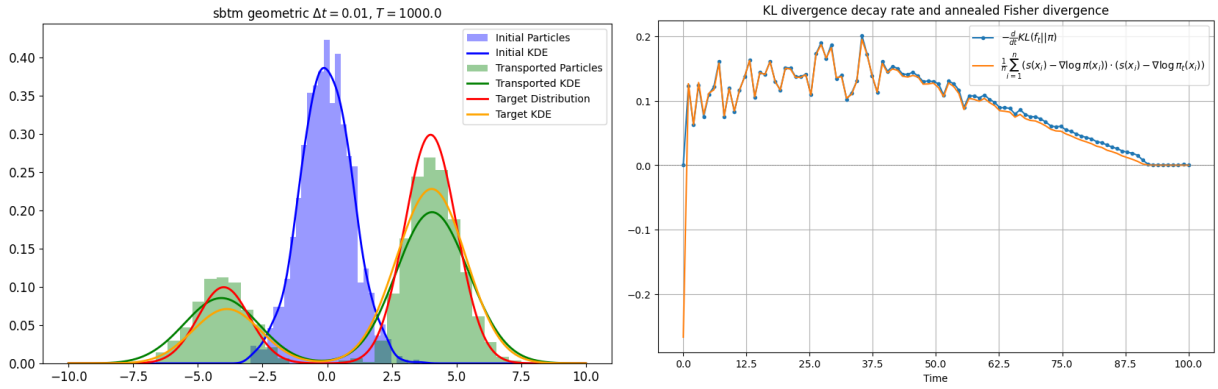


Figure 4.5: Experiment 4.4.3, well-separated 1D Gaussian mixture. Left: reconstructed density of SBTM. It approximates the solution well despite the non-log-concavity. Right: entropy dissipation of SBTM (ours) and SDE (stochastic). SBTM approximates entropy decay rate perfectly even in annealed dynamics.

examples, there is no energy barrier between modes, and vanilla Wasserstein Gradient Flow converges quickly. We use 10,000 particles and time step 0.01. While the SDE samples fill the vacuum region in Figure 4.6 around the center due to Brownian noise, the SBTM samples leave it blank due to determinism.

4.4.5 Gaussian mixture with dilation annealing

To sample from a Gaussian mixture with 16 well-spaced modes we employ the annealing schedule $\pi_t(x) = \pi\left(\frac{T}{t}x\right)$ from [29]. This is a challenging example due to the extreme non-log-concavity of the target. Figure 4.7 shows the densities at different time points demonstrating early mode splitting due to deterministic dynamics. Figure 4.8 demonstrates that the score is learned well by showing that estimate (4.4) holds almost perfectly. With the large sample size of 20,000 particles and 10,000 time steps this experiment took only 90 minutes on a single TITAN X GPU, leveraging the linear scaling of memory and time complexity of SBTM. For comparison, we were unable to use SVGD with 20,000 particles due to its $O(n^2)$ memory scaling.

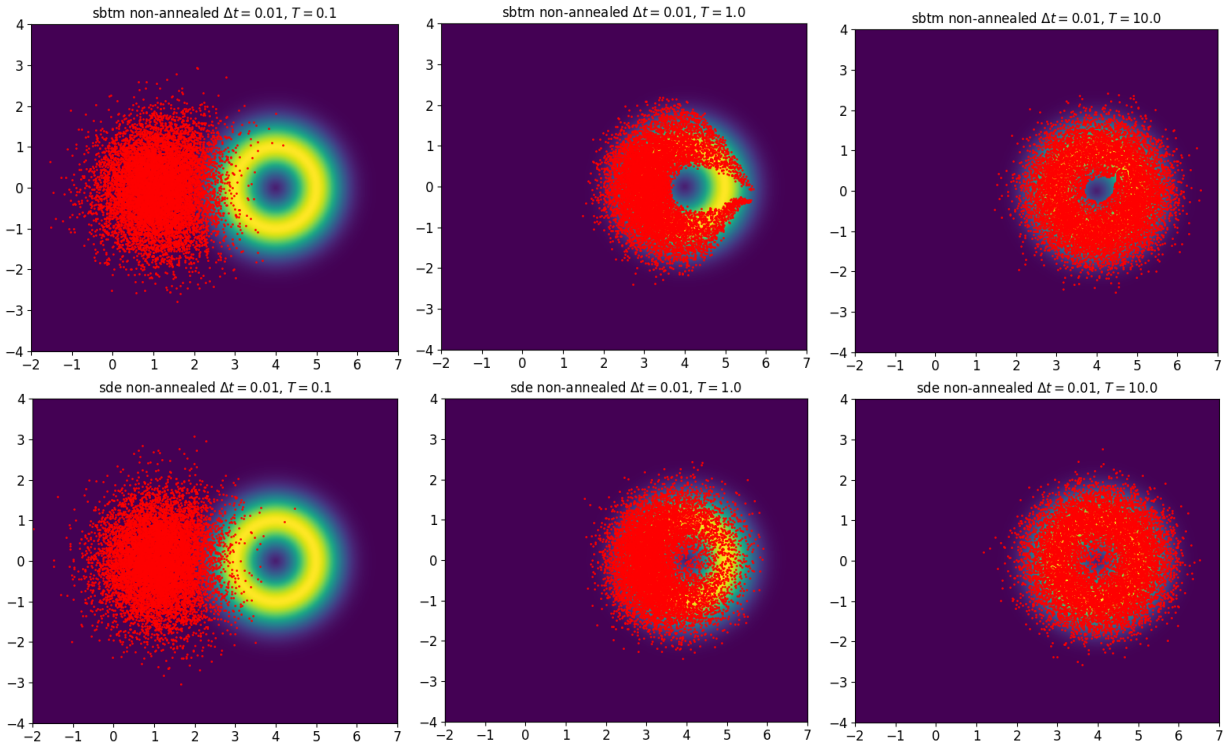


Figure 4.6: Experiment 4.4.4, noisy circle. SBTM (ours, top) leaves the vacuum region empty, while SDE (bottom) fills it, demonstrating the effect of determinism.

4.4.6 High-dimension experiments

To evaluate the scaling of our method to high-dimensional settings, we apply SBTM to produce MNIST digits [107], which corresponds to sampling a 784-dimensional distribution. We purposefully sample directly in pixel space, without using a VAE encoder, to maintain the high dimensionality of the data. To obtain $\nabla \log \pi$ we train a U-Net with score-matching. Figure 4.9 shows that SBTM produces high quality results, comparable to the SDE sample. Figure 4.10 shows that that SBTM maintains exploration in high dimensional spaces. The amount of training controls the strength of interaction between particles. With sufficient training, particles repel from each other, providing deterministic exploration. To confirm that $\nabla \log f_t$ is being meaningfully learned, we plot the cosine similarity to $\nabla \log \pi$ across

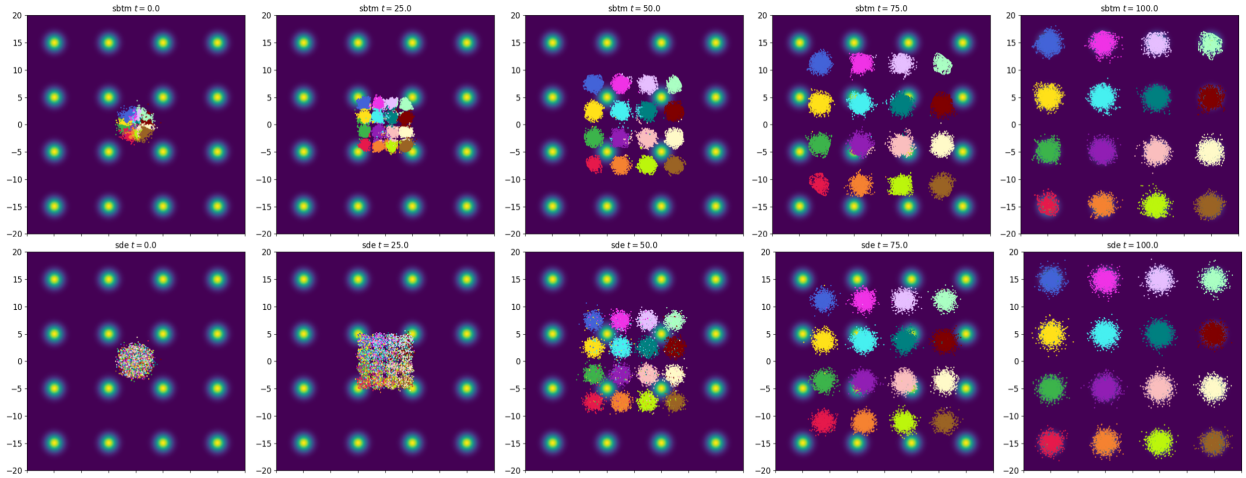


Figure 4.7: Experiment 4.4.5, well-separated 2D Gaussian mixture. Scatter plot over time. Top: SBTM (ours) with the dilation annealing. Bottom: SDE with the dilation annealing [29]. SBTM separates into modes early on, compared to the SDE.

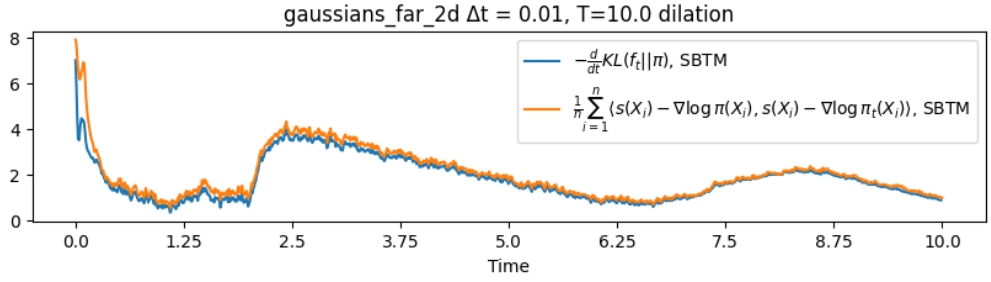


Figure 4.8: Experiment 4.4.5, well-separated 2D Gaussian mixture. The estimate in (4.4) holds empirically, indicating small score-matching loss and good score approximation in annealed dynamics.

time in Figure 4.11. More training epochs per step leads to faster learning of the target score $\nabla \log \pi$.

Figure 4.9: Experiment 4.4.6, high-dimensional. Sampled MNIST digits using $f_0 = \mathcal{N}(0, 1)$, sample size 64, time step 10^{-3}, 30 steps, variable amount of intermediate training. Digits generated by SBTM are competitive in visual quality.

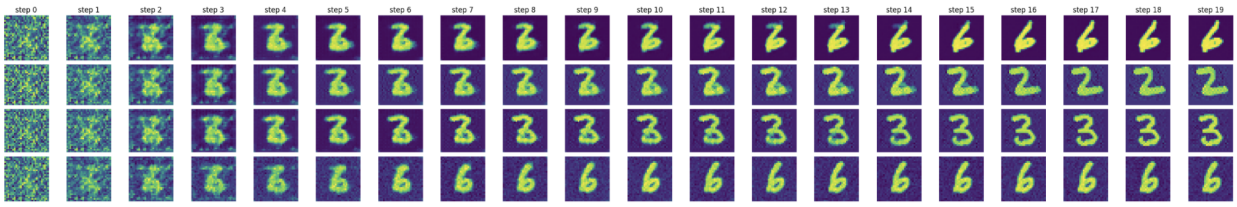


Figure 4.10: Experiment 4.4.6, high-dimensional. Starting from the same initial point, SBTM produces distinct sample trajectories depending on the training schedule. The amount of training controls the strength of interaction between particles. Top to bottom: SBTM without training (equivalent to gradient ascent on $\nabla \log \pi$), SBTM with small amount training, SBTM with large amount training, and finally the SDE (Langevin).

4.5 Conclusion

In this work we use tools and intuition from diffusion generative modeling to tackle the harder problem of sampling π given only access to $\nabla \log \pi$ but not samples from π. Our method allows for deterministic sampling with smooth trajectories and provably optimal

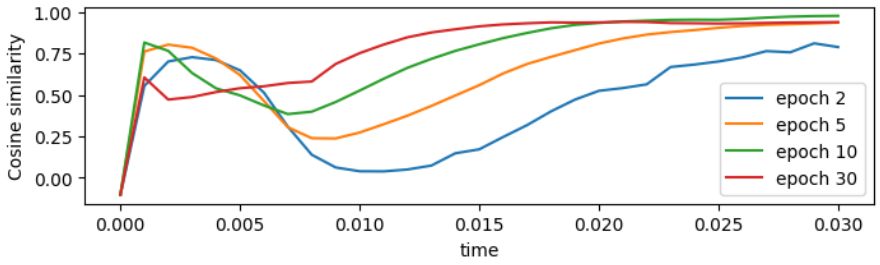


Figure 4.11: Experiment 4.4.6, high-dimensional. Cosine similarity between the true score $\nabla \log \pi$ and learned score $\nabla \log f_t$ over simulation time. Even with very little training the model learns the score well. Different lines use different numbers of epochs per time-step, effectively changing η in (4.10).

rate of entropy dissipation. Additionally, access to the learned score $\nabla \log f_t$ allows for the computation of relative Fisher info to estimate convergence, making the dynamics more interpretable. Our method integrates well with annealed dynamics to sample from challenging non log-concave densities. Finally, our method scales well both in the sample size, with $O(n)$ complexity, and in dimension, as demonstrated in the 784-dimensional example.

Chapter 5

DISCOFORMER: PLUG-IN DENSITY AND SCORE ESTIMATION WITH TRANSFORMERS

This chapter, the second of Part II, returns to the theme that runs through Chapters 2 and 4: the role of the score function $\nabla \log f$ as the link between an evolving particle ensemble and the underlying density. In Chapter 2 the score drives the deterministic Landau particle ODE; in Chapter 4 it drives the deterministic sampling ODE. In both cases, the score is unknown and is learned online – a separate neural network is trained from scratch on every new problem. *This chapter makes score estimation itself the object of study* and proposes a single pretrained operator that replaces the per-problem online training stage with one forward pass.

The starting point is that classical kernel density estimation (KDE) and modern score matching represent two complementary failure modes. KDE is distribution-agnostic and provably consistent, but suffers severely from the curse of dimensionality. Score matching scales well to high dimensions but is distribution-specific: a fresh model has to be trained for every new target. DiScoFormer (Density and Score Transformer) is a permutation- and affine-equivariant Transformer that maps an i.i.d. sample $X \in \mathbb{R}^{n\times d}$ to both the density values and the scores at the sample points – and, via cross-attention, at arbitrary query points. It is trained once on synthetic Gaussian-mixture data and applied off-the-shelf across distributions, sample sizes, and dimensions. The link to classical nonparametrics is made precise in Theorem 5.2, which shows that a single self-attention head with the right parameterization *exactly* reproduces the normalized Gaussian KDE weights.

In particular, DiScoFormer plugs directly into the SBTM-based plasma solver of Chapter 2 as a drop-in replacement for the online-trained score network: the per-problem training

is removed, and the resulting solver matches the analytic covariance of the homogeneous Landau equation (Figure 5.11 below). It also serves as a plug-in score oracle for score-debiased KDE [60], for relative Fisher information between samples drawn from two different distributions, and for differential entropy estimation.

This chapter follows [92] almost verbatim.

5.1 Introduction

Estimating a probability density f and its corresponding score function $\nabla \log f$ from i.i.d. samples is a foundational problem in statistical inference, underpinning tasks in generative modeling, stochastic control, and information theory. While classical non-parametric methods such as Kernel Density Estimation (KDE) offer strong theoretical guarantees and respect fundamental symmetries, they are hampered by a rigid bias–variance trade-off and a prohibitive "curse of dimensionality" as the dimension d increases. Conversely, modern neural score-matching models achieve high accuracy in high-dimensional settings but are typically transductive: they must be retrained for every new target distribution, precluding their use as general-purpose, "off-the-shelf" estimators.

We propose bridging this gap by framing density and score estimation as a sequence-to-operator [170] learning task. Given an i.i.d. sample $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, we seek to learn a single model that maps the entire sample X to its underlying density and score functions. Specifically, we define an operator T for density and S for scores:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{T} \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}, \qquad \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \xrightarrow{S} \begin{pmatrix} \nabla \log f(x_1) \\ \vdots \\ \nabla \log f(x_n) \end{pmatrix}. \tag{5.1}$$

A critical requirement for such an operator is that it must respect the inherent symmetries of the data. Specifically, the estimates should be permutation-equivariant with respect to the sample index and affine-equivariant with respect to the coordinate space. For a permutation

matrix P, an invertible matrix A, and a shift μ, these symmetries are defined as

$$T(PXA + \mathbf{1}\mu^\top) = P\,|\det A|^{-1}\,T(X),$$
$$S(PXA + \mathbf{1}\mu^\top) = P\,A^{-1}\,S(X).$$

Building on these principles, we introduce DiScoFormer – Density and Score Transformer. By treating the sample X as a sequence, the Transformer architecture provides permutation equivariance by construction. We achieve affine equivariance through a specialized whitening mechanism that normalizes inputs up to orthogonal transformations, combined with data augmentation to ensure rotation invariance. Unlike traditional models, our architecture utilizes cross-attention, allowing for the evaluation of f and $\nabla \log f$ at arbitrary query points without retraining.

Furthermore, we provide a theoretical bridge between modern attention mechanisms and classical non-parametrics. We demonstrate analytically that under specific parameterizations, self-attention weights recover normalized KDE weights. This expressivity result positions the Transformer not as a black box, but as a principled, data-adaptive generalization of kernel methods that can dynamically learn multiscale smoothing behaviors from data.

Empirically, the model outperforms KDE and score-debiased KDE [60] across a range of dimensions and sample sizes on both in-distribution and out-of-distribution test cases. Trained on Gaussian Mixture Model (GMM) data, the model generalizes to GMMs with more modes and to non-Gaussian targets such as the Laplace distribution, and it demonstrates favorable scaling in both n and d. By amortizing the estimation cost, our model provides high-fidelity plug-in oracles for downstream tasks, including entropy estimation, Fisher information calculation, and solving Fokker–Planck-type partial differential equations (PDEs).

In summary, our contributions are:

- DiScoFormer, a universal Transformer model for one-shot estimation of density and score functions from i.i.d. samples.

- A proof and empirical evidence establishing self-attention as a data-adaptive generalization of normalized kernel density estimation.
- Non-parametric density and score estimators that surpass classical KDE methods in accuracy across sample sizes and dimensions.
- Applications to Fisher information, differential entropy, and Fokker–Planck-type PDEs.

5.2 Related work

This work lies at the intersection of nonparametric density estimation, score-based methods, permutation-invariant neural architectures, and operator learning for probability measures.

Kernel density estimation. Classical nonparametric methods such as Parzen windows [140] and Silverman's rule-of-thumb KDE [156, 187] estimate densities via local kernel averaging. While theoretically grounded, these methods suffer from a rigid bias–variance trade-off and scale poorly with dimension [154]. Adaptive bandwidth methods [1] and fast algorithms such as the Fast Gauss Transform [73] partially alleviate computational costs but do not resolve the fundamental curse of dimensionality. Score-Debiased KDE [60] demonstrates that access to a score oracle enables bias correction, motivating our approach: we provide precisely this missing oracle as a reusable, distribution-agnostic component.

Other nonparametric density estimators. k-nearest-neighbor methods [122], orthogonal series estimators [58, 152], and normalizing flows [150, 54, 139] offer alternatives to KDE but either lack direct score estimates or require per-distribution retraining. Our model provides both density and score from a single pretrained network.

Score matching and score-based generative models. Score matching [86] and its denoising variant [186] estimate unnormalized model scores without computing partition functions. Sliced score matching [162] scales these ideas to higher dimensions. Score-based

generative models [161, 78, 165, 159] learn scores along diffusion processes for sampling, with fast solvers [125, 100] and convergence guarantees [40, 38]. These methods train on a single target density; in contrast, our model learns a universal score operator that generalizes across distributions without retraining.

Permutation-invariant architectures. DeepSets [208] and Set Transformers [108] provide foundational architectures for exchangeable data, while equivariant graph networks [134] extend these ideas to richer symmetry groups. Neural Processes [69, 57] and Perceiver IO [95] learn distributions over functions from context sets. Recent work shows that permutation-equivariant transformers can universally approximate mean-field dynamics [18]. Our architecture exploits the same symmetry but outputs statistical quantities – density and score – while additionally enforcing affine equivariance.

Attention as kernel smoothing. Formal connections between softmax attention and kernel methods have been developed through linear-attention reformulations [101], random-feature approximations [44], Nyström methods [199], explicit Gaussian-kernel replacements [42, 126], and Nadaraya–Watson regression interpretations [210]. Our Theorem 5.2 builds on this lineage by showing that self-attention weights can exactly reproduce normalized Gaussian KDE weights.

Neural operators. Neural Operators [104], DeepONet [130], and Fourier Neural Operators [111] learn mappings between infinite-dimensional function spaces. The Score Neural Operator [112] extends this framework to probability distributions via RKHS embeddings. In contrast, our transformer acts directly on raw i.i.d. samples without kernel embeddings, yielding a simpler operator that generalizes across densities and dimensions.

Particle methods and Fokker–Planck solvers. Interacting particle methods for Fokker–Planck equations [26, 133] and score-based transport modeling [20, 128, 91] require accurate score estimates at each timestep. Stein Variational Gradient Descent [121, 119, 56] similarly

relies on kernel-based score approximations. Our model provides a fast, pretrained oracle that plugs directly into these solvers, avoiding per-distribution retraining.

5.3 Methodology

In this section we describe the core ideas of the proposed method: the symmetry-aware architecture, the relationship to classical kernel density estimation, the training pipeline, and implementation.

5.3.1 Equivariance

Table 5.1: Relative MSE of affine equivariance error, averaged over 50 trials.

Transform	Rel. MSE
Permutation	0
Translation	0
Isotropic scaling	0
Anisotropic scaling	0
Rotation	5×10^{-4}
Full affine	1×10^{-4}

First, Theorem 5.1 establishes the equivariance properties of density and score estimation. The proof is folded inline below.

Proposition 5.1 (Permutation and affine equivariance of density and score evaluation)**.** *Let $f : \mathbb{R}^d \to (0, \infty)$ be a differentiable density, and let $X = (x_1, \ldots, x_n)^\top \in \mathbb{R}^{n \times d}$ be its i.i.d. sample. Define*

$$T(X) := \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}, \qquad S(X) := \begin{pmatrix} \nabla \log f(x_1) \\ \vdots \\ \nabla \log f(x_n) \end{pmatrix}.$$

Let $P \in \mathbb{R}^{n\times n}$ be a permutation matrix, $A \in \mathbb{R}^{d\times d}$ be invertible, $\mu \in \mathbb{R}^d$, and $\mathbf{1} \in \mathbb{R}^n$ be the vector of ones. Then

$$T\big(PXA + \mathbf{1}\mu^\top\big) = P\,|\det A|^{-1}\,T(X), \tag{5.2}$$

$$S\big(PXA + \mathbf{1}\mu^\top\big) = P\,A^{-1}\,S(X). \tag{5.3}$$

Proof. Write $X = (x_1^\top, \dots, x_n^\top)^\top$ and let σ be the permutation of $\{1, \dots, n\}$ corresponding to P, i.e. $(PX)_{i\cdot} = x_{\sigma(i)}^\top$. Define

$$Y := PXA + \mathbf{1}\mu^\top \in \mathbb{R}^{n\times d}.$$

Then the i-th row of Y is

$$y_i^\top = x_{\sigma(i)}^\top A + \mu^\top, \qquad \text{equivalently } y_i = A^\top x_{\sigma(i)} + \mu.$$

Consider the density obtained from f by the affine change of variables $y = A^\top x + \mu$:

$$f^{(\mu,A)}(y) := |\det A|^{-1} f\big((A^\top)^{-1}(y-\mu)\big).$$

By construction,

$$f^{(\mu,A)}(y_i) = |\det A|^{-1} f\big(x_{\sigma(i)}\big).$$

Hence, by the definition of T,

$$T(Y) = \begin{pmatrix} f^{(\mu,A)}(y_1) \\ \vdots \\ f^{(\mu,A)}(y_n) \end{pmatrix} = \begin{pmatrix} |\det A|^{-1} f(x_{\sigma(1)}) \\ \vdots \\ |\det A|^{-1} f(x_{\sigma(n)}) \end{pmatrix} = P\,|\det A|^{-1}\,T(X),$$

which proves (5.2).

For the score, differentiate $\log f^{(\mu,A)}$:

$$\log f^{(\mu,A)}(y) = -\log|\det A| + \log f\big((A^\top)^{-1}(y-\mu)\big).$$

By the chain rule,

$$\nabla_y \log f^{(\mu,A)}(y) = \big((A^\top)^{-1}\big)^\top \nabla_x \log f(x)\big|_{x=(A^\top)^{-1}(y-\mu)} = A^{-1}\nabla \log f\big((A^\top)^{-1}(y-\mu)\big).$$

Evaluating at $y_i = A^\top x_{\sigma(i)} + \mu$ gives

$$\nabla_y \log f^{(\mu,A)}(y_i) = A^{-1} \nabla \log f(x_{\sigma(i)}).$$

Thus,

$$S(Y) = \begin{pmatrix} \nabla_y \log f^{(\mu,A)}(y_1) \\ \vdots \\ \nabla_y \log f^{(\mu,A)}(y_n) \end{pmatrix} = \begin{pmatrix} A^{-1} \nabla \log f(x_{\sigma(1)}) \\ \vdots \\ A^{-1} \nabla \log f(x_{\sigma(n)}) \end{pmatrix} = P\, A^{-1}\, S(X),$$

which proves (5.3). □

To capture these symmetries, we use the Transformer architecture without positional encodings, combined with an affine normalization layer (Figure 5.2). The whitening step yields equivariance up to rigid rotations $O(d)$; training on randomly oriented GMMs encourages the network to learn approximate rotation invariance, closing the remaining gap in practice. Table 5.1 confirms this empirically.

5.3.2 Attention and KDE

Theorem 5.2 demonstrates that a single attention head can compute the classical KDE, showing that the attention mechanism has the capacity to implement kernel-based smoothing. While the proposition assumes L^2-normalized inputs, in practice the learned Q and K projections adapt to the actual input statistics after whitening.

Proposition 5.2 (Attention can compute KDE)**.** *Let $X \in \mathbb{R}^{n \times d}$ be the normalized data matrix with $\|x_i\| = 1$ for all i. Consider a single self-attention head*

$$\text{Attention}(Q, K, V) = \text{Softmax}(QK^\top)\, V$$

with

$$Q = K = \tfrac{1}{h} X, \qquad V = I_n,$$

for some bandwidth parameter $h > 0$. Then

$$[\text{Attention}(Q, K, V)]_{ij} = \frac{\exp\!\left(-\frac{\|x_i - x_j\|^2}{2h^2}\right)}{\sum_{k=1}^{n} \exp\!\left(-\frac{\|x_i - x_k\|^2}{2h^2}\right)},$$

i.e. the attention matrix coincides with the normalized Gaussian kernel of bandwidth h.

Proof. With $Q = K = X/h$, we have $QK^\top = XX^\top/h^2$, so

$$\text{Softmax}(QK^\top)_{ij} = \frac{\exp(x_i^\top x_j/h^2)}{\sum_k \exp(x_i^\top x_k/h^2)}.$$

Because all $\|x_i\| = 1$,

$$x_i^\top x_j = 1 - \tfrac{1}{2}\|x_i - x_j\|^2.$$

Substituting and simplifying gives

$$\exp(x_i^\top x_j/h^2) = e^{1/h^2}\exp(-\|x_i - x_j\|^2/2h^2),$$

and the common factor e^{1/h^2} cancels in the row-wise softmax normalization. Hence

$$\text{Softmax}(QK^\top)_{ij} = \frac{\exp(-\|x_i - x_j\|^2/2h^2)}{\sum_k \exp(-\|x_i - x_k\|^2/2h^2)}.$$

Since $V = I_n$, multiplying by V leaves this matrix unchanged, and the claim follows. □

While the above proposition establishes that the attention mechanism has the capacity to approximate KDE well, we must verify whether the model learns to do it in practice. To empirically verify the relationship between the KDE weights and attention scores, we visualize attention in Figure 5.1. As expected, the Transformer learns to attend to nearby points, with one attention head attending to far away samples instead. The attention mechanism effectively learns a kernel-like behavior, with emergent head specialization. The learned features allow for much better scaling both in the sample size and dimension, compared to KDE. In Section 5.4.8 we further visualize the attention scores of individual heads and find an emergent behavior of head specialization: specifically, head 1 specialized to look at far-away points, whereas heads 0, 2, and 5 specialized to look at close- and mid-range interactions. Finally, heads 3, 4, 6, and 7 specialized to look in specific directions. This emergent behavior further links the multi-head Transformer to kernel-based methods, but with flexible multi-scale kernels.

5.3.3 Training data

We train the Transformer to approximate the density and score of Gaussian Mixture Models (GMMs) from i.i.d. samples. GMMs are chosen because they form a dense family in the space of continuously differentiable densities, and both their densities and scores admit closed-form expressions. Training samples are generated on the fly (Algorithm 4). The objective combines the mean squared errors (MSE) of the predicted density and score through a convex weighting:

$$\mathcal{L}_T = \frac{1}{n}\|T(X,Y) - f_X(Y)\|_2^2,$$
$$\mathcal{L}_S = \frac{1}{n}\|S(X,Y) - \nabla \log f_X(Y)\|_2^2,$$
$$\mathcal{L} = \alpha\mathcal{L}_T + (1-\alpha)\mathcal{L}_S.$$

Algorithm 4 GMM dataloader.

Require: Batch size B, dimension d, sample sizes n_x, n_y, component range $[k_{\min}, k_{\max}]$.

repeat

Sample $k \in \{k_{\min}, \ldots, k_{\max}\}$.

for $b = 1$ **to** B **do**

Sample two random GMMs with k components each.

Sample X_b from the first, Y_b from the second.

Compute $f_{X_b}(y)$ and $\nabla \log f_{X_b}(y)$ for $y \in Y_b$.

end for

Output $\big(X,\, Y,\, f_X(Y),\, \nabla \log f_X(Y)\big)$.

until training stops

5.3.4 Model variants and implementation

We explore several architectural variants aimed at extending density and score estimation beyond the observed samples X, enabling evaluation at arbitrary query points Y. To achieve

this, we introduce a cross-attention mechanism that processes two complementary inputs: a context tensor encoding the observed samples and a query tensor specifying the target locations. Through this interaction, the model learns to relate queries to the statistical structure of the observed data, producing accurate estimates of both $f_X(y)$ and $\nabla \log f_X(y)$ for any $y \in Y$. The cross-attention layer thus serves as a learned nonparametric smoother, effectively generalizing density and score estimation across the input domain rather than being limited to the training samples.

Recognizing that density and score estimation are mathematically coupled – linked through the gradient of $\log f$ – we employ a unified architecture with a shared backbone and two specialized output heads. This joint formulation encourages mutual consistency between the two quantities and allows the model to leverage shared geometric and statistical features during training, improving both accuracy and sample efficiency.

5.4 Experiments

In this section we empirically investigate:

1. comparison with KDE and SD-KDE [60];
2. scaling in sample size n and dimension d;
3. out-of-distribution generalization;
4. inference-time scaling in n and d.

We also show that the learned model can serve as a score oracle in SD-KDE, as well as in deterministic SBTM-like solvers for Fokker–Planck-type equations such as the Landau equation.

While there are many techniques for learning the score $\nabla \log f$ such as ISSM [159, 165] and DDPM [125], they learn the score of a specific density undergoing a forward diffusion-like stochastic process. Since our model is density-agnostic and can be used out-of-the-box

on a sample of any density, the most comparable method is KDE and its variants. See Section 5.4.6 below for a comparison between the score matching loss and our model.

All experiments are performed on a single 48GB L40S GPU. The architecture is a permutation- and affine-equivariant Transformer. It consists of a linear embedding, four Transformer encoder layers (128-dimensional hidden size, 8 heads, GELU activation, pre-normalization), and a linear projection to the output dimension. No positional encodings are used, preserving permutation-equivariance. The model has about 800,000 parameters. We use batch size 32, sample size $n = 2048$, dropout 0.1, and draw GMMs with 1–10 modes with means in $[-3, 3]^d$ and diagonal covariances in $[0, 1]^d$. A runtime comparison with KDE is provided in Section 5.4.7.

5.4.1 Score estimation

To compare our model to KDE, we first sample a 2D Gaussian, visualize the estimated scores using a quiver plot, and compute the MSE score loss. Figure 5.3 demonstrates that even in dimension 2 and on a KDE-favorable example of a unimodal Gaussian, our model performs better. The KDE score approximation is computed using

$$k(x_i, x_j) = \exp\left(-\frac{1}{2}\sum_{\ell=1}^{d}\frac{(x_i^\ell - x_j^\ell)^2}{h_\ell^2}\right), \tag{5.4}$$

$$\widehat{s}(x_i) = \frac{1}{h^2} \odot \left(\frac{\sum_j k(x_i, x_j)x_j}{\sum_j k(x_i, x_j)} - x_i\right), \tag{5.5}$$

where $h_\ell = \sigma_\ell n^{-1/(d+4)}$ is Silverman's rule [156] applied per coordinate with σ_ℓ the sample standard deviation along dimension ℓ, and $\odot$ denotes element-wise multiplication.

For a quantitative comparison of score estimation against the KDE, we compute the Mean Squared Error (MSE) for dimensions 1 and 10 across several magnitudes of sample sizes n. See the left panel of Figure 5.5. The transformer model achieves much better accuracy than KDE even in dimension 1 and especially in dimension 10.

To judge the out-of-distribution generalization of our model, we compare the MSE loss on GMMs with 1–19 modes, while the training data consists of GMMs with 1–10 modes. The

right panel of Figure 5.5 shows that the MSE loss is monotone and stable in the number of modes. This indicates good generalization. Additionally, we evaluate on two non-Gaussian distributions: the Laplace distribution, which has a sharper peak and heavier tails than a Gaussian, and the Student-t distribution ($\nu = 3$), which has polynomial rather than exponential tail decay. Despite being trained only on GMMs, the model estimates the score successfully on both, as shown in Tables 5.2 and 5.3. Additionally, we can further improve the out-of-distribution performance using test-time training (TTT) [169, 66], using the consistency loss

$$\mathcal{L}_{\text{con}} = \frac{1}{n} \sum_{i=1}^{n} \| S(X)_i - \nabla_{x_i} \log T(X)_i \|_2^2 , \tag{5.6}$$

where $T(X)_i$ is the predicted density and $\nabla \log T(X)$ is computed via automatic differentiation. Figure 5.4 and Table 5.3 show that as few as 4 TTT steps are sufficient to improve performance on both distributions.

Table 5.2: MSE of score estimation on the 2D Laplace distribution.

n	KDE	Transformer
512	0.3810	**0.3598**
1024	0.3305	**0.2992**
2048	0.2990	**0.2756**
4096	0.2650	**0.2597**

5.4.2 Relative Fisher information

The Relative Fisher Information (RFI) measures how two densities differ in their local geometry, comparing their score fields rather than their probability mass. Unlike the KL divergence, which captures global differences, RFI focuses on the mismatch between gradients of the log-densities, offering a more sensitive criterion for evaluating score-based models and generative estimators.

Table 5.3: MSE of score estimation on the 2D Student-t distribution ($\nu = 3$).

n	KDE	No TTT	TTT 4	TTT 6	TTT 8
128	0.1515	0.1980	0.1676	0.1569	**0.1450**
256	0.1206	0.1119	0.0956	0.0908	**0.0897**
512	0.0916	0.0574	0.0488	**0.0485**	0.0514
1024	0.0812	0.1023	0.0806	0.0768	**0.0765**

Formally, for two densities f and g on $\mathbb{R}^d$, the relative Fisher information of g with respect to f is defined as

$$\mathrm{I}(g\|f) = \mathbb{E}_{x\sim f}\left[\, \|\nabla_x \log g(x) - \nabla_x \log f(x)\|_2^2 \,\right],$$

and can be approximated by the Monte Carlo estimate on samples $x_i \sim f$ with

$$\mathrm{I}(g\|f) \approx \frac{1}{n}\sum_{i=1}^{n} \|\nabla \log g(x_i) - \nabla \log f(x_i)\|_2^2.$$

Because our model outputs the scores $\nabla \log g(x_i)$ using samples y_i from g, we can compute relative Fisher information between two distributions using their samples, as visualized in Figure 5.6.

5.4.3 Density estimation via SD-KDE

Knowledge of the score $\nabla \log f$ can be exploited by score-debiased KDE (SD-KDE) [60] to reduce the bias from $O(h^2)$ to $O(h^4)$. SD-KDE sharpens samples by moving them along the score before applying KDE:

$$X \mapsto X + \frac{h^2}{2}\nabla \log f(X),$$

which counteracts KDE's smoothing bias. In practice the true score is unavailable. We therefore consider five methods:

1. **KDE (Silverman)** [156].

2. **Emp-SD-KDE**: SD-KDE with the score derived from Silverman KDE, following [60].

3. **SD-KDE (Learned Score)**: SD-KDE with the Transformer-predicted score.

4. **SD-KDE (Autograd Score)**: SD-KDE with the score from autograd of the Transformer's learned log-density.

5. **Transformer Density**: direct density/log-density from our model.

Figure 5.7 shows that incorporating a learned score substantially improves density recovery over plain KDE; Emp-SD-KDE reduces bias, while SD-KDE with our predicted score yields the most accurate estimate. In 1D bimodal GMMs (Figure 5.8), SD-KDE with the learned score best matches the truth by MSE. Our Transformer-based approaches exhibit superior accuracy and scaling. The direct density approach performs best at small n but, as expected, scaling with n favors score-based methods; learning the score is an easier objective than learning the density itself due to the difficulty of learning the normalization constant $Z = \int_{\mathbb{R}^d} f$.

5.4.4 Estimation of entropy and Fisher information

Differential entropy $H(f) = -\mathbb{E}_f \log f$ and its gradient, the Fisher information $I(f) = \mathbb{E}_f \|\nabla \log f\|_2^2$, are fundamental statistical quantities that characterize the spread of the distribution. We use the discretizations

$$H(f) \approx \frac{1}{n}\sum_{i=1}^{n} \log f(x_i) \approx \frac{1}{n}\sum_{i=1}^{n} \log T(X)_i,$$
$$I(f) \approx \frac{1}{n}\sum_{i=1}^{n} \|\nabla \log f(x_i)\|^2 \approx \frac{1}{n}\sum_{i=1}^{n} \|S(X)_i\|^2,$$

to estimate $H(f)$ and $I(f)$ using only an i.i.d. sample of f. In dimension 1 we sample a GMM with three modes, numerically compute its entropy and Fisher information using expensive grid integration and the oracle knowledge of its density and score, and compute

the estimation error. In dimension 10 we cannot afford to do grid estimation, so we sample a random Gaussian with randomized mean and covariance, and use the known analytical formulas. We use the debiased KDE method of [60] here as well. Figure 5.10 demonstrates that already in dimension 1 both the score model S and the density model T outperform Silverman KDE. In dimension 10 the learned density performs much better for entropy estimation and the learned score performs much better for Fisher information estimation.

5.4.5 Plasma simulation

In physics, simulating plasma – hot ionized gas – is used in nuclear fusion and rocket propulsion. It is often done by solving the Landau equation, a Fokker–Planck-type kinetic PDE. In 2020, [25] recognized that the Landau equation can be rewritten in the form of a transport equation with velocity

$$v(x) = -\int_{\mathbb{R}^3} A(x-y)\big(\nabla \log f(x) - \nabla \log f(y)\big) f(y)\ \mathrm{d}y$$

and solved numerically with a particle method by approximating the score via a particular KDE variant, called the blob method. On the other hand, in machine learning [20] proposed a neural network-based method of solving any Fokker–Planck equation called Score-Based Transport Modeling, while [91, 93] adapted it to the Landau equation and to the problem of sampling. The core idea is to continually train the NN s_θ on the score matching loss $\mathbb{E}_f[\|s_\theta(x)\|^2 + 2\nabla_x \cdot s_\theta(x)]$, which is equivalent to training on the MSE loss $\mathbb{E}_f[\|s_\theta - \nabla \log f\|^2]$. While [20, 91, 87] proved theoretically and demonstrated empirically that this online training effectively bounds the error, using SBTM can be slow due to the on-the-fly NN training. We reproduce experiments 5.3 and 5.4 from [91] using the learned score. Unlike SBTM, we only use the pre-trained model and do not perform any training throughout the simulation. Figure 5.11 demonstrates that the Transformer model is able to match the analytic covariance well, while the KDE-based solver struggles.

5.4.6 Comparison with the score matching loss

The Hyvärinen score matching loss [86] estimates the data score without access to the true density by minimizing

$$\mathcal{L}_{\mathrm{SM}}(\theta) = \mathbb{E}_{x\sim f(x)}\big[\, \| \, s_\theta(x) \, \|^2 + 2\, \nabla_x \cdot \, s_\theta(x)\big]\,,$$

which corresponds to the Fisher divergence $\mathbb{E}_f[\|s_\theta(x) - \nabla_x \log f(x)\|^2]$ up to a constant. To avoid explicitly computing the divergence term, one can apply a finite-difference denoising trick, replacing $\nabla \cdot \, s_\theta(x)$ with a stochastic estimator

$$\nabla \cdot \, s_\theta(x) \;\approx\; \mathbb{E}_{z\sim\mathcal{N}(0,I)}\left[\frac{z^\top\big(s_\theta(x+\alpha z) - s_\theta(x-\alpha z)\big)}{2\alpha}\right],$$

yielding a practical divergence-free approximation used in many implementations of score matching.

In Figure 5.12 we compare the performance of the score matching loss with that of the proposed Transformer model. The Transformer model is superior in two respects. First, it does not need to be retrained on new samples. Second, the score matching loss is easy to under- or over-fit, as demonstrated in the figure by training for 10, 100, and 1000 steps.

5.4.7 Runtime comparison

While KDE is known to fail in moderate and high dimensions, it is thought to be very fast. We compare the wall-clock runtime of KDE score estimation as in (5.5) and the Transformer model we propose. The Transformer model contains 800,000 learned parameters. We use both KDE and the Transformer model on a single L40S GPU with 48GB of memory. Both methods are $O(n^2)$ asymptotically due to pairwise computations (kernel evaluations for KDE, attention for Transformers). However, in practice the observed scaling differs: Figure 5.13 shows that while KDE is faster for small sample sizes $n \leq 2048$, it becomes slower after this point, especially in higher dimension. Additionally, the naive KDE implementation runs out of GPU memory after $n = 32{,}768$, while the Transformer benefits from the highly optimized

attention kernels in modern deep learning frameworks, which are difficult to match with custom KDE implementations in practice.

5.4.8 Attention head specialization

To interpret the outputs of the Transformer model we visualize the attention of the eight individual heads in layer 0 in two different ways. First, in Figure 5.14 we show the full attention matrices as heatmaps. We choose the particle ordering so that nearby particles are close in the ordering. Second, in Figure 5.15 we choose a random query point, marked with a red x, and color other points according to the strength of the query point's attention. We clearly observe an emergent behavior – heads specialize in different tasks. Head 1 specialized to look at far-away points, whereas heads 0, 2, and 5 specialized to look at close- and mid-range interactions. Finally, heads 3, 4, 6, and 7 specialized to look in specific directions. This emergent behavior further links the multi-head Transformer to kernel-based methods, but with multi-scale learned kernels.

5.5 Conclusion

We introduced DiScoFormer, a permutation- and affine-equivariant Transformer that jointly estimates density and score from i.i.d. samples in a single forward pass. We proved that self-attention recovers normalized KDE and confirmed empirically that individual heads learn multi-scale kernel-like behaviors. DiScoFormer outperforms KDE across dimensions and sample sizes and serves as a plug-in oracle for score-debiased KDE, entropy and Fisher information estimation, and Fokker–Planck-type PDEs.

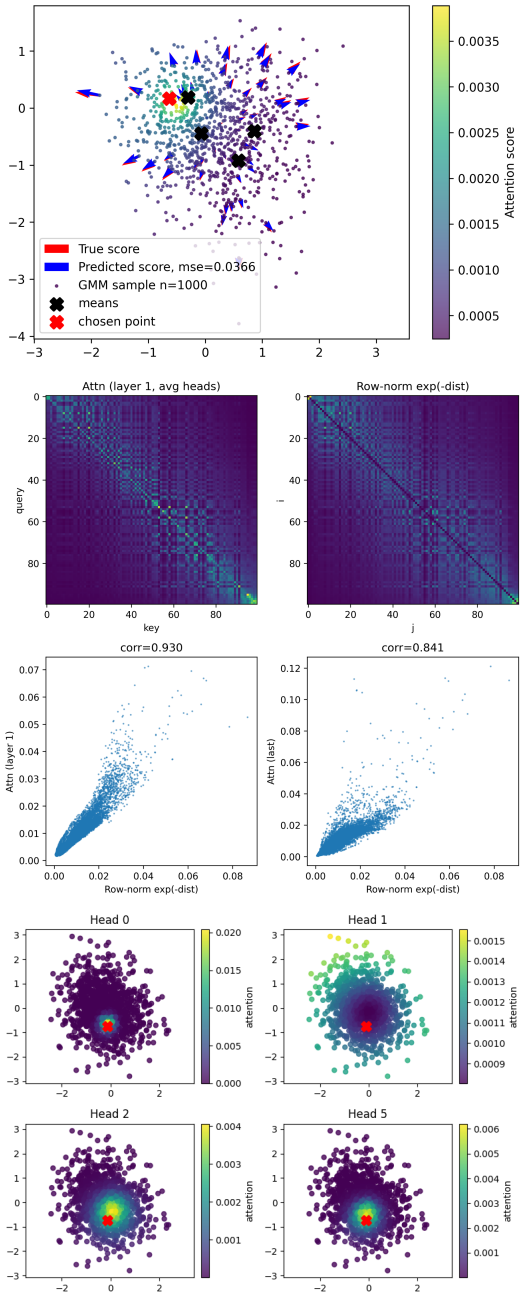


Figure 5.1: Attention visualization of the transformer model. The top panel shows the average attention scores in layer 0 with respect to the chosen point x. The heatmaps show the attention matrix and the normalized KDE matrix $D_{i,j} \propto e^{-\|x_i - x_j\|_2^2}$. The scatter plots show very high agreement between attention scores and D. The bottom four panels visualize the attention scores of individual heads, demonstrating emergent head specialization.

```
def forward(self, X, Y):
    m = X.mean(0, keepdim=True)
    Xc, Yc = X - m, Y - m
    S = Xc.T @ Xc + eps * torch.eye(d)
    A = matrix_sqrt_inv(S)
    Xw, Yw = Xc @ A, Yc @ A
    dens, score = self._core(Xw, Yw)
    return dens * torch.det(A), score @ A.T
```

Figure 5.2: The forward pass implementing affine equivariance.

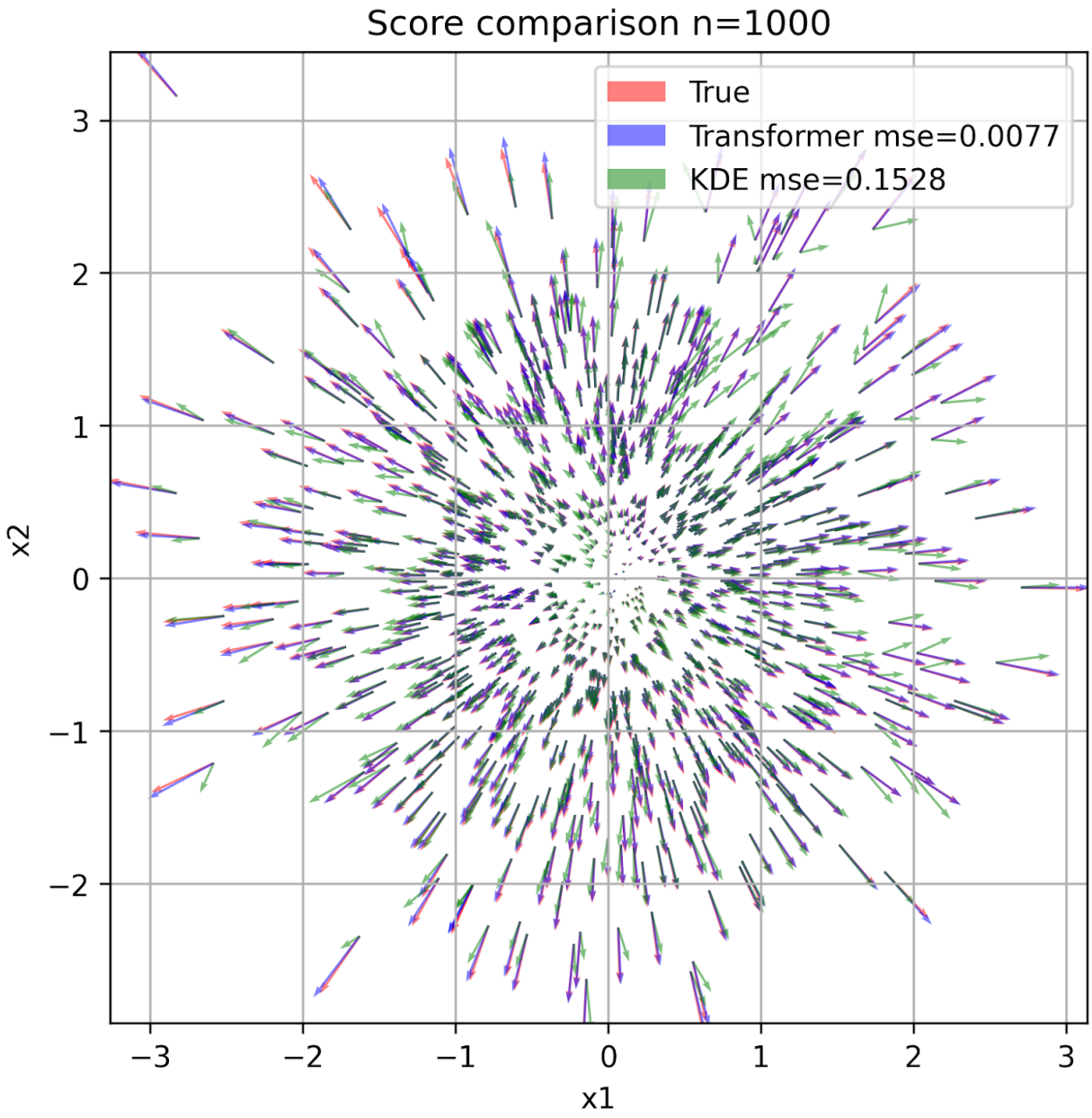


Figure 5.3: Score estimation comparison between Silverman KDE [156] and our transformer model. The transformer is more accurate, especially in the sparse regions. We plot the negated score for easier viewing.

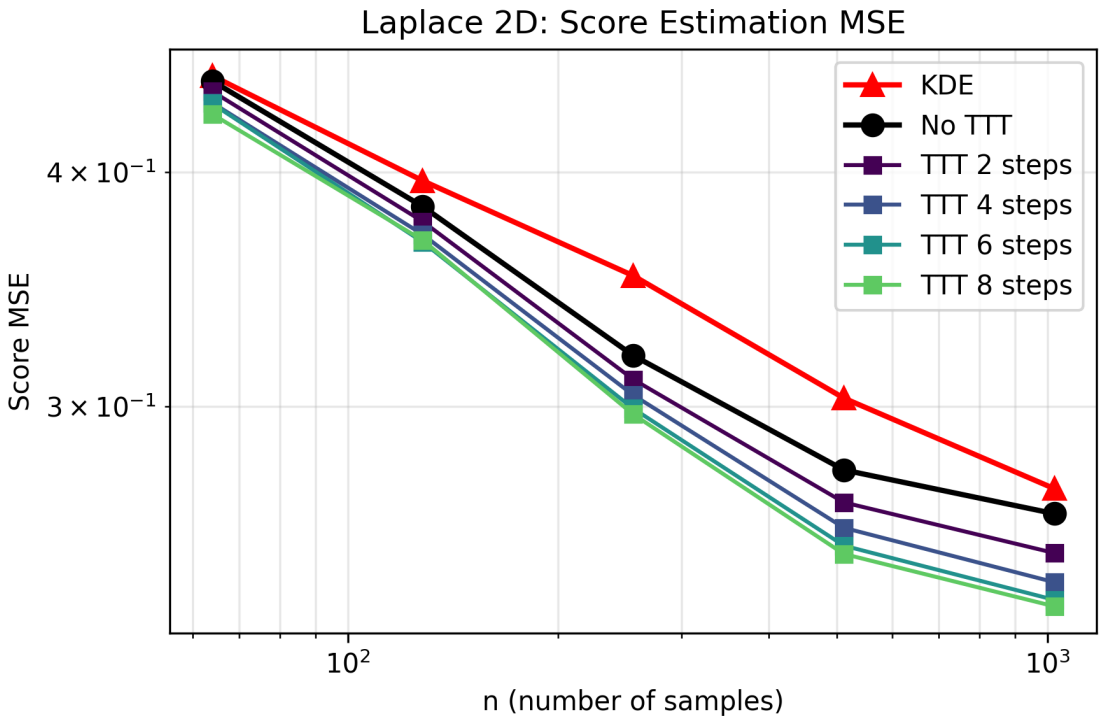


Figure 5.4: MSE of score estimation on the Laplace distribution. Test-time training (TTT) improves the out-of-distribution generalization.

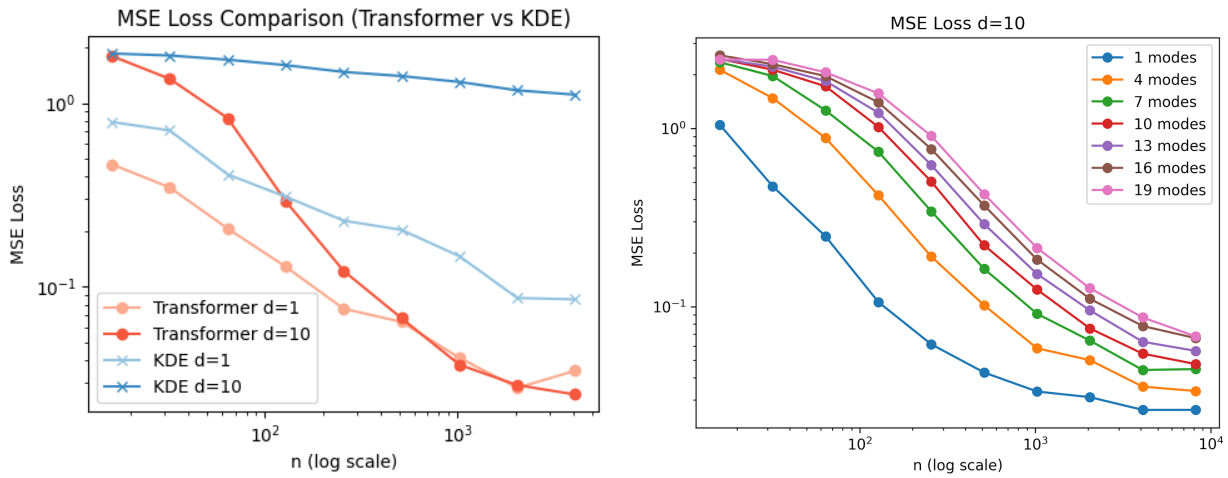


Figure 5.5: Left: MSE of score estimation in dimensions 1 and 10 on a 3-modal GMM using Transformer and KDE. The Transformer has excellent scaling in both dimension d and the sample size n. Right: MSE of score estimation using the Transformer on GMMs with different numbers of modes. Despite being trained only on GMMs with 1–10 modes, and $n = 2048$, the model exhibits excellent generalization.

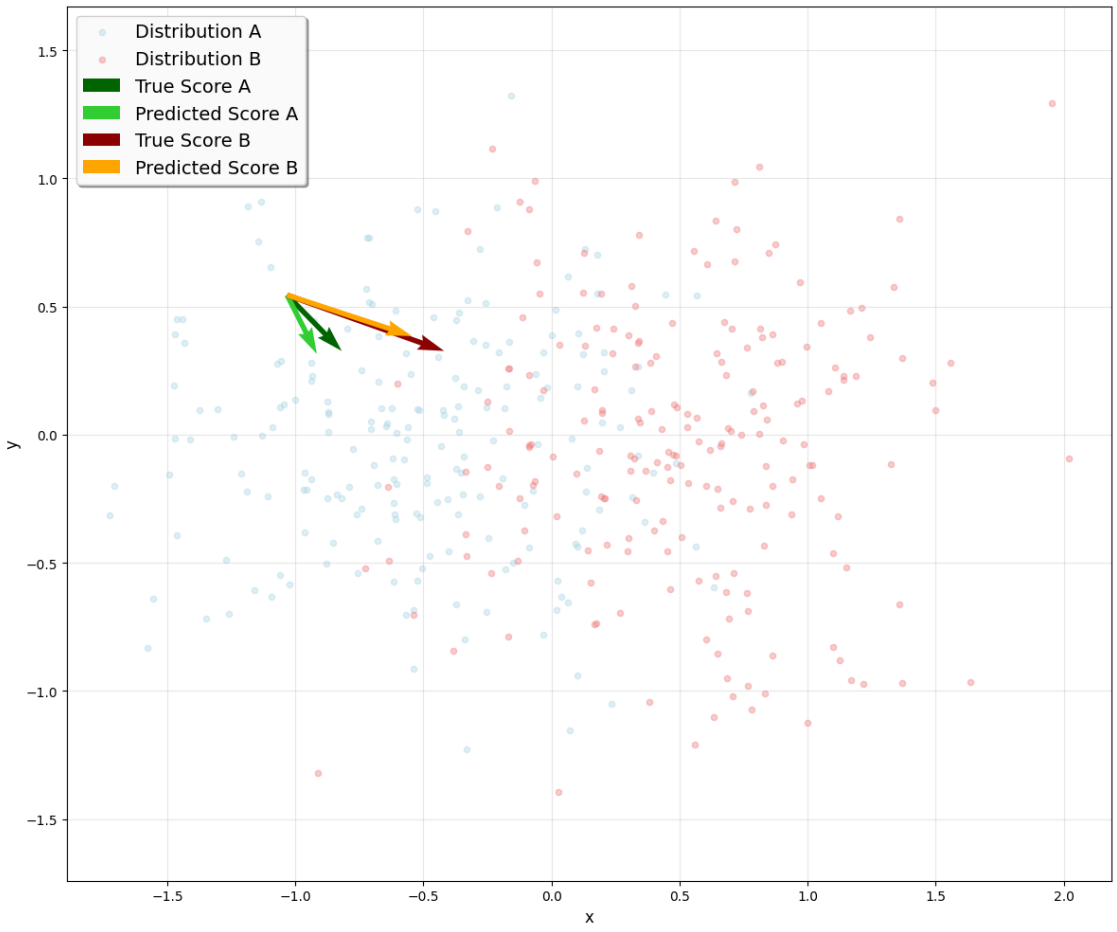


Figure 5.6: Computing relative Fisher information. Our model predicts $\nabla \log g(x_i)$ at query points x_i via cross-attention with the samples $y_i \sim g$. These predicted scores can be used to estimate the relative Fisher information.

Figure 5.7: Qualitative comparison: true density, Silverman KDE, Emp-SD-KDE, and SD-KDE with our learned score. Learned-score SD-KDE is visibly less biased.

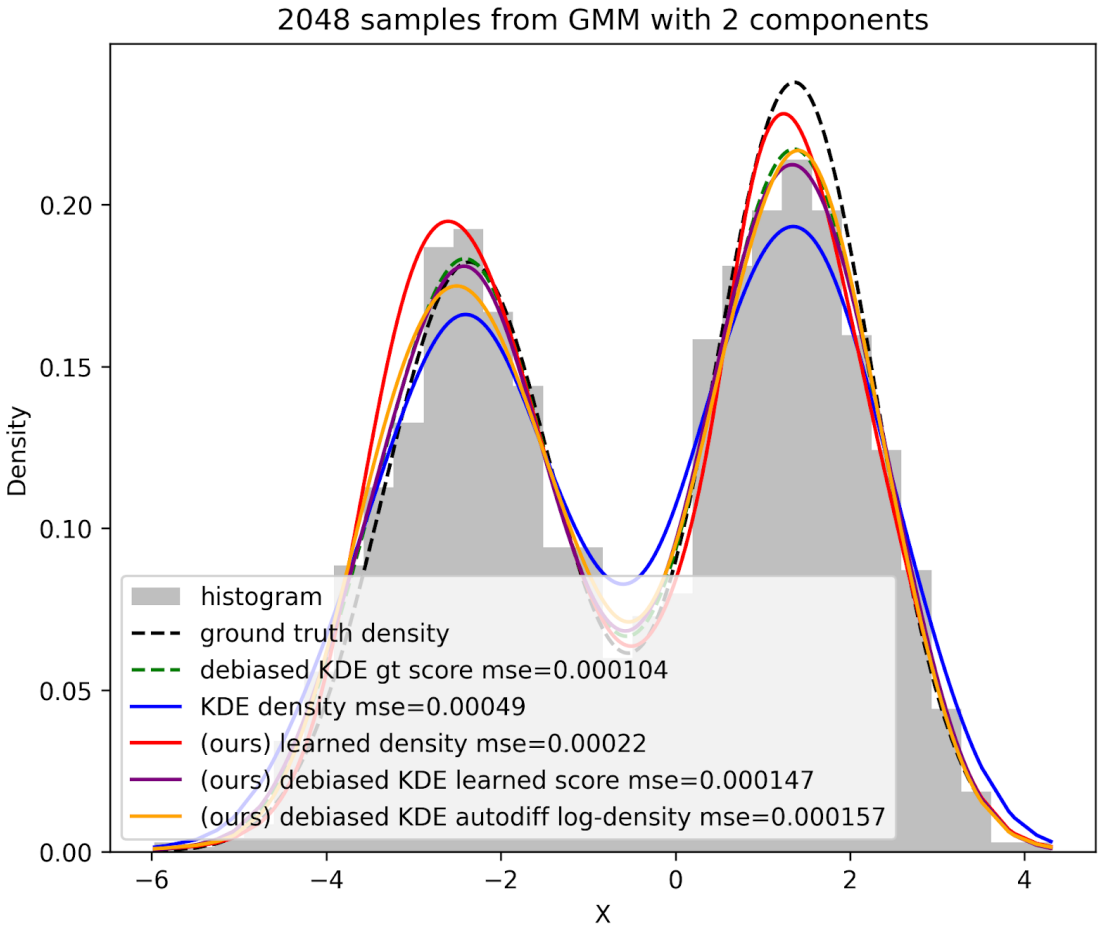


Figure 5.8: 1D bimodal GMM ($n = 2048$): SD-KDE with the learned score best approximates the density by MSE.

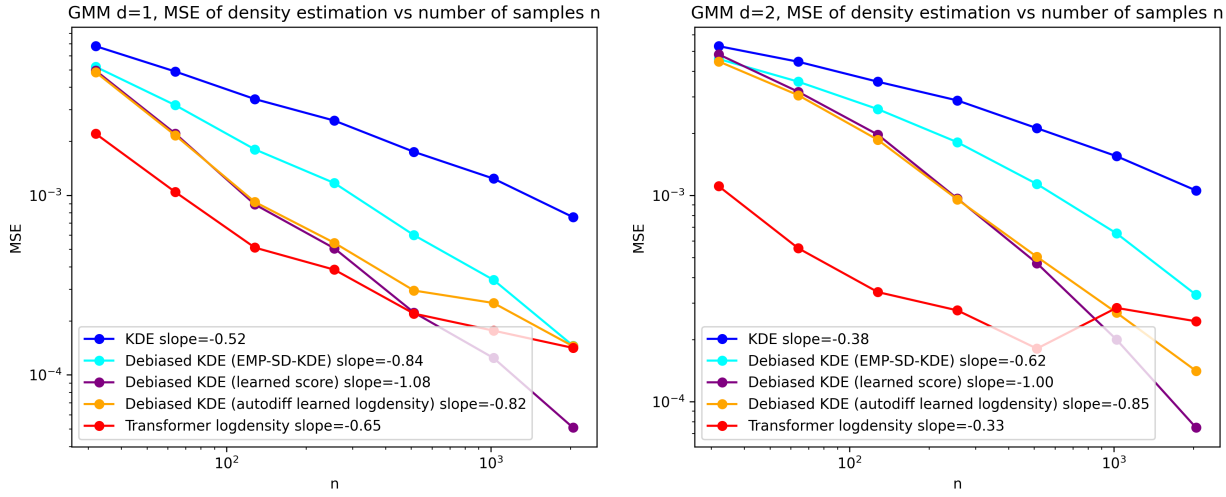


Figure 5.9: MSE of density estimation in 1D (left) and 2D (right). SD-KDE with our learned score and the Transformer model show the best scaling. The Transformer was trained only at $n = 2048$.

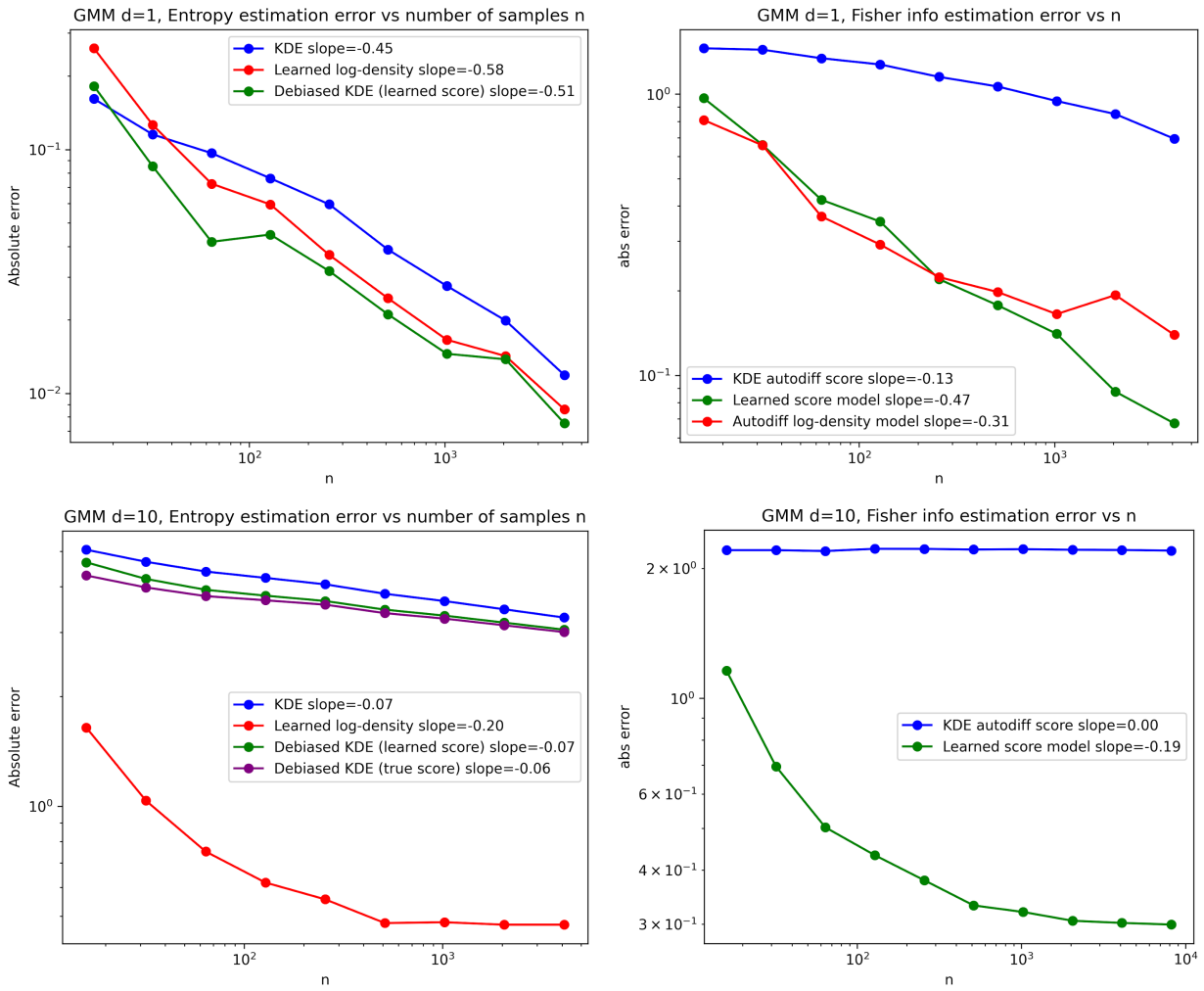


Figure 5.10: Comparison between the transformer model (learned), Silverman KDE, and score-debiased KDE for estimation of differential entropy $H(f)$ and Fisher Information $I(f)$. The Transformer's MSE is lower than that of the KDE approximation, even in dimension 1.

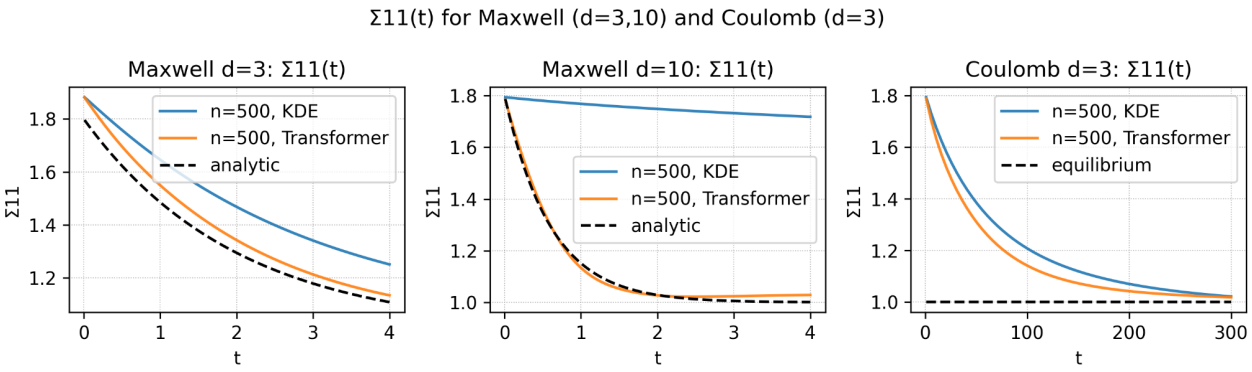


Figure 5.11: Comparison between the trained Transformer model and Silverman KDE on the task of numerically solving the homogeneous Landau equation. We plot the first entry of the covariance matrix $\Sigma_{1,1}(t)$ of the numerical solutions and ground truth, when known. The left two panels use Maxwell collisions, while the last panel shows Coulomb collisions. The Transformer outperforms KDE, and is comparable in quality to SBTM in [91].

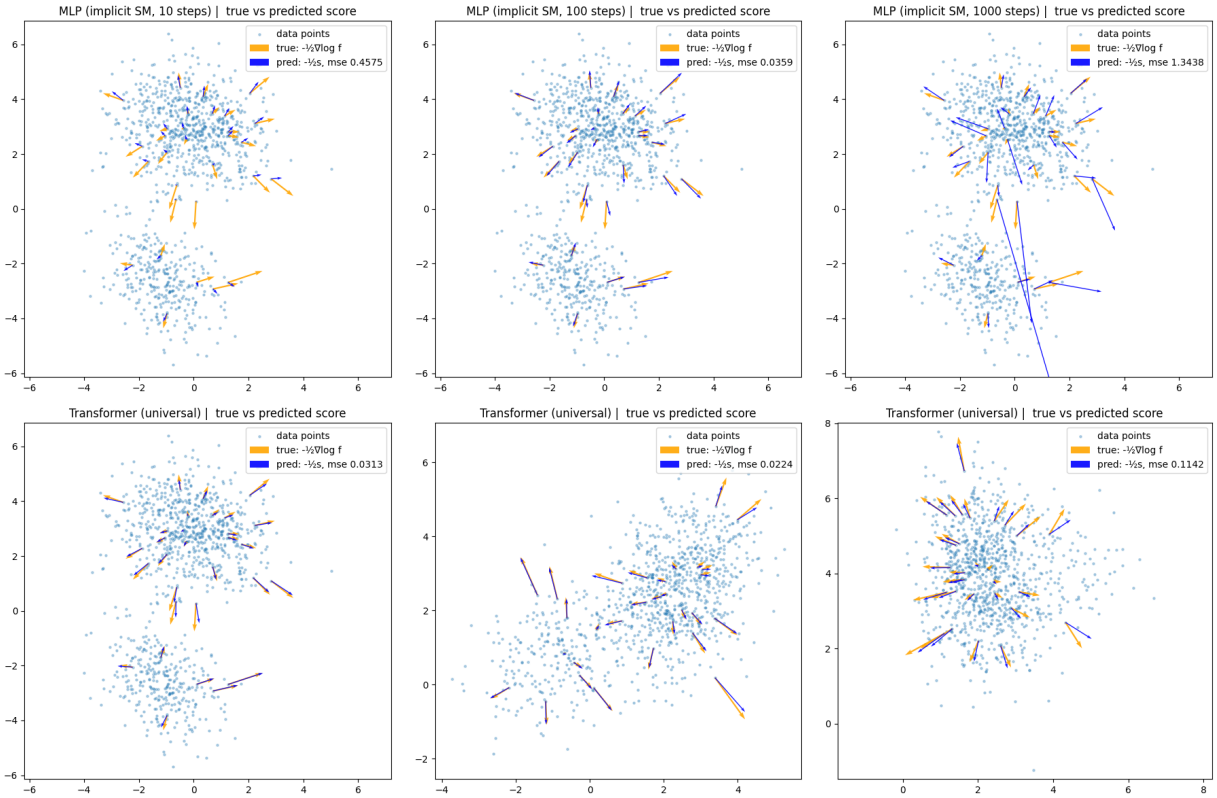


Figure 5.12: Comparison of sliced score matching and our transformer model. The transformer model can be used without retraining and does not suffer from overfitting. We plot the negated score for ease of visualization.

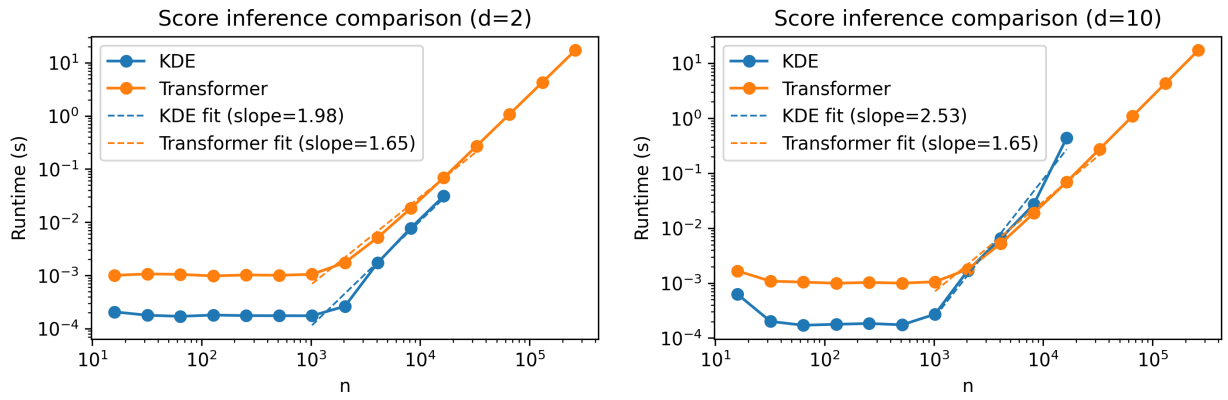


Figure 5.13: Runtime comparison between KDE and the Transformer model in 2 and 10 dimensions. Both are $O(n^2)$ asymptotically, but empirically the Transformer scales better and has improved memory efficiency. KDE encounters an OOM error at $n = 2^{15}$.

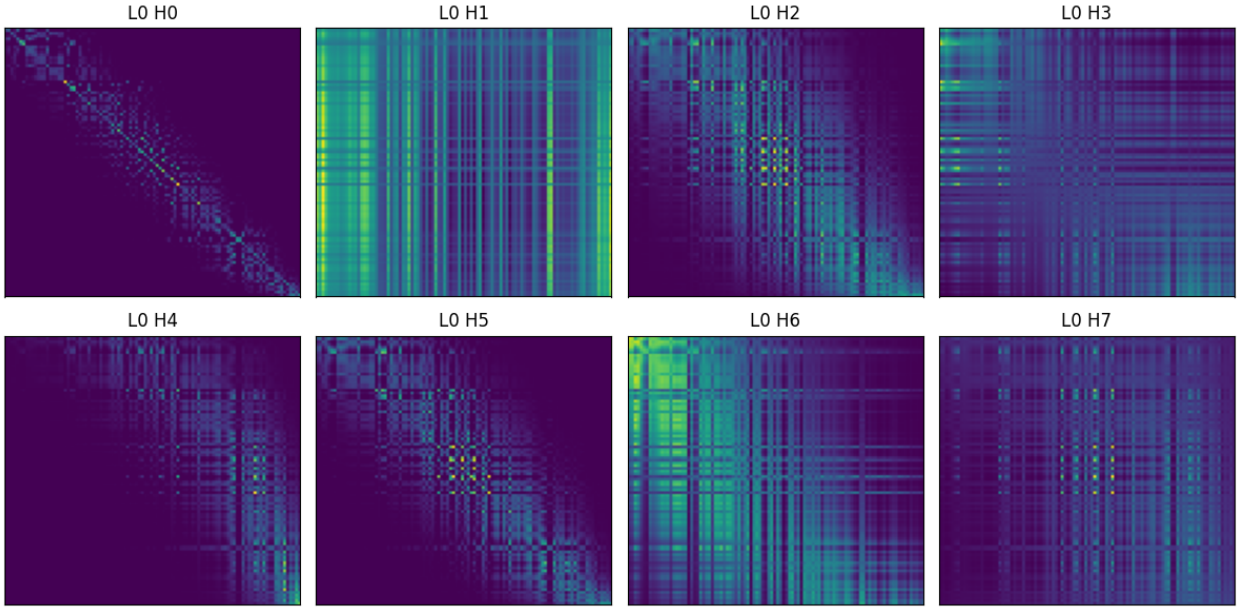


Figure 5.14: Attention of the eight individual heads of layer 0 as a heatmap. Yellow color denotes higher attention weight and blue is lower. We choose the particle ordering so that nearby particles are close in the ordering. Heads 0, 2, and 5 specialize on nearby points, heads 1 and 6 specialize on far-away points, whereas heads 3, 4, and 7 attend in specific directions.

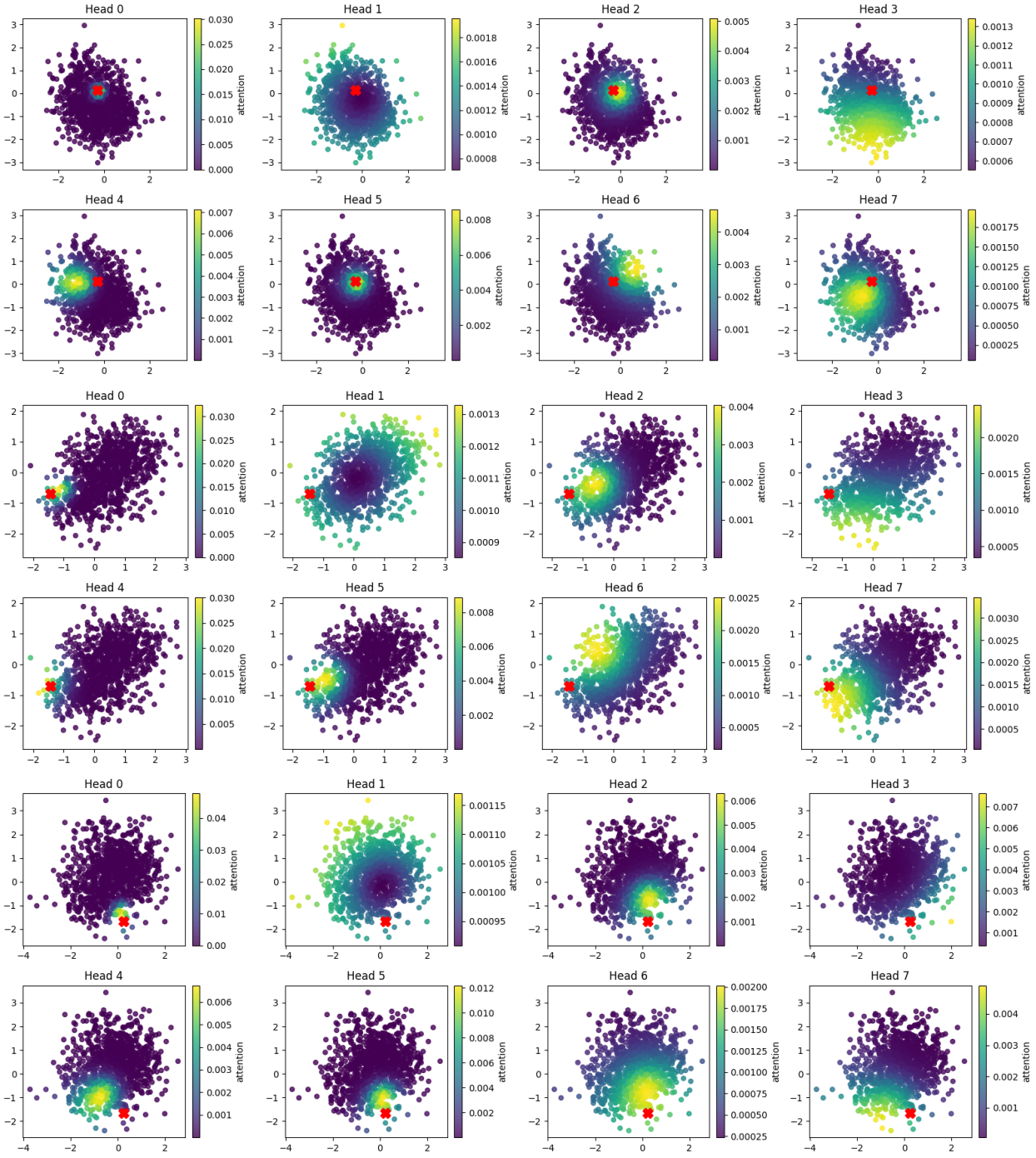


Figure 5.15: Attention of the eight individual heads of layer 0 as scatter plots. We choose a random query point, marked with a red x. Yellow color denotes higher attention weight and blue is lower. Heads 0, 2, and 5 specialize on nearby points, head 1 specializes on far-away points, whereas heads 3, 4, 6, and 7 attend in specific directions.

Chapter 6

SEMANTIC SEARCH OVER 9 MILLION MATHEMATICAL THEOREMS

This chapter opens Part III, which shifts the dissertation from neural methods for plasma simulation (Part I) and score-based sampling and density estimation (Part II) to AI-assisted tools for mathematics itself. Where the previous parts use neural networks to drive numerical solvers, this part uses them to help mathematicians and proof assistants find, repair, and verify mathematical content. The three chapters of Part III mirror the workflow of doing mathematics: *discovery* (Chapter 6), *repair* (Chapter 7), and *formalization* (Chapter 8).

The discovery problem this chapter addresses is mundane but pervasive: a mathematician (or an AI agent) wants to find a specific theorem in the literature, not a paper that might contain it. Tools like Google Scholar, arXiv, and even modern web-augmented LLMs operate at the document level; matching the granularity of mathematical knowledge itself requires treating individual theorem statements as first-class retrieval objects. We therefore extract a corpus of 9.2 million theorem statements from arXiv and seven other mathematical sources – the largest unified collection of human-authored, research-level theorems to date – represent each with a short natural-language "slogan" generated by a large language model, and embed slogans and queries into a shared semantic space. On 111 queries written blind by professional mathematicians, the resulting search engine achieves 45.0% Hit@20 at the theorem level, more than double the 19.8%–27.0% obtained by ChatGPT 5.2 with search and Gemini 3 Pro.

This chapter is based on [5].

6.1 Introduction

Mathematical knowledge is organized around discrete results: theorems, lemmas, propositions, and corollaries. These statements serve as the fundamental units of reasoning for both human mathematicians and automated proof systems [144, 202, 196]. A researcher proving a new result must first determine whether the statement already exists in the literature, and similarly, an AI agent generating formal proofs benefits from retrieving relevant lemmas to guide its search. Yet most existing tools – Google Scholar, arXiv, and even modern LLMs with web access – operate at the level of entire documents, forcing users to manually scan papers when they seek a specific statement.

This gap is increasingly significant. arXiv hosts over 2.4 million papers, including more than 690,000 in mathematics [70]. A study of over 14,000 withdrawn arXiv preprints found that 2.5% were retracted because the authors' results already appeared in prior literature [147]. For example, [145, 211, 155] were withdrawn after discovering their main results had been previously established. AI systems face the same problem: the Erdős Problems Project documented cases where AI tools "solved" open problems that had been established decades earlier [61, 195, 103] – underscoring the need for theorem-level search.

In this work, we construct a corpus of over 9 million theorem statements from arXiv, the Stacks Project, ProofWiki, and five other sources, and study semantic retrieval at scale. We represent each theorem using a natural-language "slogan" generated by an LLM, then embed slogans and queries into a shared semantic space. Our main contributions are:

1. **A large-scale theorem corpus.** We release over 9 million theorem statements with rich metadata, the *largest* collection of informal mathematical theorems to date.

2. **A systematic study of representation choices.** We analyze how the context, LLM choice, embedding model, and prompting strategy affect retrieval. Notably, we find that embedding theorems via natural-language slogans significantly outperforms embedding their raw LaTeX formulations.

3. **State-of-the-art retrieval.** On 111 queries from professional mathematicians, we achieve 45.0% Hit@20 at the theorem level, outperforming ChatGPT 5.2 with search (19.8%) and Gemini 3 Pro (27.0%). For paper-level retrieval, we achieve 56.8% Hit@20 compared to 37.8% for Google Search.

Our results demonstrate that semantic theorem search is feasible at web scale. The project page is at theoremsearch.com. The dataset is available at huggingface.co/datasets/uw-math-ai/theorem-search-dataset.

6.2 Related work

Mathematical information retrieval. Early mathematical IR work focused on formula-level retrieval, with the NTCIR Math Tasks [4, 209] establishing benchmarks for formula search over arXiv and Wikipedia. The ARQMath shared tasks [132] extended this to mathematical question answering over Math Stack Exchange. A recent survey [47] notes that while formula retrieval has progressed, semantic understanding of mathematical statements remains an open challenge – the gap our work addresses.

Dense retrieval and LLM-augmented search. Dense Passage Retrieval [99] showed that dual-encoder architectures can outperform sparse methods like BM25. Sentence-BERT [148] enabled efficient semantic similarity via Siamese networks, while E5 [190], Qwen3 Embedding [212], and Gemma Embedding [182] have pushed embedding quality further. ColBERT [102] introduced late interaction for fine-grained token matching. Retrieval-augmented generation (RAG) [109] combines retrieval with language model generation, enabling systems to ground responses in retrieved documents. These advances underpin our approach, though mathematical text poses unique challenges due to symbolic notation.

Search for formal mathematics. LeanSearch [67] provides semantic search over Mathlib4's 230,000+ theorems by generating natural-language descriptions and using dense retrieval. ReProver [202] uses retrieval-augmented generation to select premises during proof

search, and Numina-Lean-Agent [116] integrates multiple search tools, including LeanDex, to retrieve lemmas across libraries. LeanFinder [127] focuses on user intent, while LeanExplore [12] combines embeddings with BM25+ and PageRank. [96] demonstrate that language models can translate between formal and natural language mathematics. However, formal libraries cover only a fraction of mathematical knowledge; our work extends informalization-based retrieval to millions of LaTeX theorem statements.

Scientific literature search. General academic search engines (Google Scholar, Semantic Scholar) and arXiv provide paper-level retrieval but cannot target individual theorems. The zbMATH database [167] offers curated paper-level indexing. Large language models with web access, such as GPT-4 [2] and Gemini [72], can answer mathematical questions and sometimes locate relevant papers, but as we show in our experiments, they often provide incorrect theorem references or fail to locate specific statements. Our work treats theorem statements as first-class retrieval objects, enabling users to find specific results rather than papers that might contain them.

6.3 Data collection

Figure 6.1 gives an overview of the pipeline used to parse theorem statements from source documents, generate natural-language representations, and embed them for retrieval.

Table 6.1: Distribution of theorem types in our dataset.

Theorem Type	**Count**
Lemma	3,280,463
Theorem	2,864,668
Proposition	2,067,371
Corollary	1,034,259

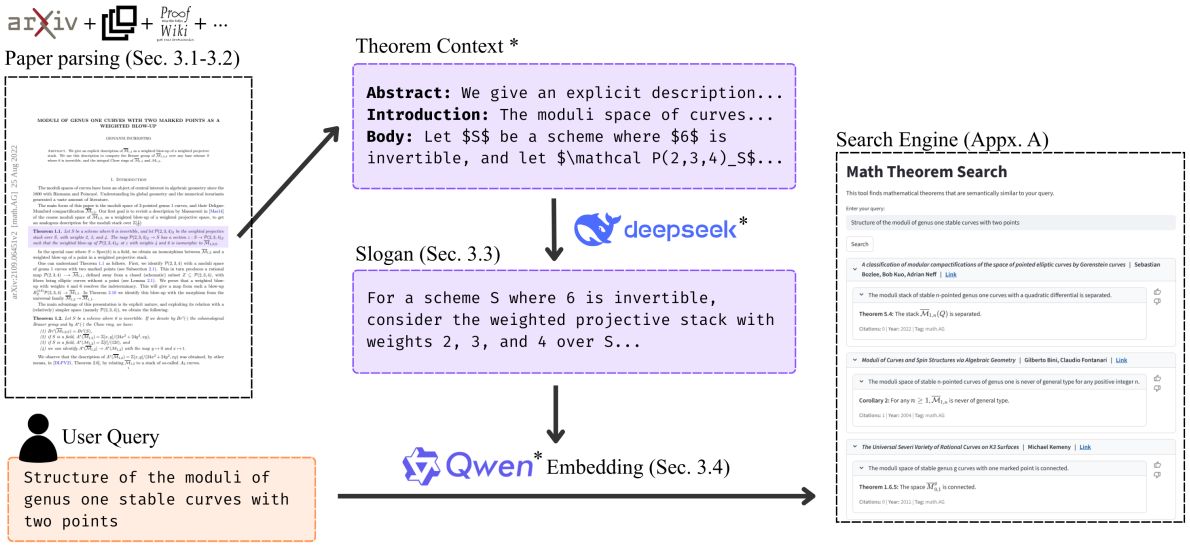


Figure 6.1: Overview of the creation of our database and search engine. Items with a "*" are variable and we tested different methods during our experiments (Section 6.4).

6.3.1 Theorem parsing from arXiv

We source 99.5% (9.2 million) of our theorems from papers in the arXiv database tagged with `math`, `stat`, `cs`, `physics`, `eess`, `econ`, `q-fin`, or `q-bio`. A breakdown of the statement types and arXiv categories across the dataset is provided in Table 6.1 and Figure 6.2; the full arXiv-tag distribution is reported in the project repository and the source paper.

Our objective in parsing theorems is to extract theorem *names* and *bodies*. A theorem *name* has three components:

1. the theorem type (e.g. Theorem, Proposition, Lemma, or Corollary);
2. an optional reference number; and
3. an optional note. For example, a theorem name might look like "Theorem 3.9 (Shokurov reduction)".

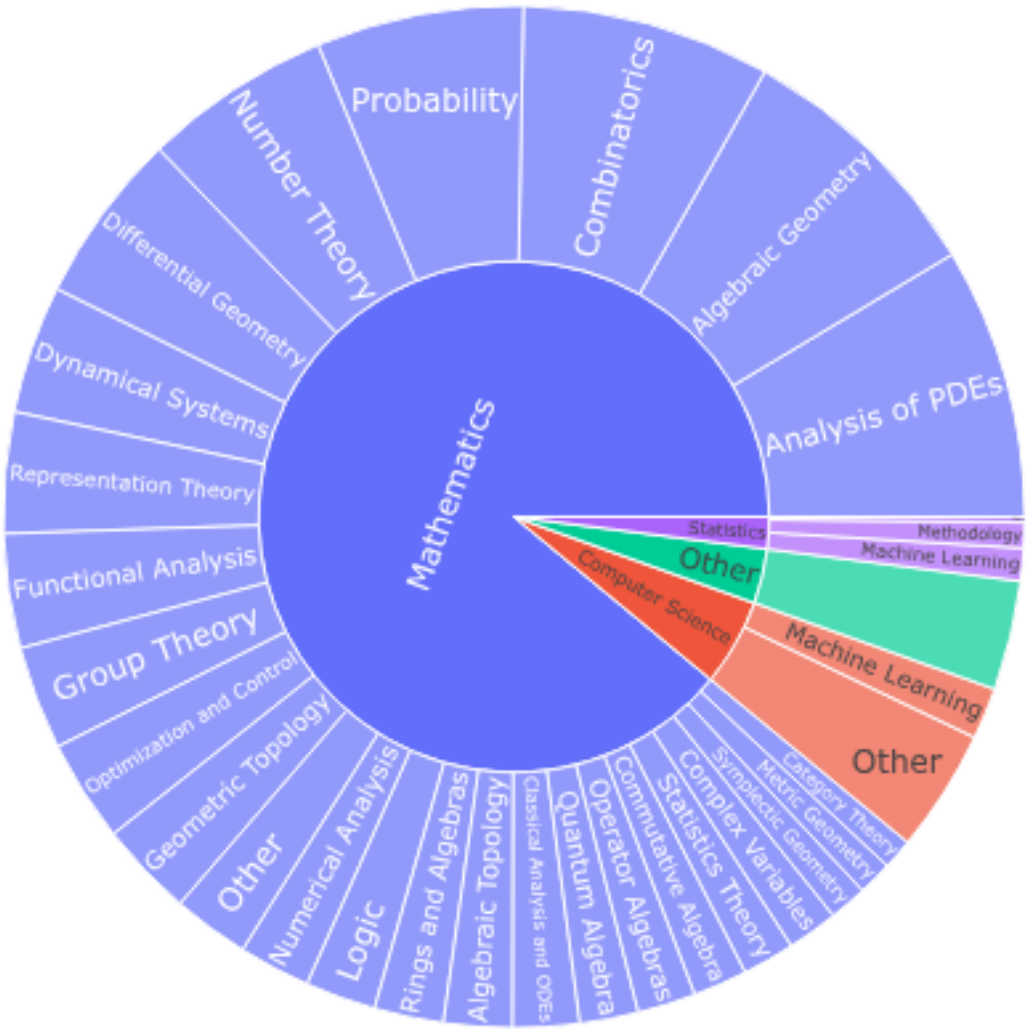


Figure 6.2: Sunburst plot of the distribution of arXiv tags across theorems in our dataset.

We define the *body* as the LaTeX content of the theorem, in which basic author-defined macros are expanded. For instance, if an author defines `\R` as a macro for `\mathbb{R}` in the LaTeX preamble, we replace all occurrences of `\R` accordingly.

The variety of ways authors define theorem environments makes parsing theorems difficult. Thus, we parse theorems for each paper using three strategies:

1. *Node search with plasTeX.* We use the Python library plasTeX [8] to convert LaTeX sources into a structured node tree. Every command and environment corresponds to a node. We traverse this tree to find environments corresponding to theorem-like structures. Then, we extract the theorem type, reference number, note, and body from the node metadata. This method successfully parses 422 thousand papers, yielding approximately 6.9 million theorems. However, plasTeX occasionally truncates theorem bodies when a paper's source relies on LaTeX packages unknown to the parser. To remove such malformed extractions, we apply simple heuristics, such as filtering out theorems shorter than 8 characters or ending in " and" or " let".

2. *TeX logging.* As a fallback to plasTeX, we generate and inject a custom LaTeX package that logs theorem data. When a paper's source is compiled, the package records the theorem type, reference number, note, and body for all theorem environments. We parse 137 thousand papers this way, yielding approximately 1.8 million theorems.

3. *Regex-based parsing.* As a fallback to TeX logging, we use regular expressions to identify theorem delimiter tokens such as `\begin{...}` and `\end{...}` or `\proclaim` and `\endproclaim`. We then parse the content between delimiters to extract the theorem's note and body. Reference numbers are not captured unless they are explicitly stated. We parse 30,000 papers this way, yielding approximately 542,000 theorems.

6.3.2 Theorem parsing from other sources

The remaining 0.5% (38,974) of our theorems come from a variety of other sources: ProofWiki [146] (23,871), the Stacks Project [175] (12,693), the Open Logic Project [174] (745), the CRing Project [172] (546), Stacks and Moduli [7] (506), the HoTT Book [176] (382), and An Infinitely Large Napkin [32] (231). These sources were chosen for their verifiability through

open-source contributions and structured, accessible LaTeX files. We favored graduate-level texts, as their theorems would be useful to a mathematician and are often cited by researchers.

The Stacks Project, Open Logic Project, CRing Project, HoTT Book, and Infinitely Large Napkin are all hosted as open-source repositories on GitHub. Theorems from these sources were exclusively delimited in the source by `\begin{...}` and `\end{...}` tags. This uniform structure allowed us to build a single regex-based parser, similar to our regex-based parser for arXiv, that could extract theorems from all five sources. Our parser first normalizes shorthand environment names, then extracts the theorem name, body, label, and type from each environment. When a theorem body begins with a bracketed note, we append it to the theorem name in parentheses for easier identification. Labels defined with `\label{...}` are captured to preserve cross-referencing information from the original text. Since these repositories use GitHub's file structure, we generate source URLs pointing to the original `.tex` files, enabling users to verify theorems against their source material. In addition, a custom theorem counter was used to reliably generate reference numbers.

ProofWiki required a different approach, as it is a wiki-based encyclopedia built on MediaWiki rather than LaTeX. ProofWiki stores content in wikitext format with mathematical expressions enclosed in `<math>...</math>` tags. We developed a separate parser that interfaces with the MediaWiki API to programmatically retrieve pages from relevant categories. For each page, we extract the theorem statement by identifying section headers and capturing content up to the proof section. The wikitext is then cleaned by removing MediaWiki-specific markup such as `<onlyinclude>` tags, template calls, and wiki links, while converting `<math>` blocks to standard delimiters. Since ProofWiki assigns a unique URL to each theorem, we preserve these links to allow direct verification.

6.3.3 Theorem representation

Theorems in our corpus are exclusively represented in LaTeX, and often lack concise natural-language summaries. To obtain searchable textual representations and improve retrieval

performance, we generate a short natural-language description, or *slogan*, for each theorem using a large language model (LLM). This converts the retrieval task from a symmetric search over formal notation to an asymmetric task where informal queries retrieve formalized content [190]. Given the parsed theorem body, we prompt the DeepSeek V3 model [114] to produce a concise, declarative English description of the theorem's main result. Prompts instruct the LLM to avoid symbolic notation, proof details, and references to surrounding document structure in the generated slogans. The resulting slogans are stored as the primary textual representation for theorem retrieval.

To study the effect of additional mathematical context on slogan quality and retrieval performance, we evaluate three slogan-generation strategies:

1. **Body Only**: the prompt includes only the parsed theorem body.

2. **Body + Abstract**: the prompt includes the theorem body together with the paper abstract.

3. **Body + Introduction**: the prompt includes the theorem body together with the paper introduction.

Each strategy uses a similar prompt template (the exact wording is listed in the source paper). Their major differences are in the contextual fields provided to the model. Model temperature is fixed at 0.2, and the maximum output tokens is fixed at 1024. Slogans are generated independently for each variant and treated as separate representations of the same underlying theorem during evaluation. The total cost of building the corpus was approximately $6,000 USD: $4,000 for LLM API calls to generate slogans across all 9.2 million theorems, and $2,000 for compute and S3 storage.

6.3.4 Result retrieval

We embed theorem slogans and user queries using the Qwen3-Embedding-8B model [212], which maps natural-language inputs to fixed-dimensional vectors. All theorem slogans are

embedded offline following their generation by the LLM and stored in a PostgreSQL database [168] with the pgvector extension [142]. The database employs a Hierarchical Navigable Small World (HNSW) index [131] combined with binary quantization, enabling fast approximate nearest-neighbor search. User queries are embedded at inference time using the same method. Within this shared embedding space, we retrieve the top-k theorems ranked by their Hamming distance, then re-ranked by cosine similarity.

6.3.5 Validation set

Enlisting the help of three research mathematicians, we curated a set of 111 distinct math queries across 14 arXiv tags, mainly Algebraic Geometry, Analysis, and PDEs. These queries search for distinct theorems, lemmas, corollaries, and propositions written by a small number of authors whose work our subject-matter experts are well acquainted with. Crucially, queries were written *blind*: each mathematician composed natural-language descriptions of theorems they knew from memory, without access to the corpus or its slogans, to prevent any leakage of retrieval representations into the evaluation queries. We then asked them to select a larger dataset of papers in which they were confident that the results of the queries they wrote appeared at most once; this larger dataset consisted of 7,356 papers. We used this larger dataset to guide our decisions on which embedders to use, which LLM to use for the slogans, how to prompt the embedder, and how much context to give to the LLM when generating the slogans (body of the result only, body+abstract, body+first section). Every query–theorem pair was subsequently verified in a two-stage quality check: an LLM confirmed that the target theorem exists in the corpus and semantically matches the query, and a second mathematician independently reviewed each pair to ensure correctness.

We remark that while small, this size is typical of mathematical information retrieval datasets, such as the query set of LeanSearch for Mathlib4 [67] and ARQMath-3 [132], which contain 50 and 78 queries, respectively.

6.4 Experiments

In this section, we report the evaluation performance of existing retrieval methods against our search engine. We outline the experiment setup, compare the performance against existing retrieval tools, and perform an ablation study on both the context window and the LLM used for slogan generation.

6.4.1 Metrics

To measure the performance of our search engine against existing literature-review tools, we employ two commonly used information retrieval metrics, following standard practice in mathematical retrieval benchmarks [132, 47]. Hit@k, also known as the hit rate, measures whether the correct result appears anywhere in the top k results:

$$\text{Hit@}k = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \max_{1 \le j \le k} \mathbb{I}(r_{i,j}),$$

where $\mathcal{Q}$ is the set of queries, $r_{i,j}$ is the j-th result of the i-th query, and $\mathbb{I}(r_{i,j})$ equals 1 if and only if $r_{i,j}$ is an exact match. We also use Mean Reciprocal Rank (MRR@k) [46], which takes the reciprocal of the rank at which an exact match is discovered:

$$\text{MRR@}k = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \frac{1}{\text{rank}_i},$$

where rank_i is the position of the first relevant result.

6.4.2 Setup

We chose three embedders to use in our experiments: Google's Gemma 0.3B [182], and both Qwen3 0.6B and 8B [212]. Our baseline methods consist of the following: Filtered Google Search (`site:arxiv.org` + Query), arXiv advanced search, ChatGPT 5.2 with search, and Gemini 3 Pro [72].

Both Google and arXiv are unable to return specific math statements, so their performance is graded on their ability to locate the correct paper. Conversely, LLM-based systems

like ChatGPT and Gemini occasionally return correct results with incorrect reference numbers; we grade these as misses on theorem-level retrieval and treat them as paper matches for paper-level retrieval.

We emphasize that these baselines are not designed for theorem-level retrieval, but we evaluate them to reflect the tools currently used by LLMs and research mathematicians in the absence of specialized theorem-level search systems.

6.4.3 Main results

Table 6.2: Results on the validation set, comparing embedder performance against existing literature-search tools. Values are reported as theorem-level / paper-level. Google Search and arXiv search cannot return specific theorem statements, so only paper-level results are reported. Our corpus was restricted to arXiv papers for evaluation.

Model	Hit@1	Hit@10	Hit@20	MRR@20
Baseline				
Google Search†	0.162	0.378	0.378	0.237
arXiv Search†	0.009	0.018	0.027	0.011
ChatGPT 5.2	0.117	0.180	0.198	0.139
Gemini 3 Pro	0.171	0.252	0.270	0.196
Our methods				
Gemma 0.3B (LaTeX)	0.027 / 0.054	0.090 / 0.117	0.090 / 0.135	0.041 / 0.070
Gemma 0.3B	0.081 / 0.108	0.189 / 0.225	0.252 / 0.306	0.118 / 0.154
Qwen3 0.6B	0.081 / 0.153	0.234 / 0.342	0.270 / 0.351	0.132 / 0.215
Qwen3 8B	0.171 / 0.243	0.387 / 0.505	**0.450** / 0.568	0.243 / 0.328
Qwen3 8B + Reranker	**0.189** / **0.324**	**0.432** / **0.613**	**0.450** / **0.631**	**0.270** / **0.416**

†Paper-level only: these tools cannot return individual theorems.

Our results are presented in Table 6.2. We find that the embedder Qwen3 8B outperforms

existing literature-review methods across all metrics. In particular, Qwen3 8B achieves substantially higher Hit@10 and Hit@20, indicating that the correct theorem is frequently retrieved within the top candidate set even when it is not ranked first. This behavior is desirable in large-scale theorem retrieval, where downstream reranking or human inspection can refine results once relevant candidates have been surfaced. We further apply a cross-encoder reranker (Qwen3-Reranker-0.6B [212]) to rescore the top-100 candidates retrieved by Qwen3 8B; this improves theorem-level Hit@1 from 17.1% to 18.9% and MRR@20 from 24.3% to 27.0%, confirming that late interaction over full query–slogan pairs captures fine-grained semantic distinctions that the bi-encoder alone misses.

For search methods at the paper level, Qwen3 8B continues to outperform the baseline models, achieving higher Hit@1 and MRR. The arXiv search is limited in the fields it can search over and does not search within the paper body, so it was only able to locate two papers across the whole query set in the top-20 results.

One notable effect we observed concerns how ChatGPT and Gemini rank retrieved results. Although both models used web-search-based retrieval, they would often organize their outputs primarily at the paper level, returning multiple theorems from the same paper in consecutive positions before moving to another paper. This ranking pattern reduces result diversity, which lowers Hit@k despite sometimes achieving high early precision. On average, Gemini returns only 10.98 distinct papers per query, compared to 16.89 for Qwen3 8B, indicating that our retrieval pipeline surfaces a broader cross-section of the literature.

Our approach is particularly advantageous for retrieving auxiliary lemmas and technical results that appear deep within a paper, far from the title and abstract. Google Search and LLM-based tools rely heavily on paper-level metadata and thus struggle to surface results whose content is not reflected in the abstract, whereas our system indexes every theorem independently via its slogan, making it equally as capable of retrieving a minor lemma in a later section as a paper's headline theorem.

6.4.4 Retrieval-augmented LLM reasoning

Beyond quantitative retrieval metrics, we investigate whether our theorem corpus can improve LLM reasoning on research-level mathematical questions. We posed a question about the boundary of a KSBA compactification of elliptic surface pairs to Claude (Opus 4.5). Without retrieval augmentation, Claude produced a confidently stated but *incorrect* answer, concluding that all boundary pairs must retain a fibration structure. The argument, while superficially plausible, relied on incorrect claims about the behavior of slc surface pairs under KSBA compactification.

When the same question was posed with access to our theorem database as a RAG tool, Claude retrieved relevant results from [10, 11, 94, 9] and arrived at the *correct* answer: such boundary pairs do exist, and they are pseudoelliptic surfaces obtained by contracting the section of an elliptic fibration. The RAG-augmented response cited specific theorem numbers and provided a logically valid chain of reasoning grounded in retrieved results. This illustrates a key failure mode of LLM reasoning – without access to the relevant literature, the model confabulates plausible-sounding but incorrect mathematical arguments – and demonstrates that retrieval augmentation over our corpus can directly address it. The full query and both responses are reproduced in Section 6.8.

6.4.5 Ablation studies

We perform ablation studies on the context windows used for slogan generation, the LLMs used to generate slogans, and the document preparation process. For each ablation, we limit the corpus to arXiv papers labeled with the algebraic-geometry tag (math.AG) and authored by individuals whose work contains an exact match to one of our queries. This filtering reduces the corpus to 7,356 statements written by 8 primary authors.

Context windows

Table 6.3 presents the effects of context on slogan generation. Increasing or decreasing the amount of context provided to LLMs when generating math slogans considerably affects our search engine's performance, while holding the embedder constant. We find that supplying the LLM with additional paper context improves the retrieval performance of its generated slogans. Performance falters when the LLM is only given the abstract, but improves significantly with the inclusion of the first section of the paper, defined by `\section{}`. We attribute this improvement to the LLM making better sense of the statement with more context, especially since theorems alone tend to be at most 3–4 sentences long and may reference earlier sections of the paper. Furthermore, the introduction of most math papers typically outlines the results and the existing research upon which they build. As a result, context-rich slogans better capture the semantic intent of the theorem, yielding more reliable and robust retrieval across queries.

Table 6.3: Comparing context-window size in retrieval performance. Embedded with prompt using Qwen3 8B [212]. Slogans generated with DeepSeek V3 [114].

Model	Hit@1	Hit@10	Hit@20	MRR@20
Body Only	0.342	0.658	0.737	0.451
w/ Abstract	0.316	0.645	0.737	0.429
w/ First Section	**0.368**	**0.737**	**0.763**	**0.496**

Slogan generator

The LLM used to generate the slogans can also significantly affect our engine's search performance, as some LLMs are better trained on mathematical texts than others and thus generate better informal versions of math statements. We find that leading proprietary models, such as Claude's Opus 4.5 and Gemini 3 Pro [72], outperform open-source models

such as DeepSeek-V3 [114], as summarized in Table 6.4.

Table 6.4: Comparing LLM slogans in retrieval performance. Embedded with prompt using Qwen3 8B [212]. Body + Abstract.

Model	Hit@1	Hit@10	Hit@20	MRR@20
DeepSeek V3.1	0.316	0.645	0.737	0.429
DeepSeek R1	0.276	0.671	0.697	0.388
Gemini 3 Pro	0.368	0.750	0.816	0.507
Claude Opus 4.5	**0.395**	**0.776**	**0.842**	**0.536**

Task instruction

Table 6.5: Comparing prompting performance. Slogans generated by DeepSeek V3.1 [114] on the Body+Abstract context window.

Model	Hit@1	Hit@10	Hit@20	MRR@20
Unprompted				
Gemma 0.3B	0.303	0.539	0.566	0.376
Qwen3 0.6B	0.224	0.487	0.526	0.297
Qwen3 8B	0.250	0.553	0.618	0.332
Prompted				
Gemma 0.3B	0.197	0.434	0.487	0.265
Qwen3 0.6B	0.237	0.566	0.658	0.346
Qwen3 8B	**0.316**	**0.645**	**0.737**	**0.496**

We examine the impact of the task instructions used during embedding on retrieval performance in Table 6.5. We find that by providing a math-retrieval instruction, both

Qwen embedders achieve higher performance than without any instruction at all. However, this effect is reversed with Gemma, with empty instruction performing better, though worse than Qwen 8B with instruction. We use a modification of the prompts made by [67].

6.4.6 More experiments

We additionally ran our ablations with a larger pool of embedders, including Multilingual-E5-Large-Instruct [191], zbMath-Bert [167], and KaLM-Embedding-V2.5 [213, 81]. Table 6.6 extends the context-window comparison of Table 6.3 to all six embedders, and Table 6.7 extends the slogan-LLM comparison of Table 6.4 across the same expanded set of embedders. The qualitative trends from the smaller comparisons are preserved: adding the paper's first section to the LLM context improves retrieval, and Claude Opus 4.5 produces the strongest slogans for most embedders. zbMath-Bert performs poorly across all settings, consistent with the limited capacity of the small in-domain encoder relative to the larger general-purpose embedders.

6.4.7 Embedding-space analysis

Following [67, 96], we convert LaTeX theorem statements into natural-language slogans before embedding, since embedders struggle with symbol-heavy notation [19, 141] and mathematicians typically query in informal language. To verify that sloganization induces semantically meaningful structure, we use PCA and UMAP as diagnostic tools on a random sample of 10,000 theorems (1,000 from each of the 10 most common arXiv categories).

Figure 6.3 shows PCA projections for three categories: Algebraic Geometry, Probability Theory, and Statistics Theory. Conceptually distant fields (Algebraic Geometry vs. Probability) are well separated, while closely related fields (Probability vs. Statistics) overlap substantially – consistent with mathematical intuition. UMAP projections over all ten categories in Figure 6.4 reveal that Qwen3 8B produces tighter, better-separated clusters than Gemma 0.3B, consistent with the retrieval gap in Table 6.2.

Table 6.6: Extended results comparing context-window size in retrieval performance across six embedders. Embedded without task instructions.

Model	Hit@1	Hit@10	Hit@20	MRR@20
Body Only				
zbMath-Bert 0.3B	0.053	0.158	0.237	0.086
Gemma 0.3B	0.184	0.474	0.539	0.290
Qwen3 0.6B	0.224	0.461	0.513	0.299
Qwen3 8B	0.224	0.553	0.632	0.313
KaLM-Embedding-V2.5	0.131	0.355	0.434	0.203
Multilingual-E5-Large-Instruct	0.171	0.395	0.500	0.237
w/ Abstract				
zbMath-Bert 0.3B	0.092	0.184	0.237	0.113
Gemma 0.3B	0.303	0.539	0.566	0.376
Qwen3 0.6B	0.224	0.487	0.526	0.297
Qwen3 8B	0.250	0.553	0.618	0.332
KaLM-Embedding-V2.5	0.197	0.474	0.513	0.272
Multilingual-E5-Large-Instruct	0.197	0.421	0.553	0.283
w/ First Section				
zbMath-Bert 0.3B	0.092	0.184	0.237	0.122
Gemma 0.3B	0.237	0.500	0.566	0.321
Qwen3 0.6B	0.263	0.526	0.645	0.352
Qwen3 8B	0.211	0.605	0.645	0.343
KaLM-Embedding-V2.5	0.184	0.447	0.474	0.272
Multilingual-E5-Large-Instruct	0.224	0.474	0.539	0.304

6.5 Search tool

The search tool is currently accessible via HuggingFace Spaces at huggingface.co/spaces/uw-math-ai/theorem-search. Users can perform semantic searches using natural language (e.g.,

Figure 6.3: PCA visualizations of theorems in our dataset using slogan embeddings. Left: Gemma embedding. Right: Qwen3 8B embedding.

"a rational variety is simply connected"). The sidebar interface allows results to be filtered by relevant metadata (e.g. author sets, arXiv category tags, and journal publication status). To optimize retrieval time, the search uses a two-stage retrieval architecture. At inference time, the natural language query is encoded using the Qwen3-8B embedding model and binary-quantized into 4096 bits. An initial pool of clamp(max(200, $12k$), 200, 800) candidates is returned using the HNSW index and Hamming distance, where k is a user-specified number of results to return. The candidate pool is then filtered by metadata and reranked using cosine similarity on the original 4096-dimensional embeddings. When weighting by citations is enabled, the candidates are instead re-ranked using the following composite scoring function:

$$\text{score} = \text{cosine similarity} + \lambda \cdot \log\big(\max(\text{citations}, 1)\big),$$

where λ is the user-specified citation-weight parameter. The interface retrieves the top k results, each of which displays the theorem slogan, the theorem body rendered in LaTeX, any

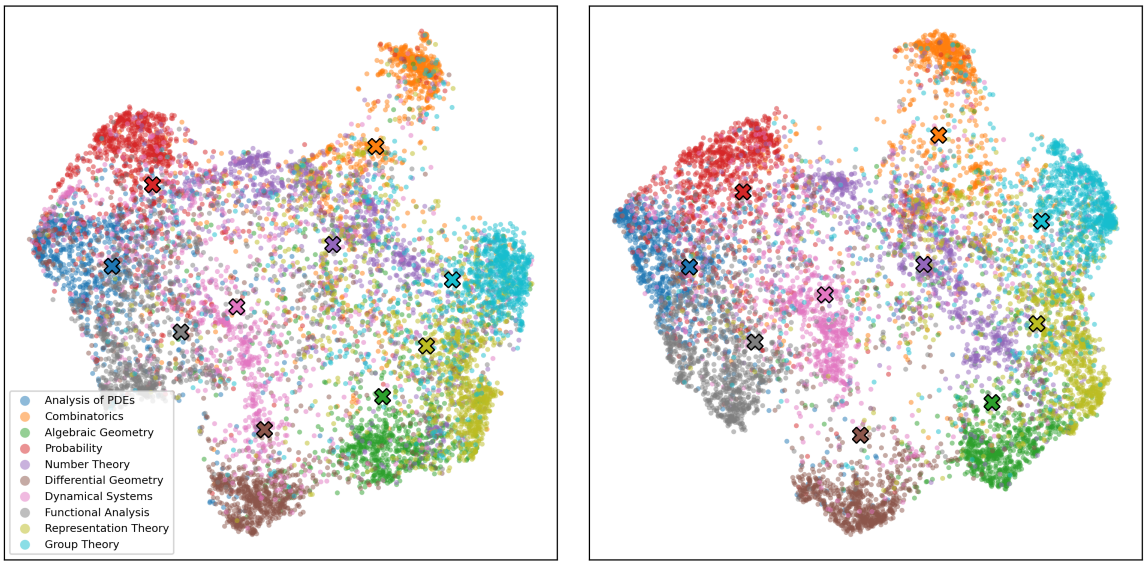


Figure 6.4: UMAP visualizations of 10,000 theorem-slogan embeddings across the ten most common arXiv categories. Gemma 0.3B (left) and Qwen3 8B (right). Qwen3 8B produces tighter, better-separated clusters.

relevant paper metadata, and a link to the paper. Finally, a lightweight feedback system allows users to rate each result's relevance to their query. The resulting end-to-end latency is approximately 3 seconds per query, compared to 60+ seconds for LLM-based search tools such as ChatGPT and Gemini, which must perform multiple web searches and parse documents at inference time. See Figure 6.5 for the interface.

Because our slogans are multi-sentence descriptions, the embedding model produces vectors that are optimized for longer inputs. Short keyword-style queries (e.g. "Landau equation") can therefore be dominated by padding or generic context, leading to poor retrieval. We find that simply repeating a short query (e.g. "Landau equation Landau equation") substantially improves result quality by strengthening the signal of the key terms in the embedding. The search tool applies this duplication automatically when the query is below a length threshold.

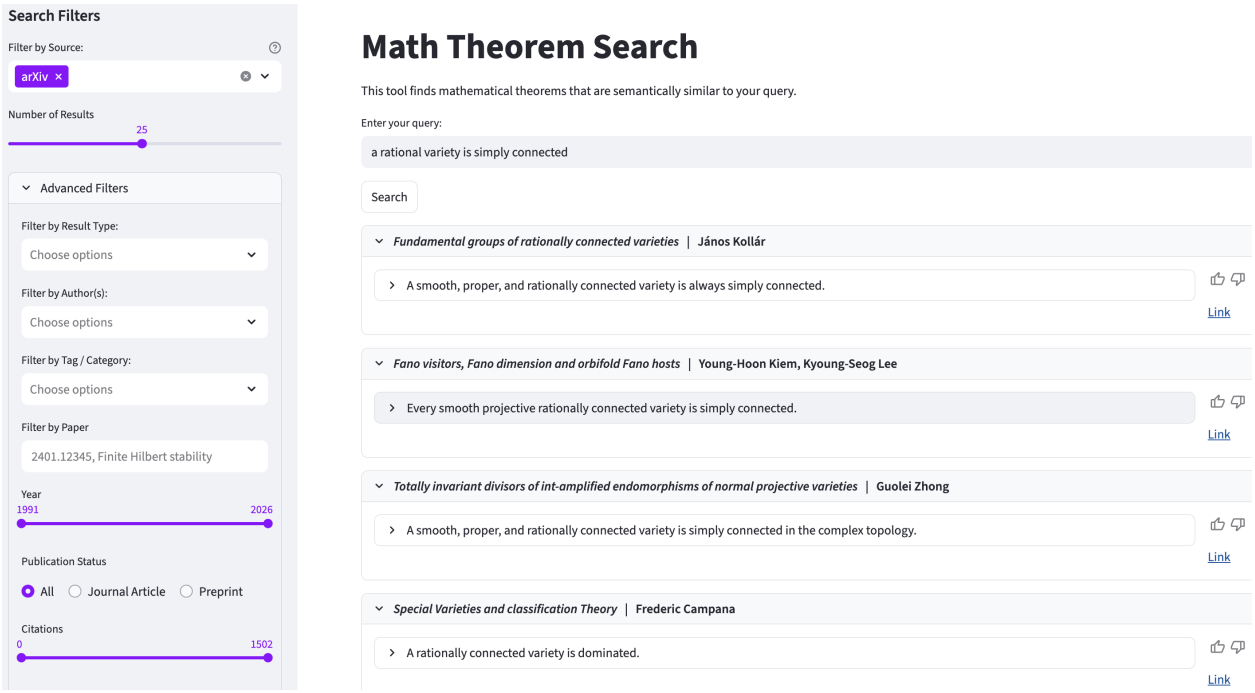


Figure 6.5: The interface of the Theorem Search Tool. The main window displays the top results: paper name, paper authors, theorem statement, and link to the paper. In the left pane users can filter by paper metadata: type of result (theorem/proposition/lemma/corollary), author(s), arXiv tag, specific paper, year, and publication status. The thumbs up/down allow for user feedback, used to further improve the search results.

In addition to the web interface, we provide a REST API at `api.theoremsearch.com/search` that accepts JSON queries and returns ranked theorem results, enabling programmatic integration into research workflows. We also expose a Model Context Protocol (MCP) server at `api.theoremsearch.com/mcp`, allowing AI agents and LLM-based tools to use theorem search as a retrieval tool during reasoning and proof generation. The project page, including links to the search tool, API, MCP server, and dataset, is available at `theoremsearch.com`.

6.6 Conclusion

Mathematical knowledge is organized around discrete results – theorems, lemmas, propositions, and corollaries – yet existing search tools operate at the document level, forcing researchers and AI agents alike to manually locate specific statements within papers. In this chapter, we addressed this gap by constructing a corpus of over 9.2 million theorem statements extracted from arXiv and seven other sources, constituting the largest unified collection of human-authored, research-level theorems to date.

We showed that representing theorems via natural-language slogans generated by an LLM, rather than embedding raw LaTeX directly, yields substantially better retrieval performance. Our systematic ablations revealed that context matters: slogans generated with access to the paper's introduction outperform those generated from the theorem body alone, and the choice of both the slogan-generating LLM and the embedding model significantly affects downstream retrieval quality. On a curated evaluation set of 111 queries written by professional mathematicians, our best configuration achieves 45.0% Hit@20 at the theorem level and 56.8% Hit@20 at the paper level, outperforming ChatGPT 5.2 with search, Gemini 3 Pro, and Google Search.

By treating theorems as first-class retrieval objects and enriching them with natural-language descriptions, we enable new forms of access to mathematical knowledge – whether for retrieval-augmented generation by LLMs, for premise selection in formal proof search, or for literature review by expert mathematicians. We have deployed a public search interface described in Section 6.5. Qualitative user feedback from research mathematicians, reported in Section 6.7, further illustrates the practical utility of our system.

6.7 User feedback

In this section, we present feedback from two users on our theorem search engine.

6.7.1 Feedback one

We will first include a high-level description of the first feedback.

The user wanted to check for a specific reference, which we will call "Theorem 1", which the user knew was true. They thought it follows from another result, which we call "result X," a bit stronger than Theorem 1. It did, after a small lemma; in other words, (result X + small lemma) implies what the user wanted. This shifted the user's attention from finding a reference for Theorem 1 to finding a reference for the small lemma. The user searched for the small lemma in our tool, and the first result was a reference for a generalization of the small lemma.

A more detailed description now follows. We advise the non-mathematician reader to skip the next paragraph.

The user wanted to check that a smooth variety over a separably closed field has a k-point. The user thought it follows from the étale local structure of smooth morphisms, which captured the correct idea but reduced the problem to showing that k-points are dense in $\mathbb{A}^n_k$. In essence, the étale local structure theorem translated the original problem of checking that a smooth variety over a separably closed field has a k-point, to checking that for k separably closed, the set of k-points of $\mathbb{A}^n$ is dense. The user entered the query "if k is separably closed, the set of k-points of $\mathbb{A}^n$ is dense in $\mathbb{A}^n$." The first result was correct: Tag 056U in the Stacks Project.

6.7.2 Feedback two

We now describe a second use case. As before, we first provide a high-level summary.

The user searched for a result known to experts but lacked a specific citable reference; we call this result "Theorem X." Many existing references point to similar results, but not exactly to Theorem X. The user queried our tool and found a result, which we call "Theorem Y," whose citation list directly referenced Theorem X. By using our tool, the user located the desired theorem with fewer queries than traditional search methods, which had previously

led only to indirect references.

A more detailed description follows. We advise the non-mathematician reader to skip the next two paragraphs.

The user needed a structure theorem for abelian p-Lie algebras, as defined by N. Jacobson. The user conjectured that these algebras are classified analogously to finite abelian groups, provided the base field is algebraically closed. On MathOverflow, an answer directed the user to a book that states the desired result. The book explained that the result follows from the structure theory for modules over a PID, without providing a precise proof, and attributes the technique to Jacobson.

The book *Infinite-Dimensional Lie Superalgebras* [13] contains the result the user was looking for, labeled the 3.2 nil-radical Theorem, attributed to Seligman, 1967. However, the user skimmed through the hundred or so pages of Seligman's reference several times without finding where the desired result is proved. When the user queried our tool with "over an algebraically closed field, any abelian p-Lie algebra splits as a torus and a sum of nilpotent cyclic Lie algebras," the first result was the desired statement: Theorem 2.7 in [180].

6.8 Claude experiment

We now report the experiment coupling Claude with our theorem search tool as a RAG system. This experiment tests whether retrieval-augmented generation over our theorem corpus can correct LLM reasoning failures on a research-level mathematical question. For discussion of this experiment in context, see Section 6.4.4.

Query to Claude

Consider the closure, in the KSBA moduli space, of the moduli space of pairs $(S, aB + bF)$ where S is an elliptic surface, B is a section of the projection $S \to C$ which gives the elliptic surface structure to S, the fibers of $S \to C$ are smooth or nodal, F are some fibers of $S \to C$, and $0 < a, b < 1$. The coefficients are chosen so that $(S, aB + bF)$ is KSBA-stable (so don't need to reprove it). I would like to know if

there exist a surface pair $f : (X', aB' + bF') \to C'$ appearing on the boundary of the KSBA-compactification, not admitting a fibration over a nodal curve C', with fibers of f that are of dimension one.

Without access to our tool, Claude answered **incorrectly**:

Claude's response without RAG (incorrect)

No. Every surface pair appearing on the boundary of this KSBA compactification admits a fibration over a (possibly nodal) curve with one-dimensional fibers. The fibration structure cannot be lost in the limit.

Why the fibration persists. The essential point is that the elliptic fibration structure is rigidly encoded in the boundary divisor, and this encoding survives in the KSBA limit.

The fiber class remains nontrivial. In the interior, F is a fiber of $S \to C$, so $F^2 = 0$ and $(K_S + aB + bF) \cdot F > 0$ (since $K_S \cdot F = 0$, $B \cdot F = 1$, so this equals $a > 0$). In any flat limit, the limiting divisor F' retains $F'^2 = 0$ and $(K_{X'} + aB' + bF') \cdot F' > 0$. So F' is a nontrivial nef class with zero self-intersection, which defines a fibration $f : X' \to C'$.

Stable reduction preserves the fibered structure. Given a one-parameter degeneration $(\mathcal{S}_t, \Delta_t) \to \mathcal{C}_t$, after finite base change and birational modification one extends the fibration to the central fiber. Each component of the limiting surface X' either:

- maps surjectively onto a component of the nodal base curve C' with one-dimensional fibers, or
- maps to a point (a "sprouted" component).

But a component mapping to a point would be an entire surface sitting over a single point of C', and the slc and ampleness conditions prevent this from occurring in the

stable model: such a component would need $K + \Delta$ to be positive on it, yet its gluing locus with the rest of X' is along a single fiber curve, and the combinatorics of the slc condition force it to be absorbed (contracted) when passing to the ample model of $K_{X'} + \Delta'$.

The section prevents collapse. The section B' (limit of the section B) meets each fiber in a point, ensuring the map $X' \to C'$ has generically one-dimensional fibers throughout. If any component lost the fibration structure, B' restricted to that component would be a curve not meeting the fiber class transversally, contradicting the deformation-invariant intersection number $B \cdot F = 1$.

Summary. The divisor data $(aB+bF)$ is sufficiently rigid that all KSBA-stable limits retain a fibration $f : X' \to C'$ over a nodal curve C', with all fibers of f being one-dimensional (curves). The boundary pairs differ from interior pairs in that C' may be nodal, X' may be reducible/non-normal (slc), and fibers may be more degenerate—but the fibration itself always exists.

When given access to our theorem database, Claude arrived at the **correct** answer:

Claude's response with RAG (correct)

Yes. Such boundary pairs exist; they are pseudoelliptic surfaces.

- By Corollary 1.3 of [10], the log canonical model of an irreducible slc elliptic surface $(X \to C, S + F_{\mathcal{A}})$ with $K_X + S + F_{\mathcal{A}}$ big is either the relative log canonical model (an elliptic surface over C), or a *pseudoelliptic surface* obtained by contracting the section S.

- By Corollary 6.10 of [10], the section is contracted if and only if $(C, f_*F_{\mathcal{A}})$ is not a weighted stable pointed curve; concretely: $C \cong \mathbb{P}^1$ with $\sum a_i \leq 2$, or C has genus 1 with all $a_i = 0$.

- A pseudoelliptic surface admits no morphism to a curve with one-dimensional fibers: the section (which was isomorphic to the base) has been contracted to a point, so all former fibers pass through a single point and cannot be separated by a map to a curve.

- By Proposition 3.22 of [11], an explicit family is provided: for an elliptic surface over $\mathbb{P}^1$ with $\deg \mathcal{L} = 2$ and $\mathcal{A} > 0$, the log canonical model is a pseudoelliptic surface (section contracted), which is KSBA-stable and lies on the boundary.

- By Corollary 6.7 of [94], in a stable degeneration of elliptic surface pairs, the special fiber maps to a nodal curve C', but pseudoelliptic components map to a *point* of C', so f does not have all fibers of dimension one.

- Properness of the moduli space of slc fibered surface pairs (Corollary 5.11 of [9]) guarantees these pseudoelliptic limits lie in the closure.

Table 6.7: Extended results comparing LLM slogans in retrieval performance across six embedders. Embedded without task instructions. Body+Abstract.

Model	Hit@1	Hit@10	Hit@20	MRR@20
DeepSeek V3.1				
zbMath-Bert 0.3B	0.092	0.184	0.237	0.113
Gemma 0.3B	0.303	0.539	0.566	0.376
Qwen3 0.6B	0.224	0.487	0.526	0.297
Qwen3 8B	0.250	0.553	0.618	0.332
KaLM-Embedding-V2.5	0.197	0.474	0.513	0.272
Multilingual-E5-Large-Instruct	0.197	0.421	0.553	0.283
DeepSeek R1				
zbMath-Bert 0.3B	0.000	0.079	0.105	0.014
Gemma 0.3B	0.158	0.421	0.474	0.228
Qwen3 0.6B	0.132	0.316	0.408	0.199
Qwen3 8B	0.118	0.355	0.461	0.195
KaLM-Embedding-V2.5	0.066	0.289	0.461	0.140
Multilingual-E5-Large-Instruct	0.066	0.250	0.355	0.111
Gemini 3 Pro				
zbMath-Bert 0.3B	0.053	0.171	0.237	0.092
Gemma 0.3B	0.250	0.553	0.632	0.348
Qwen3 0.6B	0.211	0.487	0.539	0.290
Qwen3 8B	0.237	0.566	0.684	0.348
KaLM-Embedding-V2.5	0.197	0.434	0.553	0.265
Multilingual-E5-Large-Instruct	0.197	0.513	0.579	0.297
Claude Opus 4.5				
zbMath-Bert 0.3B	0.053	0.184	0.211	0.087
Gemma 0.3B	0.303	0.553	0.579	0.387
Qwen3 0.6B	0.224	0.539	0.605	0.325
Qwen3 8B	0.263	0.632	0.737	0.394
KaLM-Embedding-V2.5	0.197	0.421	0.526	0.271
Multilingual-E5-Large-Instruct	0.224	0.526	0.592	0.322

Chapter 7

LEARNING TO REPAIR LEAN PROOFS FROM COMPILER FEEDBACK

This chapter is the second of Part III, which traces the discovery → repair → formalization arc of AI-assisted mathematics. Chapter 6 addressed *discovery* – how to find a specific theorem in millions of statements; this chapter addresses *repair* – how to fix a Lean proof that fails to compile. In real interactive proof development, mathematicians (and AI agents) routinely write partial proofs, watch the compiler complain, and edit incrementally until verification succeeds. Yet the public training data for neural theorem provers is the opposite: only the final compiling proofs are preserved, and intermediate failures are discarded. We argue that this asymmetry leaves a strong supervised signal on the table.

We construct *APRIL* (Automated Proof Repair in Lean), a dataset of 260,000 tuples in which each tuple pairs a deliberately broken Lean proof, the compiler's error message and local goal state, a verified repair, and a natural-language diagnosis grounded in the same feedback. Failures are produced by four controlled mutation strategies (theorem substitution, tactic substitution, single-line redaction, multi-line redaction), so every example is anchored to a real compiler trace in a fixed Lean 4 environment. Supervised fine-tuning on APRIL substantially improves repair accuracy: a 4B-parameter Qwen3 model goes from 1.1% to 27.4% single-shot repair, slightly exceeding Goedel-Prover-V2-32B under the same no-search protocol.

This chapter is based on [188].

7.1 Introduction

Formal theorem proving offers a strict setting for studying machine learning: proofs must be written in a formal language and verified by a proof assistant such as Lean, leaving no ambiguity of correctness. This property allows learning systems to receive exact verification signals and has motivated a surge of recent work on learning-based automated theorem proving. Large Language Models (LLMs) have achieved substantial gains on formal benchmarks and even reached Olympiad-level performance [113, 36, 3].

Despite these advancements, the dominant objective in current systems remains end-to-end proof generation: given a formal statement, the goal is to produce a complete proof that verifies successfully. Training data therefore consists primarily of correct proofs. Evaluation is likewise based on whether a valid proof is eventually found. Failures encountered during generation are typically used to guide exploration or policy optimization, but are not treated as supervised learning targets. As such, models receive limited supervision for interpreting diagnostics, explaining what went wrong, or proposing targeted edits. Recent work has also shown that iterative proof refinement using compiler feedback is 32-128x more efficient than pass@k [36].

In contrast, human proof development is inherently iterative and error-driven. Proof engineers routinely write partial proofs, inspect compiler diagnostics, and incrementally revise the code until verification succeeds. Intermediate failures are fully expected in the development process and the same feedback is used both to decide what to change and to communicate why the change is correct. However, most publicly available proof corpora preserve only final proofs and omit failed attempts. Large-scale libraries such as mathlib provide extensive collections of formally verified theorems, but do not record intermediate error states encountered during development [173]. Datasets derived from natural language autoformalization pipelines, including Lean Workbook and Herald, provide aligned natural–formal pairs and synthetic proof data, but likewise consist almost entirely of valid formal artifacts rather than trajectories of failed attempts and subsequent repairs [206, 68]. As a

result, models trained on these resources have little direct exposure to feedback-conditioned iteration, including both producing corrected code and articulating diagnoses grounded in compiler messages.

This work studies proof correction from compiler feedback, and introduces a dataset that supports two aligned feedback-conditioned tasks: (1) producing a corrected proof, and (2) producing a natural-language diagnosis and fix suggestion grounded in the same feedback. These targets share the same underlying evidence (error message and local proof state), but emphasize different outputs: formal repair versus human-interpretable debugging. To enable controlled analysis, we construct datasets by systematically mutating correct proofs to generate plausible failures produced by controlled mutations, and we evaluate language models under supervised finetuning without reinforcement learning or inference-time search, isolating the model's ability to interpret diagnostics and apply targeted repairs.

Our contributions are threefold. First, we introduce a large-scale dataset of 260K Lean proof-repair examples, each consisting of an erroneous proof, compiler diagnostics including error messages and local proof state, and a corresponding corrected proof. Second, we develop a systematic mutation pipeline that generates realistic failures by substituting semantically related theorems, swapping similar tactics, and eliciting plausible but incorrect line and multi-line completions from language models. Third, we demonstrate that supervised finetuning on this data substantially improves repair accuracy under our evaluation setting: 27.4% correction accuracy in our single-shot repair evaluation (no search/iteration), compared to 1.1% for the base model and 26.8% for Goedel-Prover-V2-32B under the same protocol. Because Goedel is typically deployed with search/iteration, this comparison should be interpreted as an ablation of compiler-feedback-conditioned repair rather than end-to-end proving performance.

7.2 Related work

Neural Theorem Proving in Lean. Recent learning-based systems for Lean primarily target end-to-end proof generation, combining large language models with search, reinforce-

ment learning, and synthetic data pipelines [197, 149, 113, 189]. Infrastructure such as LeanDojo has enabled systematic benchmarking and retrieval-augmented proving [202], while Lean 4 provides the formal foundation for this work [51]. Evaluation across these systems is framed in terms of theorem-level success metrics such as pass@k on benchmarks like miniF2F [214].

Verifier Feedback and Proof Repair. Verifier feedback and errors play an important role in existing pipelines, but primarily as control signals for exploration and policy optimization rather than as supervised correction targets [113, 197, 3, 105, 198, 149]. More recent refinement-based systems incorporate compiler diagnostics into prompts to regenerate proofs across multiple iterations, demonstrating that iterative repair can be substantially more efficient than independent sampling [36, 215]. Our work studies proof correction from compiler feedback as a supervised learning problem, providing aligned repair and diagnosis targets grounded in the same compiler messages and local proof state.

Proof Datasets and Training Corpora. Large-scale libraries such as mathlib provide extensive collections of formally verified theorems, but do not record intermediate error states encountered during development [173]. Datasets derived from natural language autoformalization pipelines, including Lean Workbook, Herald, and NuminaMath, provide aligned natural–formal pairs and synthetic proof data, but likewise consist almost entirely of valid formal artifacts rather than trajectories of failed attempts and subsequent repairs [206, 68, 110]. The scarcity of error trajectories in public corpora remains a key bottleneck for training models that can interpret and act on compiler diagnostics.

Program Repair from Diagnostic Feedback. In software engineering, program repair has been widely studied as a supervised or self-supervised learning problem over paired erroneous and corrected programs, including approaches that condition on compiler error messages to guide targeted edits [76, 203, 16]. Several lines of work also generate repair

supervision at scale when paired error–fix data is scarce, using compilers or analyzers to validate synthetic errors and candidate fixes [204, 193]. Our work applies a similar philosophy to the formal verification setting: we systematically mutate correct Lean proofs to generate plausible failures, then train models for feedback-conditioned repair.

7.3 Methodology

7.3.1 Data collection

Each entry in APRIL consists of (i) an erroneous Lean proof that fails to compile, (ii) a corresponding fixed proof that is syntactically similar but compiles successfully, (iii) the compiler error messages and proof state, (iv) a natural-language explanation of the errors, and (v) a natural-language fix suggestion describing how to transform the erroneous proof into the corrected one.

Because large-scale corpora of human-generated Lean errors are scarce while correct proofs are widely available, we construct APRIL backward: beginning with a correct proof and systematically introducing errors into it. Although our failures are induced by controlled mutations, every example is anchored in the expected elaboration behavior: the error message, offending location, and local goal state are produced by the Lean compiler in a fixed environment, and the repair target is a verified proof in that same environment. We start from verified correct proofs sourced from the Herald, Lean Workbook, and Numina datasets, retaining only those that compile under Lean 4.22.0-rc4. We then introduce errors using the mutation strategies described below and keep only mutated proofs that trigger compiler errors. To retrieve compiler feedback and compilation rates, we use Lean-Interact to interact with the Lean REPL [143]. For each retained example, we use DeepSeek-V3-0324 [115] to generate an explanation of the compiler feedback and a corresponding fix suggestion.

Theorem Mutation Errors Theorem mutation is designed to imitate errors that arise from subtle semantic mismatches, where a theorem is almost applicable but differs in its required premises or conclusion. To construct such mutations, we begin with proofs that

compile successfully and extract the theorems used in each proof. Each resulting (proof, theorem) pair is treated as a separate mutation candidate. For each selected theorem occurrence, we retrieve semantically related declarations using the LeanExplore semantic search engine [12], filter out trivial replacements (including the original theorem and namespace-only variants), and perform a single substitution by replacing exactly one occurrence of the selected theorem identifier in the proof text. The remaining candidates typically differ in their hypotheses or conclusion in ways that are small but proof-critical, and the resulting error may surface later in the proof, far from the mutation site.

Tactic Mutation Errors Tactic mutations substitute a similar, yet incorrect tactic, e.g., swapping `nlinarith` with `linarith`, `norm_num`, or `ring`. We define a fixed set of tactic equivalence classes based on common proof roles, including arithmetic solvers, rewriting tactics (e.g., `rw`, `simp`, `simp_rw`), structural tactics (e.g., `intro`, `intros`, `rintro`), and proof-construction tactics (e.g., `apply`, `refine`, `exact`, `assumption`). Substitutions are only performed within the same class to ensure syntactic validity and plausibility. For each proof, we randomly select between one and three swappable tactic occurrences and substitute each with a similar alternative; we keep only unique mutated proofs that fail to compile.

Line Mutation Errors To produce a line mutation error, we begin with the corrected proof and at random replace one proof line (after the main `by`) with `REDACTED`, preserving indentation. This redacted proof is passed to DeepSeek-V3-0324 [115] with instructions to provide the redacted line. The model-generated completion is reinserted and compiled with Lean; only mutated proofs that fail to compile are retained and duplicates are removed.

Multi-Line Mutation Errors Multi-line mutations are produced similarly to line mutations, except we redact the entirety of the proof after the randomly selected line and allow the model to produce as many lines as it pleases to replace the redacted lines. To ensure the erroneous proof resembles the correct proof, no more than half of the proof is redacted.

Explanation and Fix Generation All mutated proofs are compiled with Lean, and only those that produce a compiler error are retained, since some changes still lead to correct proofs. For each retained theorem mutation, we also store contrastive metadata, including the names and formal statements of both the intended (correct) theorem and the substituted (incorrect) theorem. For each retained erroneous proof, we automatically generate a natural language explanation of the failure and a suggested fix using a separate language model, prompted with the original proof, the mutated proof, the Lean compiler error message, and any available mutation metadata.

7.3.2 Model finetuning

We finetune Qwen3-4B-Instruct-2507 [201], Kimina-Prover-Distill-8B, and Goedel-Prover-V2-8B using an identical supervised finetuning (SFT) pipeline with Low-Rank Adaptation (LoRA) [80]. All training procedures and hyperparameters are shared across all the experiments.

Training Format and Objective Each training example is converted into a chat-formatted prompt using the corresponding model tokenizer. The input consists of a system message and a user message concatenating the prover error, local proof state, and failing proof. The supervised target is the assistant completion, structured to encourage an explicit diagnostic reasoning step prior to code generation. The data format is as illustrated in Figure 7.3.

LoRA Setup We apply LoRA adapters with rank 32 to both attention and MLP projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`) in each transformer block. This configuration is identical across all base models.

Optimization and Training Hyperparameters We use AdamW [123] with learning rate 1×10^{-4}, cosine decay, and linear warmup. Training runs for up to 15,000 steps with effective batch size 8 and maximum sequence length 2048. Early stopping with patience 5

(i.e., 1,250 steps) terminates training if validation loss does not improve, and we select the checkpoint with lowest validation loss. Full hyperparameter details are provided in Table 7.1.

Compute and Implementation Details All experiments are conducted on NVIDIA L40S and H200 GPUs using bfloat16 precision. We enable FlashAttention-2 [49] when available, otherwise falling back to SDPA.

Table 7.1: Optimization and training hyperparameters used in all experiments.

Hyperparameter	**Value**
Evaluation metric	`eval_loss`
Effective batch size	8
Optimizer	AdamW
Learning rate	1×10^{-4}
LR scheduler	Cosine decay
Warmup ratio	0.1
Weight decay	0.01
Max gradient norm	1.0
Sequence length	2048
Sequence packing	Disabled
Precision	bfloat16 (bf16)
Evaluation frequency	Every 250 steps
Checkpoint frequency	Every 250 steps
Checkpoint selection	Best validation loss
Rank (r)	32
Scaling factor (α)	64
Dropout	0.1
Bias	None
Target modules	`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`

7.4 APRIL

7.4.1 Dataset composition

Table 7.2: Statistics of correct proofs after filtering. We report the number of raw and filtered proofs for three source datasets, along with the average proof length, average number of `have` statements per proof, and number of proofs containing at least one `have` statement. Across all datasets, we observe significant variation in proof length and `have` statement usage, with proofs sourced from human annotators in Numina's dataset demonstrating the most complexity in terms of length and `have` statements.

Dataset	Raw	Filtered	Avg. Lines	Avg. `have`	Contain `have`
Herald	30,190	16,010	4.02	0.13	1,500
Lean Workbook	10,433	9,491	2.71	0.16	820
Numina Autoformalizer	6,039	5,925	10.20	2.17	2,839
Numina Human	9,428	8,066	50.93	8.90	6,128
Total	56,090	39,492	14.21	2.24	11,287

Correct Proof Sources. To construct paired correct and incorrect proofs that remain structurally aligned, we begin from large public corpora of Lean proofs and inject controlled errors into initially correct proofs. We collect correct proofs from the Herald, Lean Workbook, and NuminaMath-Lean datasets, and retain only samples that successfully compile under Lean 4.22.0-rc4.

In total, APRIL contains 39,492 unique compiled theorems with diverse proof styles and complexity (Table 7.2). Approximately 40.5% of proofs originate from Herald, consisting of modular fragments extracted from intermediate states of human-written mathlib4 proofs. Another 24.0% come from Lean Workbook and are short, low-complexity machine-generated proofs. The remaining proofs are drawn from NuminaMath-Lean: 15.0% are produced by

an autoformalizer and 20.4% are annotated by human experts. These proofs exhibit longer average length and more frequent use of `have` statements, indicating higher structural complexity.

Generated Incorrect Proofs. From the 39,492 compiled theorems, we generate 260,125 incorrect proofs using four mutation operators (Figure 7.4): (i) theorem substitution, (ii) tactic replacement, (iii) line-level redaction or modification, and (iv) multi-line redaction or modification. Each mutation yields a syntactically valid Lean file that fails to compile and produces a concrete compiler error trace.

Each dataset instance contains:

- the original correct proof,
- a mutated incorrect proof,
- the Lean compiler error message and local goal state at the failure location,
- a repaired proof target, and
- a natural-language explanation and fix suggestion aligned with the compiler feedback.

Theorem substitution errors constitute the largest fraction of incorrect proofs (59.5%), reflecting the prevalence of type- and goal-mismatch failures in real proof development.

7.4.2 Dataset split

To prevent data leakage, we split the dataset at the level of original theorems rather than individual mutated proofs. All mutated variants derived from the same original theorem are assigned to the same split. This ensures that the model cannot observe identical correct proofs across training and evaluation. We additionally anonymize all theorem declarations by renaming them to a canonical identifier (`lean_problem`), preventing models from exploiting cross-dataset name correlations or memorizing problem-specific identifiers.

We perform stratified splitting based on (i) source dataset (Herald, Lean Workbook, NuminaMath-Lean) and (ii) proof length, preserving the distribution of proof complexity and domains across splits. The proportions of each mutation type are also maintained in each split (Table 7.3). Table 7.4 reports basic per-split statistics on the underlying correct proofs, and Table 7.5 reports the distribution of source datasets across splits.

Table 7.3: Number of erroneous proofs by split.

Error Type	Train	Val	Test	Total
Full	249,027	9,263	1,835	260,125
Tactic	59,688	2,064	398	62,150
Line	17,098	1,021	200	18,319
Theorem	148,362	5,415	1,093	154,870
Multi-Line	23,879	763	144	24,786

Table 7.4: Basic dataset statistics by split.

Split	Num	Avg. Lines	Avg. `have`	Contain `have`
Train	38,292	14.22	2.25	10,966 (28.64%)
Val	1,000	13.74	1.85	267 (26.70%)
Test	200	14.29	1.71	54 (27.00%)
Total	39,492	14.21	2.24	11,287 (28.58%)

7.5 Results

We evaluate proof repair accuracy on a held-out test set of 1,835 erroneous proofs spanning all four mutation types. A repair is considered successful if the model's output compiles under

Table 7.5: Statement source distribution by split.

Split	Lean Workbook	Herald	Numina Auto	Numina Human
Train	9,159 (23.92%)	15,471 (40.40%)	5,833 (15.23%)	7,829 (20.45%)
Val	276 (27.60%)	450 (45.00%)	77 (7.70%)	197 (19.70%)
Test	56 (28.00%)	89 (44.50%)	15 (7.50%)	40 (20.00%)
Total	9,491 (24.03%)	16,010 (40.54%)	5,925 (15.00%)	8,066 (20.42%)

Lean 4.22.0-rc4. We compare against the base Qwen3-4B-Instruct model (no finetuning) and Goedel-Prover (8B/32B), using the same single-shot repair interface (no search/iteration).

7.5.1 Main results

Table 7.6 reports proof repair accuracy across models and error types. Finetuning on APRIL yields large gains across all models. For Qwen3-4B-Instruct, finetuning increases repair accuracy from 1.1% to 27.4% (25×). Under the same protocol, this slightly exceeds Goedel-Prover-V2-32B (26.8%). Finetuned 8B models reach 31–35% repair accuracy, outperforming the 32B Goedel baseline.

All results are single-shot repairs from a failing attempt, not theorem-level success under multi-sample search or exploration.

7.5.2 Analysis by error type

Table 7.6 further breaks down repair performance by mutation category, revealing systematic variation in difficulty across error types while preserving consistent trends across models. Tactic mutations yield the highest repair rates, with top performance reaching 42.5%, reflecting the relatively local nature of these errors within a fixed proof context. Theorem mutations exhibit intermediate difficulty, while line mutations are the most challenging, with maximum accuracy of 13.5%, as they often involve correcting semantically inconsistent or hallucinated

Table 7.6: Proof repair accuracy by model and error type when training jointly for repair and explanation. Results outside the parentheses correspond to models jointly finetuned on all error types and evaluated in each category. Parenthesized values denote models finetuned separately on a single error type and evaluated on that same type. Section 7.5.3 analyzes the effect of removing explanations.

Baseline Model	Full	Tactic	Line	Theorem	Multi-Line
Goedel-Prover-V2-8B	15.5%	19.6%	20.0%	12.7%	19.4%
Goedel-Prover-V2-32B	26.8%	34.2%	28.5%	23.0%	32.6%
Kimina-Prover-1.7B	8.4%	15.1%	13.5%	4.8%	10.4%
Kimina-Prover-8B	11.1%	17.3%	14.5%	7.9%	13.9%
Qwen3-4B-Instruct-2507	1.1%	1.8%	2.5%	0.5%	0.0%
Finetuned Goedel-8B	34.6%	41.7%	18.5%	36.8%	20.8%
Finetuned Kimina-8B	31.9%	38.9%	18.5%	34.1%	14.6%
Finetuned Qwen3-4B	27.4%	39.7%	16.0%	26.8%	13.2%

proof steps.

Importantly, models trained jointly on the full dataset maintain strong performance when evaluated on individual error categories. For example, Qwen3-4B-Instruct fine-tuned on the full dataset achieves competitive accuracy across tactic, theorem, and multi-line errors without explicit specialization. Comparable behavior is observed for Goedel-Prover-8B and Kimina-Prover-8B, where training on the full dataset results in only modest differences relative to models trained separately on each error type.

The limited degradation observed under joint training indicates that the dataset's mutation types share substantial structural overlap. Rather than inducing negative interference, joint training enables models to learn repair strategies that generalize across error categories. This property is particularly important for realistic proof repair settings, where error types are heterogeneous and not known a priority.

7.5.3 Effect of explanations

We ablate the effect of jointly supervising natural-language explanations alongside proof repair. Specializing exclusively on repair increases pass@1 from 27.4% to 31.2%. This reveals a controllable trade-off: training exclusively for repair maximizes autonomous pass@1, while joint supervision yields human-interpretable diagnoses that can support human-in-the-loop debugging or downstream tool-using agents.

The explanations that finetuned Qwen produce are valuable in their own right, however. Model explainability is often hard to come by, and this is especially true for error correction in Lean. Top models (Goedel and Kimina) are specialized in error correction to the point that they have lost much of their ability to produce meaningful natural language, including to explain their work, making their reasoning very difficult to understand. Training on our dataset allows Qwen to improve significantly at correcting Lean errors and at the cost of an only slight reduction in error correction accuracy it can also produce explanations of its reasoning. As a small demonstration of the downstream utility of explanations, we provide DeepSeek with the same failing instance augmented by an explanation, and observe substantially higher success rates when using explanations produced by the explanation-trained model. DeepSeek succeeds with a rate of 4% when aided by base Qwen's explanations and 29% when aided by the trained model's explanations, performing better than either model individually.

7.6 Conclusion

We introduced Lean proof repair as a supervised learning task: given a failing proof and compiler feedback, predict a corrected proof that compiles. To support this, we constructed APRIL, a dataset of paired erroneous and verified proofs augmented with compiler diagnostics and additional natural-language annotations. Supervised finetuning on this data substantially improves repair accuracy, with a finetuned 4B model outperforming larger open-source provers under our evaluation setup. These results suggest that error-centric su-

Table 7.7: Pass@1 on error types matching the training regime. Δ denotes the change from w/o exp (color-coded).

Model	Method	w/o exp	w/ exp	Δ
Goedel-8B	Full	36.7%	34.6%	-2.1%
	Tactic	48.5%	46.5%	-2.0%
	Line	25.5%	21.5%	-4.0%
	Theorem	37.5%	37.0%	-0.5%
	Multi-Line	24.3%	22.9%	-1.4%
Kimina-8B	Full	36.9%	31.9%	-5.0%
	Tactic	49.7%	47.2%	-2.5%
	Line	23.5%	23.5%	0.0%
	Theorem	37.4%	37.0%	-0.4%
	Multi-Line	26.4%	21.5%	-4.9%

pervision is a strong and underutilized training signal for building agentic theorem provers that can iteratively refine proofs from feedback.

The APRIL dataset is available on HuggingFace under `uw-math-ai/APRIL`, with fine-tuned models with and without explanations available under `uw-math-ai/gAPRIL-w-exp` and `uw-math-ai/gAPRIL-wo-exp` respectively.

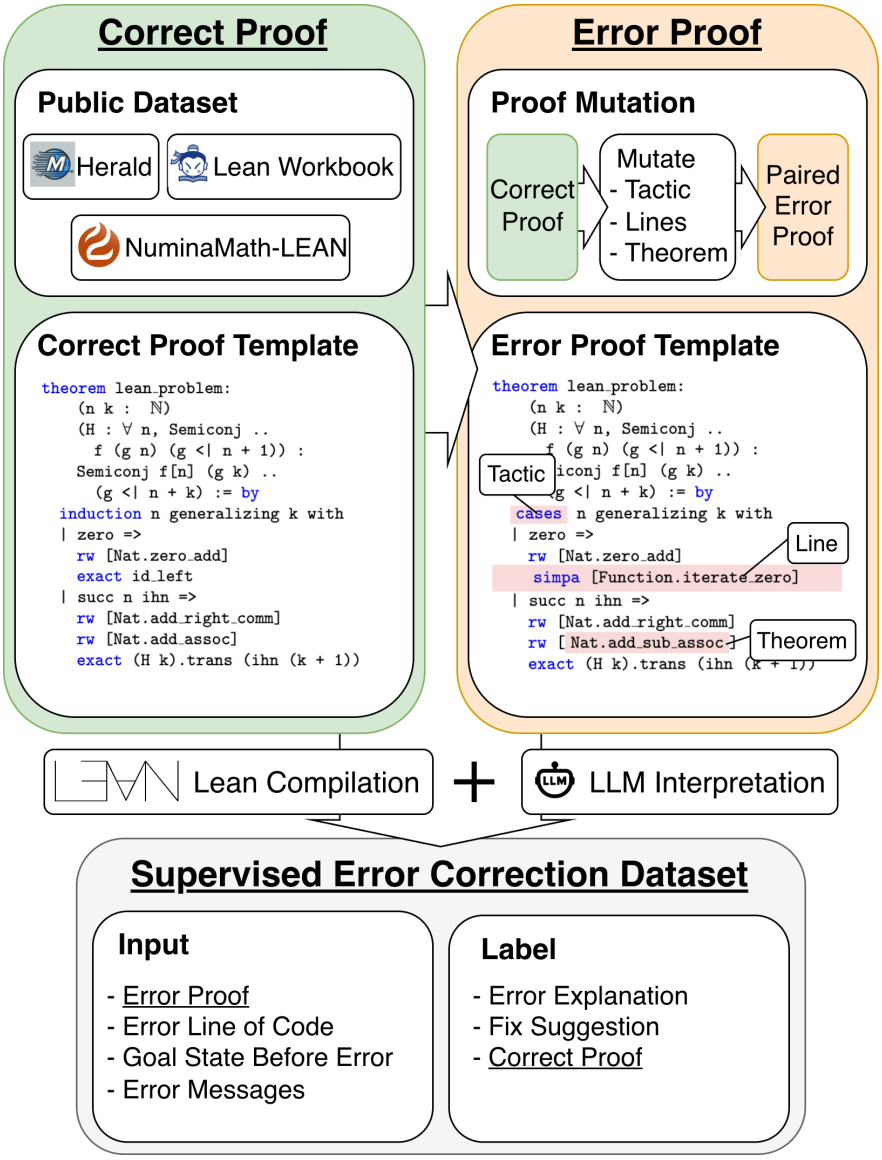
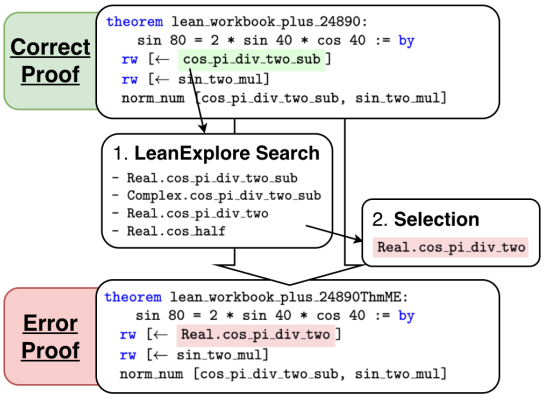
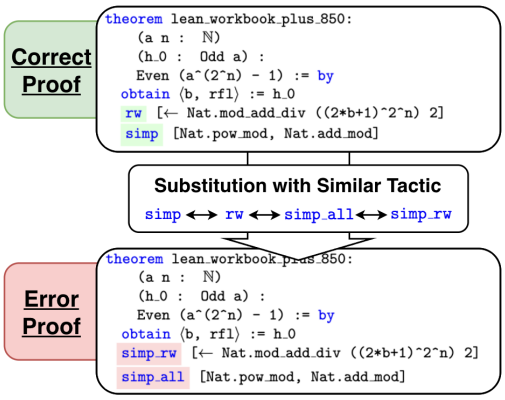


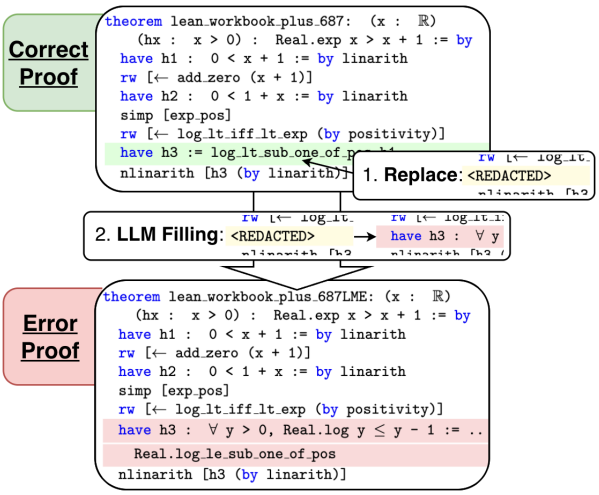
Figure 7.1: Overview of our dataset collection pipeline that collects paired correct and incorrect proof and their contextual information. We collected correct proofs from public datasets (Herald, Lean Workbook, NuminaMath-Lean). The paired error proofs that maintain the same proof sketch are generated by mutating tactics, lines of code, or theorems. Then, we used the Lean compiler to filter out the error proofs and extract their error messages, error lines, and goal states. Finally, we prompted an LLM to generate error explanations and fix suggestions based on the paired proofs and Lean Infoview. The resulting dataset pairs erroneous proofs and error information with structured labels containing the error interpretation from LLM and the corresponding correct proofs.



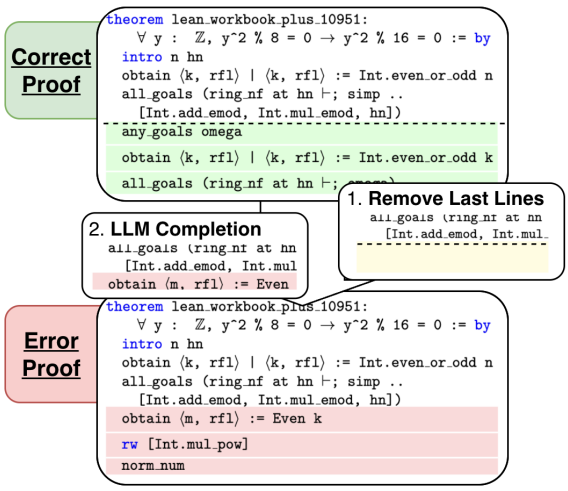
(a) Theorem mutation error



(b) Tactic mutation error



(c) Line mutation error



(d) Multi-line mutation error

Figure 7.2: Examples of the four mutation error types with illustrations of their generation models.

Prompt
[System] You are diagnosing a single failing proof **[User]** Explain the error, suggest a fix, and provide the corrected proof based on the context: Incorrect Proof: `<Proof with Generated Errors>` State: `<Goal State Before the Error from InfoView>` Line at error: `<Error-Occurred Line of Code>` Lean error: `<Error Messages from InfoView>`
Completion
[Assistant] Explanation: `<Explanation of Error Cause>` Fix: `<Code Manipulation Fix Suggestion>` Corrected Proof: `<Corrected Proof>`

Figure 7.3: A template of our finetuning data format.

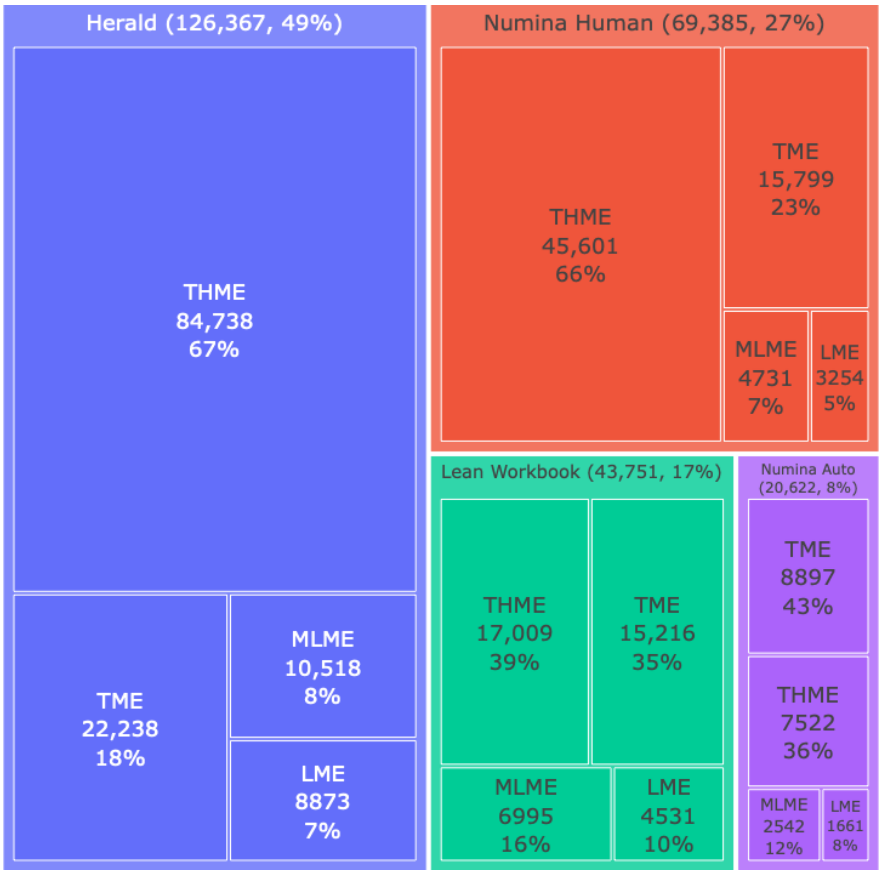


Figure 7.4: Statistics based on mutation type. For each dataset, we report the total number of erroneous proofs generated and detail the distribution of specific mutation types: Line Mutation Error (LME), Multi-line Mutation Error (MLME), Tactic Mutation Error (TME), and Theorem Mutation Error (THME). The percentages denote the proportion of each mutation type within its respective dataset.

Chapter 8

SEMI-AUTONOMOUS FORMALIZATION OF THE VLASOV–MAXWELL–LANDAU EQUILIBRIUM

This chapter closes Part III, which traces the discovery → repair → formalization arc of AI-assisted mathematics. Chapter 6 addressed *discovery* – how to find a specific theorem in millions of statements – and Chapter 7 addressed *repair* – how to learn from compiler feedback to fix proofs that fail to compile. This chapter addresses *formalization* itself: assembling discovery, proof generation, repair, and verification into a single end-to-end workflow on a research-level theorem.

The theorem is the one already proved on paper in Section 3.3: every sufficiently smooth steady state of the full Vlasov–Maxwell–Landau system on the torus, with Coulomb collisions, must be a global Maxwellian with vanishing electric field and constant magnetic field. This is the analytical anchor of Part I, originally *conjectured* from the numerical evidence in Chapter 2 before being proved on paper, and now *formalized* in Lean 4 with zero `sorry`'s. The formalization was completed in 10 days at a cost of $200 by a single mathematician who wrote zero lines of code – the proof was generated by an AI reasoning model (Gemini DeepThink), translated into Lean by an agentic coding tool (Claude Code), and closed lemma-by-lemma by a cloud-based prover (Aristotle). The Lean kernel provides the final guarantee of correctness that none of these models can offer alone.

The chapter focuses on the formalization *process* rather than the mathematics: the tools used (Section 8.3), the six-phase timeline (Section 8.4), the project statistics (Section 8.5), and the lessons learned about the capabilities and failure modes of current AI tools for formalization (Section 8.6), including a definition-alignment bug caught by an expert reviewer post-completion. The mathematical proof itself is given in Section 3.3 and is not reproduced

here. This chapter is based on [88].

Project	Prover	LOC	New thm	New defs	Team	AI role	Logs	Domain
Gauss [137]	Lean 4	181K‡	✗	✗	6+§	sorry elim.	✗	Geometry
Bayer & David [15]	Isabelle	20K	✓	✓	15	none	✗	Num. theory
This work	Lean 4	**10K**	✓	✓	**1**	**all code**	✓	**Math. phys.**
Numina [117]	Lean 4	∼8K	✓	✓	13	collab.	✗	Analysis
AxiomProver [33]	Lean 4	4.3K	✓	∼	21	fully auton.	✗	Num. theory
AxiomProver [34]	Lean 4	2.7K	✓	∼	21	fully auton.	✗	Algebra
2HDM [177]	Lean 4	1.6K	✗†	✓	1	minimal	✗	Physics
Aristotle [166]	Lean 4	1.4K	✓	✗	1	fully auton.	✗	Combin.
AxiomProver [31]	Lean 4	0.2K	✓	∼	5	fully auton.	✗	Alg. geom.

Table 8.1: 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.

8.1 Introduction

Formal verification of mathematics in proof assistants like Lean 4 [50, 173] 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 [179]. This bottleneck has limited formal verification to a

small community of specialists.

Recent advances in AI-assisted theorem proving are changing this landscape rapidly (Figure 8.1). Large language models can now generate Lean tactics [36], agentic coding tools can manage entire Lean projects [117], and cloud-based automated theorem provers can close individual lemmas without human intervention [3].

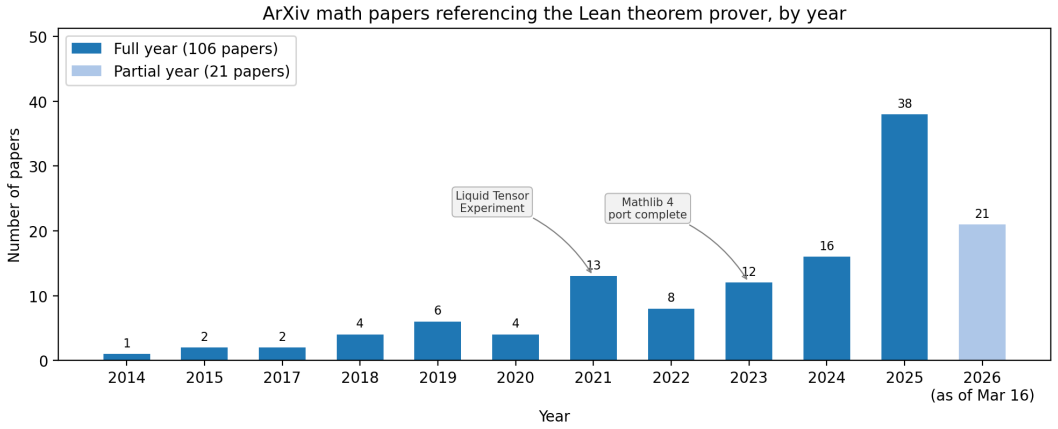


Figure 8.1: ArXiv papers using Lean for formalization, by year. Data from the arXiv metadata snapshot (March 2026), filtered for references to the Lean theorem prover.

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 [34] from a natural-language problem statement alone, GPT-5.2 generating informal proofs of open Erdős problems that were then auto-formalized in Lean by Aristotle [166], and Google DeepMind's Aletheia agent [63, 62] autonomously producing research papers in natural language. [117] formalized effective Brascamp–Lieb inequalities (∼8K lines) with human experts collaborating on the Lean code, and [15] 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 8.1):

- **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 subscription + $0 for Aristotle and Gemini).

8.2 The Theorem

8.2.1 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:

$$v \cdot \nabla_x f + (E + v \times B) \cdot \nabla_v f = \nu\, Q(f, f), \tag{8.1}$$

$$\nabla \times B = J = \int_{\mathbb{R}^3} v\, f \, \mathrm{d}v, \tag{8.2}$$

$$\nabla \cdot E = \int_{\mathbb{R}^3} f \, \mathrm{d}v - \rho_{\text{ion}}, \tag{8.3}$$

$$\nabla \cdot B = 0, \tag{8.4}$$

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).

8.2.2 Main Result

Theorem 8.1 (Informal; see [89] for the complete proof)**.** *Let $f > 0$ be a smooth steady-state solution of the VML system with Coulomb collisions on $\mathbb{T}^3 = (\mathbb{R}/\mathbb{Z})^3$, with Schwartz-class velocity decay and a polynomial log-growth bound. Then:*

1. *f is a spatially uniform Maxwellian: $f(x, v) = \frac{\rho_{\text{ion}}}{(2\pi T)^{3/2}} \exp\!\big(-\frac{|v|^2}{2T}\big)$ for some $T > 0$;*

2. the electric field vanishes: $E(x) = 0$*;*

3. the magnetic field is constant: $B(x) = B_0$*.*

The formal LEAN statement is shown in Figure 8.2. It takes 12 hypotheses, all with clear physical meaning: 2 physical parameters ($\nu > 0$, $\rho_{\text{ion}} > 0$), strict positivity of f, smoothness conditions (C^3 in velocity, C^2 in space via periodic lift, C^2 for B), Schwartz-class velocity decay, a polynomial score bound ($|\partial_{v_i} f| \leq C(1 + \|v\|)^K f$), and the four steady-state equations (8.1)–(8.4). The polynomial score bound is a genuine assumption not derivable from Schwartz decay alone; it is satisfied by Maxwellians and physically reasonable perturbations. A 13th hypothesis – a polynomial log-growth bound ($|\log f(x, v)| \leq C(1+\|v\|)^K$) – is derived from the score bound and smoothness inside the proof.

8.2.3 Proof Architecture

The proof follows a classical entropy method in seven steps (Figure 8.3):

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.

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.

8.3 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).

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).

Aristotle (Harmonic). A cloud-based automated theorem prover for LEAN 4 [3]. 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.

8.4 Process

8.4.1 Timeline

The project unfolded over 10 days of active development (March 1 – 10, 2026), preceded by the Gemini DeepThink session on February 28. Table 8.2 summarizes the key milestones.

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 8.2: Project timeline and sorry count evolution.

8.4.2 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 [184, 28, 74, 75, 53], 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.

8.4.3 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).

8.4.4 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 MATHLIB representation `Fin 3 → 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.

8.4.5 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.

8.4.6 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 ∼4,000 lines of analytical estimates.

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.

8.4.7 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.

8.5 Statistics

Table 8.3 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 only $200 for the Claude Max subscription, which provides generous usage.

The development trajectory is shown in Figures 8.4 and 8.5: the sorry count follows a sawtooth pattern as scaffolding and proving alternate, while the LOC history shows the four-phase structure of the project. Figure 8.11 reveals that roughly half of the final codebase

is supporting infrastructure – definitions, the torus instance, Coulomb kernel integrability files, and decay helpers – rather than the core proof chain. This reflects the substantial gap between MATHLIB's current coverage and the background material needed for research-level mathematical physics: the Landau operator, torus differential calculus, Schwartz decay machinery, and Coulomb singularity estimates all had to be built from scratch. Figure 8.6 shows session activity across two machines, with near-round-the-clock coverage during the final sprint. Figure 8.7 breaks down token consumption, revealing the 254:1 input-to-output ratio characteristic of formal verification. Figure 8.8 summarizes tool usage across 17,334 invocations, and Figure 9 shows the outcomes of 220 Aristotle submissions.

8.6 Lessons Learned

8.6.1 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.

8.6.2 Aristotle is Transformative

The cloud-based theorem prover Aristotle [3] 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 8.10). 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.

8.6.3 Definition Alignment Requires Expert Review

After the formalization was announced on the Lean Zulip, Sébastien Gouëzel (a Mathlib maintainer) immediately identified a definition-alignment bug: the original theorem used `ContDiff` $\mathbb{R}$ $\top$, which in MATHLIB means *analytic*, not C^∞ as intended. Neither the AI agent nor the mathematician caught this during 10 days of development. Gouëzel remarked: *"I find it a little bit worrisome that AI can overlook such discrepancies – if there is one in the only line I've checked, how many others are there?"*

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. Gouëzel 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.

8.6.4 Automation Helps but is Not Strictly Necessary

The `/babysit` loop (Section 8.4) 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.

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.

8.6.5 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 8.4) 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.

8.6.6 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.

8.6.7 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.

8.6.8 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.

8.7 Related Work

Table 8.1 compares recent research-level formalizations with publicly available code.

AI-assisted theorem proving. Seed-Prover [36] achieves state-of-the-art results on competition mathematics using reinforcement learning with Lean feedback. Harmonic's Aristotle [3] achieved gold-medal performance on the 2025 IMO and auto-formalized GPT-5.2-generated proofs of open Erdős problems in Lean [166]. Google DeepMind's Aletheia agent [63, 62] autonomously produced research papers in natural language, including solutions to four open Erdős conjectures. AxiomProver autonomously generated formal Lean proofs of Fel's open conjecture [34], a result on partial regularity of primes [33], and parity of k-differentials [31] – all from natural-language problem statements with zero human guidance. Unlike our work, these systems are specialized for theorem proving, and the formalized results are typically short combinatorial or algebraic arguments rather than large-scale formalizations requiring hard analysis (Coulomb singularity estimates, PDE energy methods).

Agentic formalization. Numina-Lean-Agent [117] combines Claude Code with MCP tools and formalized effective Brascamp–Lieb inequalities (∼8K lines) with two human experts collaborating on the Lean code. Archon [65] uses a dual-agent architecture and demonstrated full automation of FirstProof problems. [15] formalized their own new theorem on Diophantine complexity bounds in 20,000 lines of Isabelle *in parallel* with the mathematical research – an "in-situ formalization" – with 13 student collaborators writing the proof assistant code by hand and no AI assistance. Our work differs in that a single mathematician wrote zero code of any kind: the mathematical proof was generated by Gemini, and all Lean code was generated by Claude Code from natural-language prompts.

Large-scale autoformalization. Math.Inc's Gauss agent [137] eliminated ∼160K lines of sorry's in the sphere packing formalization (Viazovska's Fields Medal-winning result) in three weeks, growing the codebase from ∼22K to 181K lines. The theorem statements and definitions had been written by a team of 6+ human experts; Gauss's role was sorry elimination – filling in proofs where the goal was unambiguous. [179] formalized 130,000 lines of Munkres' topology textbook in two weeks for ∼$100, using ChatGPT/Claude Sonnet with a fast proof checker (Megalodon). Both projects are larger in scale but formalize known results; our work targets a new theorem requiring new definitions.

Formalization in physics. [177] formalized the stability of the two-Higgs-doublet-model potential in Lean/PhysLean (1.6K lines), discovering an error in a widely cited 2006 paper – an example of formalization catching bugs in published proofs. Unlike our work, this formalized existing results with minimal AI assistance.

Transparency. A distinguishing feature of our work is full transparency. Systems like Seed-Prover [36] are closed-source, and even open systems like Numina-Lean-Agent [117] do not publish the complete interaction logs. We publish all 229 human prompts (with timestamps), all 213 commits, the full Gemini dialogue, and the complete development log

of all 122 `/babysit` cycles.

8.8 Limitations

Mathematical novelty. The theorem – that every smooth positive steady state of the full VML system with Coulomb collisions on the torus must be a global Maxwellian – does not appear in the literature as a stated and proved result. Each proof technique is individually classical: the H-theorem and null-space characterization of the Landau operator are standard [184, 27]; the polynomial-matching argument that forces constant temperature and Killing's equation on the bulk velocity appears in [27] and [53]; and the maximum-principle argument for the nonlinear Poisson–Boltzmann equation is textbook. However, prior work assembles these ingredients only for simpler systems. [53] prove steady-state uniqueness for the spatially inhomogeneous Boltzmann equation *without* electromagnetic fields ($E = B = 0$). [74, 75] study the full VML and Vlasov–Poisson–Landau systems but prove *dynamical* stability of small perturbations near an assumed Maxwellian, embedding the relevant macroscopic constraints inside linearized energy estimates rather than stating static uniqueness as a standalone result. What is new is the complete static characterization for the electromagnetically coupled system: the observation that the Lorentz force contributes only $\mathcal{O}(|v|^1)$ terms to the Vlasov equation for a local Maxwellian (because $(v \times B) \cdot v = 0$), so the $\mathcal{O}(|v|^3)$ and $\mathcal{O}(|v|^2)$ constraints of [27] and [53] carry over unchanged; the Ampère-law integration argument that forces the bulk velocity to vanish; and the closure via Gauss's law with a neutralizing background. These steps are individually straightforward, but their assembly into a single self-contained proof for the full VML system appears to be new; the complete proof is given in [89] and reproduced in Section 3.3.

Definition alignment. As discussed in Section 8.6, the definition-alignment bug that survived the entire development process illustrates a broader risk: similar undetected misalignments in other definitions (e.g., `torusCurlX`, `LandauOperator`) cannot be ruled out without a thorough expert audit.

Reproducibility. The process depends on the specific capabilities of Claude Opus 4 (March 2026), Aristotle (March 2026), and Gemini (February 2026). These models are updated frequently; the same process may yield different results with future or past model versions.

Cost. While the subscription cost was $200, the API-equivalent cost was ∼$6,300 (with prompt caching) or ∼$42,700 (without). The gap between the subscription and API cost reflects the Claude Max plan's unlimited usage model. At API pricing without caching, the project would not be as accessible.

Human effort. Although no code was written, the process required significant mathematical expertise and approximately 50 hours of active supervision (Table 8.3). It requires a mathematician who understands the proof well enough to detect when the agent is weakening the theorem. As one Zulip commenter noted, "the human verification process... is critical" and cannot yet be replaced by another LLM.

8.9 Conclusion

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 Deep-Think) 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 cost of $200, 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.

```
theorem CoulombConcreteTheorem42
    (f : Torus3 → (Fin 3 → ℝ) → ℝ)
    (E B : Torus3 → Fin 3 → ℝ)
    (ν ρ_ion : ℝ)
    (hν : 0 < ν)                                              -- (1)
    (hρ_ion : 0 < ρ_ion)                                      -- (2)
    (hf_pos : ∀ x v, 0 < f x v)                               -- (3)
    (hf_smooth_v : ∀ x, ContDiff ℝ 3 (f x))                   -- (4)
    (hf_smooth_x : ∀ v, ContDiff ℝ 2
      (periodicLift (fun x => f x v)))                        -- (5)
    (hB_smooth : ∀ i, ContDiff ℝ 2
      (periodicLift (fun x => B x i)))                        -- (6)
    (hSchwartz : UniformSchwartzDecay f)                      -- (7)
    (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)                        -- (8)
    (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)               -- (9)
    (hAmpere : ∀ x,
      torusCurlX B x = fun i => ∫ v, v i * f x v)             -- (10)
    (hGauss : ∀ x,
      torusDivX E x = (∫ v, f x v) - ρ_ion)                   -- (11)
    (hDivB : ∀ x, torusDivX B x = 0)                          -- (12)
    :
    ∃ (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 8.2: 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 8.2.3. Here `Torus3` is `Fin 3 → AddCircle 1`, `periodicLift` lifts torus functions to $\mathbb{R}^3$ for differentiation, and `equilibriumMaxwellian` is the Maxwellian $\rho_{\text{ion}}/(2\pi T)^{3/2}\exp(-|v|^2/2T)$.

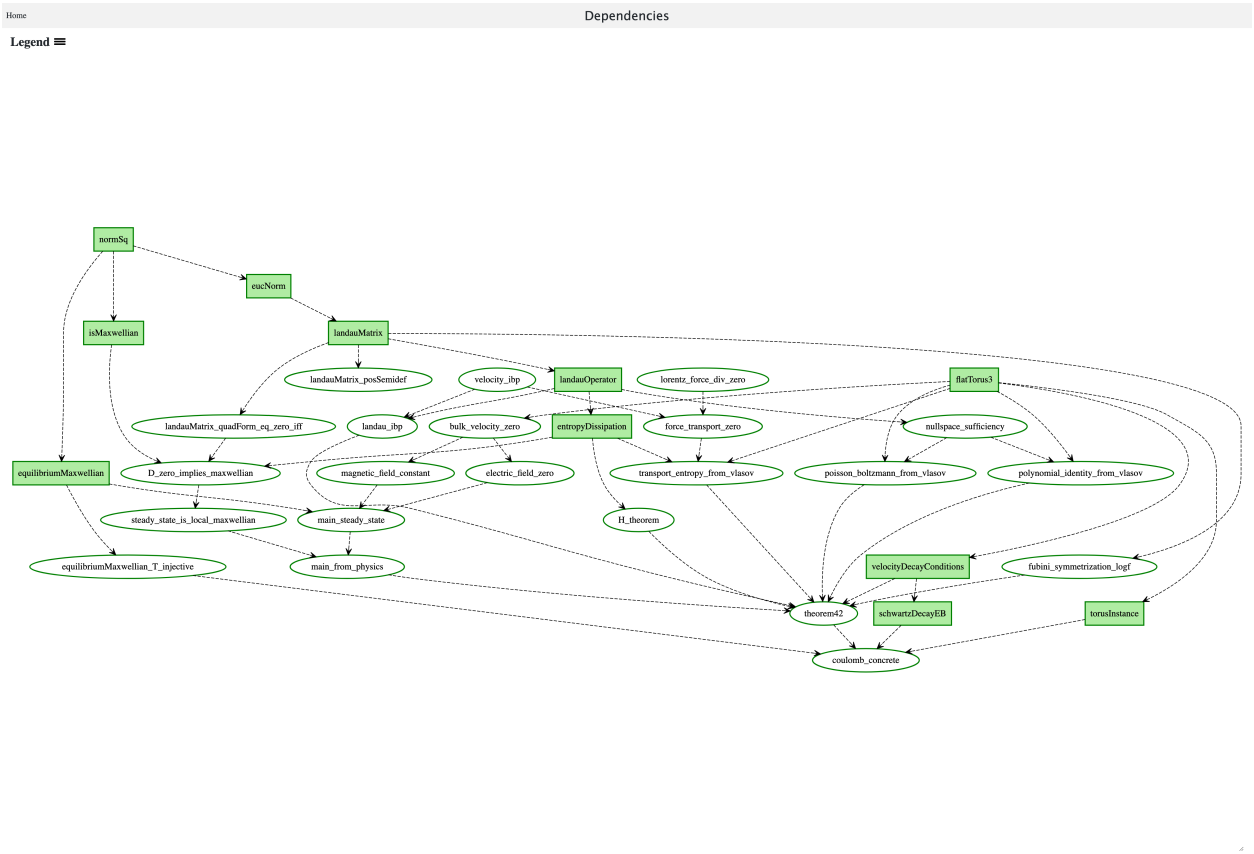


Figure 8.3: Proof dependency graph, generated with LeanBlueprint [135]. All nodes are green (fully proved). The graph flows from primitive definitions (`normSq`, `eucNorm`, `landauMatrix`) at the top through intermediate results (`H_theorem`, `D_zero_implies_maxwellian`, the field and conservation lemmas) down to `theorem42` and its concrete Coulomb instantiation `coulomb_concrete` at the bottom. An interactive version is available in the repository.

Metric	Value
LEAN 4 files	34
Lines of code	10,445
Theorems	39
Lemmas	186
Definitions	28
`sorry`'s	0
Development period	10 days
Git commits	213
Claude Code sessions	32 (across 2 machines)
Human prompts (excl. slash commands)	229
`/babysit` cycles	122
Assistant turns	27,186
Tool calls	17,334
Tokens consumed (input)	2.8 billion
Tokens consumed (output)	11 million
Input-to-output ratio	254:1
Aristotle submissions	220
Proved	111 (50%)
Disproved	28 (13%)
Returned with `sorry`	66 (30%)
Failed	15 (7%)

Table 8.3: Project statistics. The 229 human prompts exclude bare slash commands like `/babysit`; all are archived with timestamps. The mathematician estimates ∼50 hours of active supervision over 10 days.

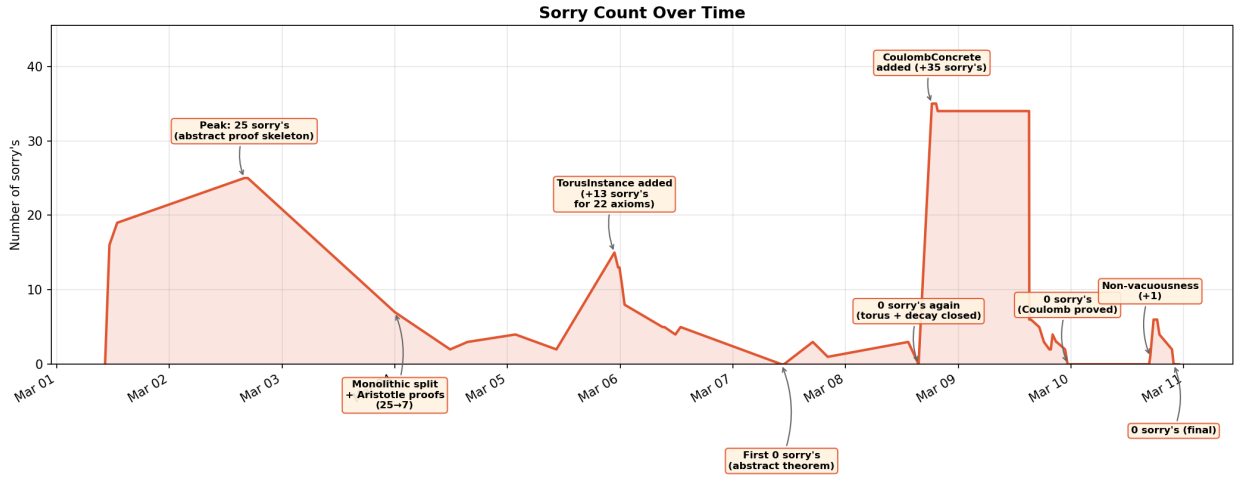


Figure 8.4: Sorry count in `main/` over time (85 commits). The sawtooth pattern reflects alternating phases of scaffolding (sorry's accumulate) and proving campaigns (sorry's eliminated). Key events: peak 25 on Mar 2 (abstract proof chain stated); Aristotle drops count from 25→7 on Mar 3; first 0 sorry's 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).

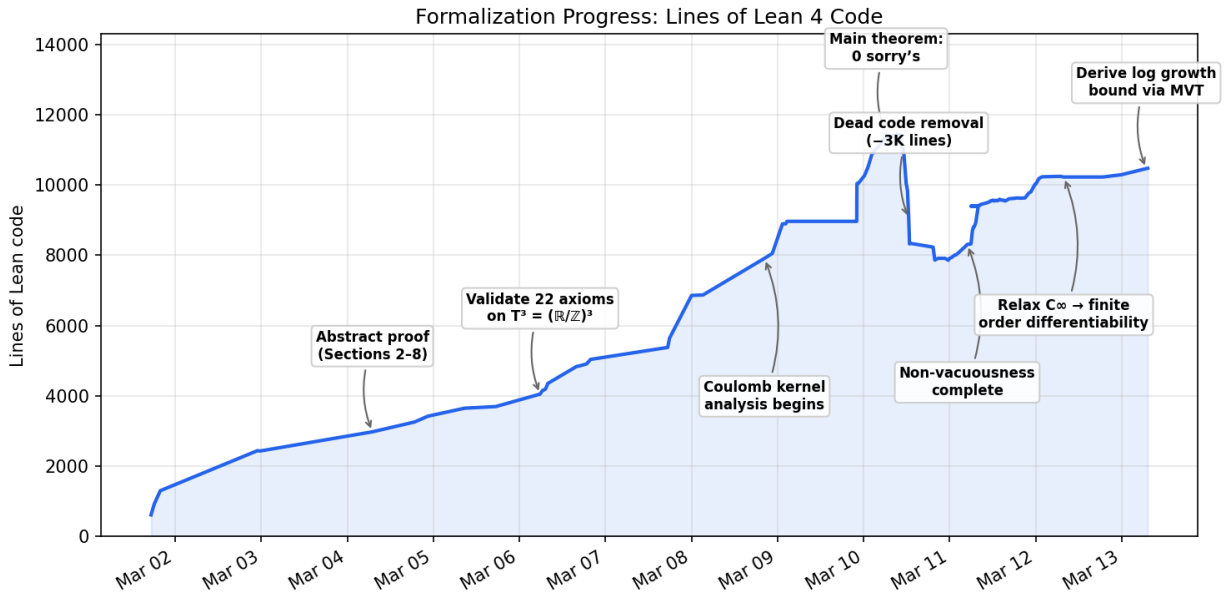


Figure 8.5: LEAN lines of code over time (all 213 commits). The project progressed in four phases: (1) Abstract proof chain (Mar 3 – 6), formalized against an abstract `FlatTorus3` typeclass. (2) Coulomb kernel analysis (Mar 7 – 9), handling the singularity at $r = 0$ – the sharp LOC increase reflects ∼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.

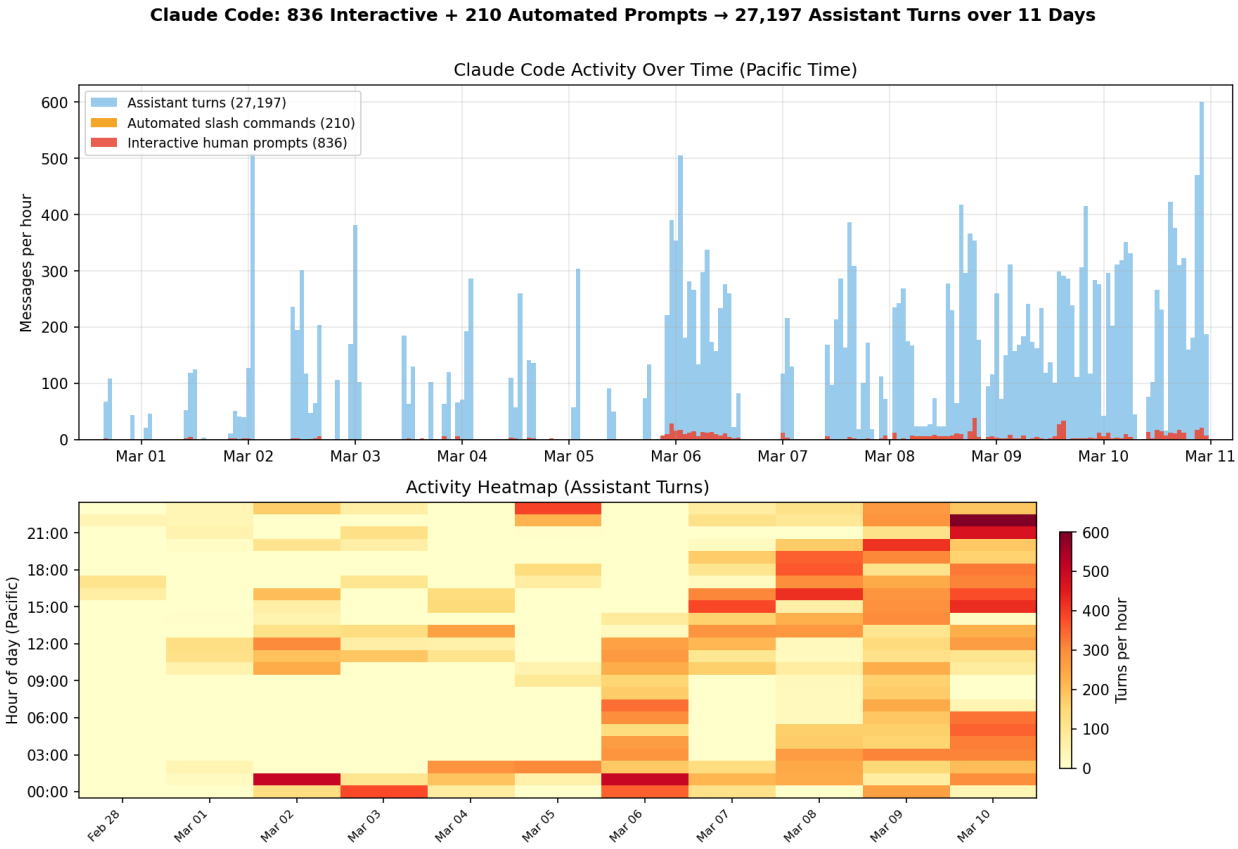


Figure 8.6: Claude Code activity across two machines (local + university server). Top: messages per hour (red = interactive human, orange = automated, blue = assistant). Bottom: hour-of-day heatmap. The 27,186 assistant turns versus 229 human prompts give a 119:1 ratio, reflecting the high autonomy of 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.

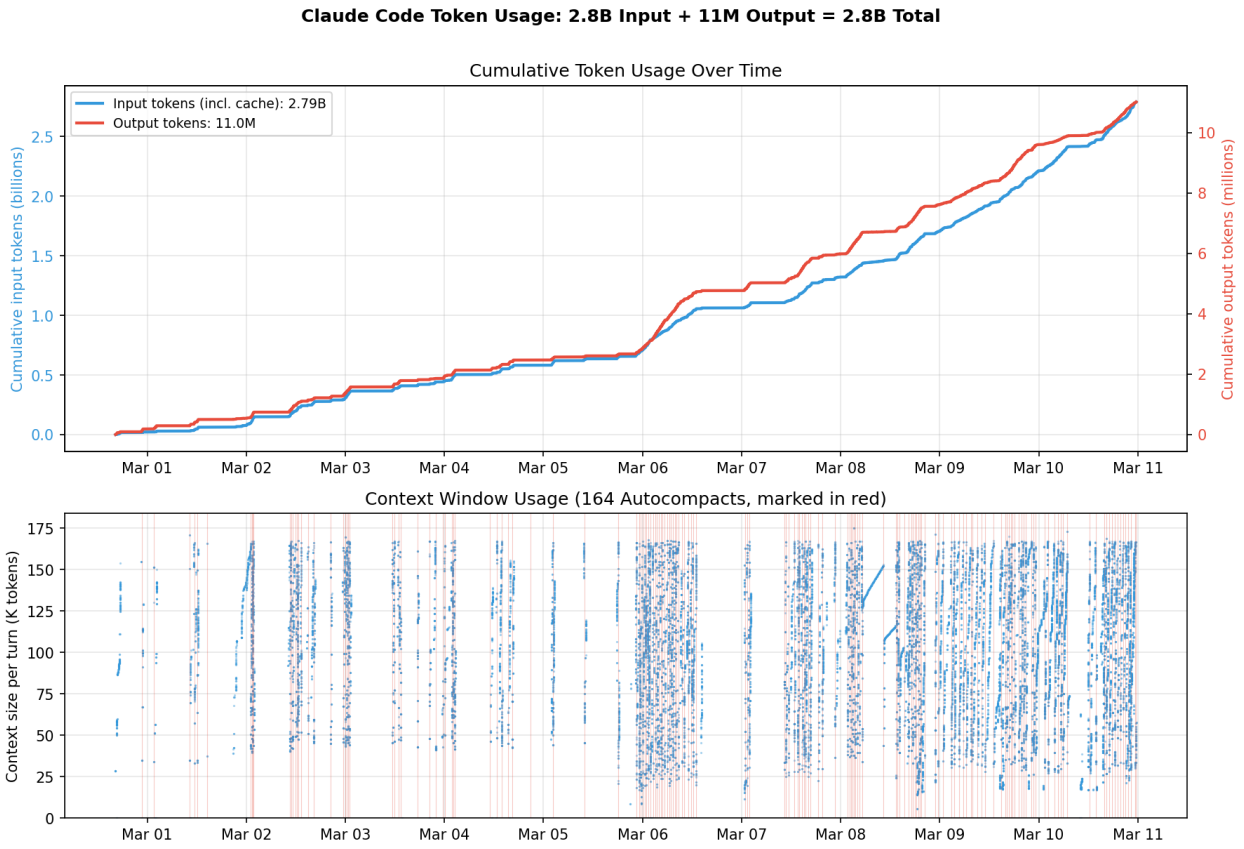


Figure 8.7: Token consumption across the project. Top: cumulative input tokens (blue, 2.8B) and output tokens (red, 11M). The 254:1 input-to-output ratio reflects the read-heavy nature of formal verification – Claude reads far more context than it writes. Bottom: per-turn context size, showing a sawtooth 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.

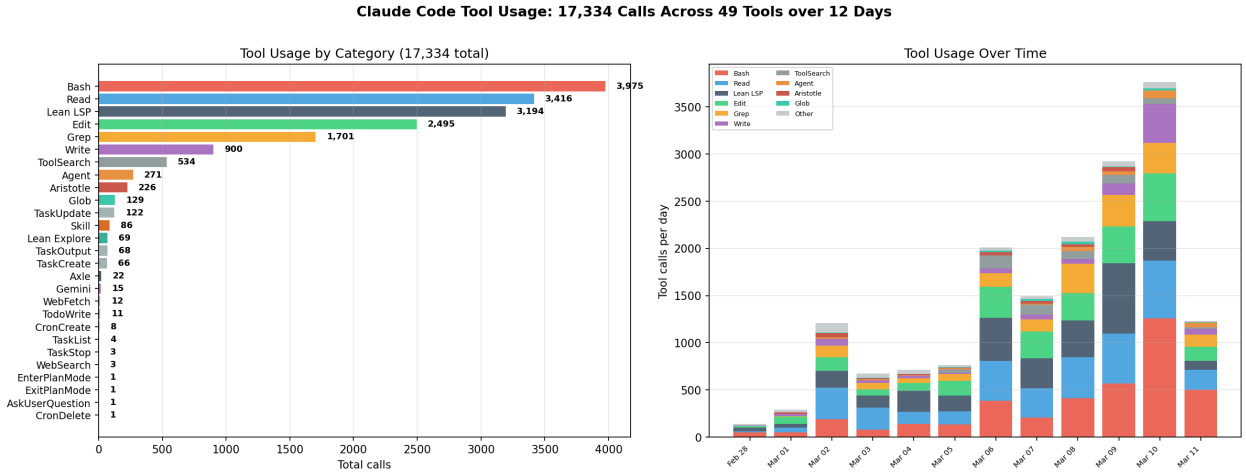


Figure 8.8: Tool usage across 17,334 invocations. Left: totals by category. The core editing loop dominates – Bash (3,975), Read (3,416), Edit (2,495), Grep (1,701), Write (900) – reflecting the iterative nature of formal verification. LEAN LSP tools (3,194 combined) provided real-time feedback. Right: 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.

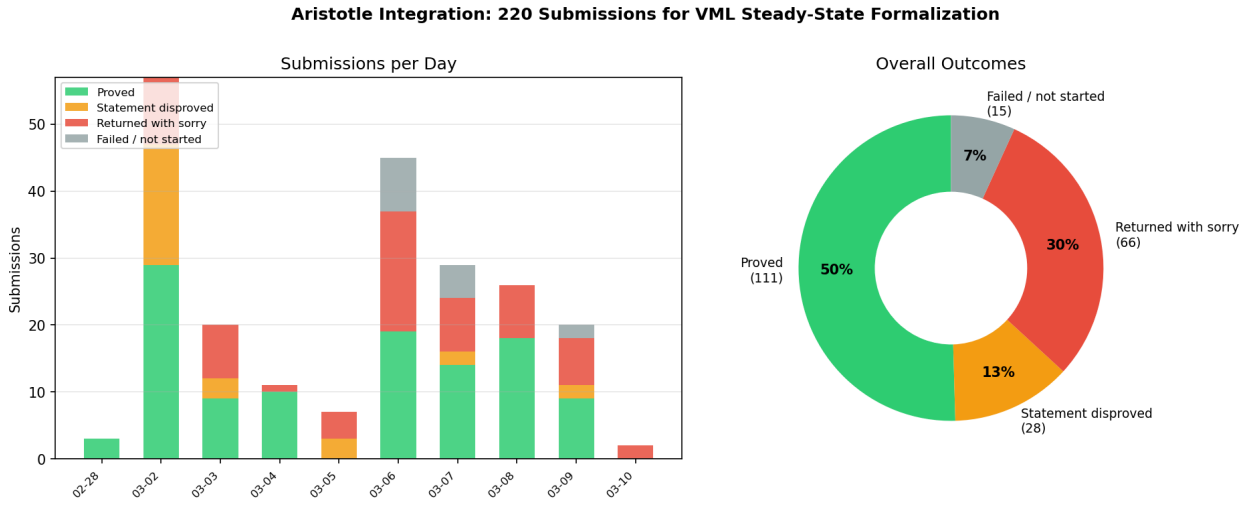


Figure 8.9: Outcomes of 220 Aristotle submissions. Aristotle proved 111 lemmas cleanly (50%), disproved 28 false statements (13%), returned 66 with sorry's still present (30%), and 15 failed or were never started (7%). The disproved statements were particularly valuable – they caught false conjectures 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).

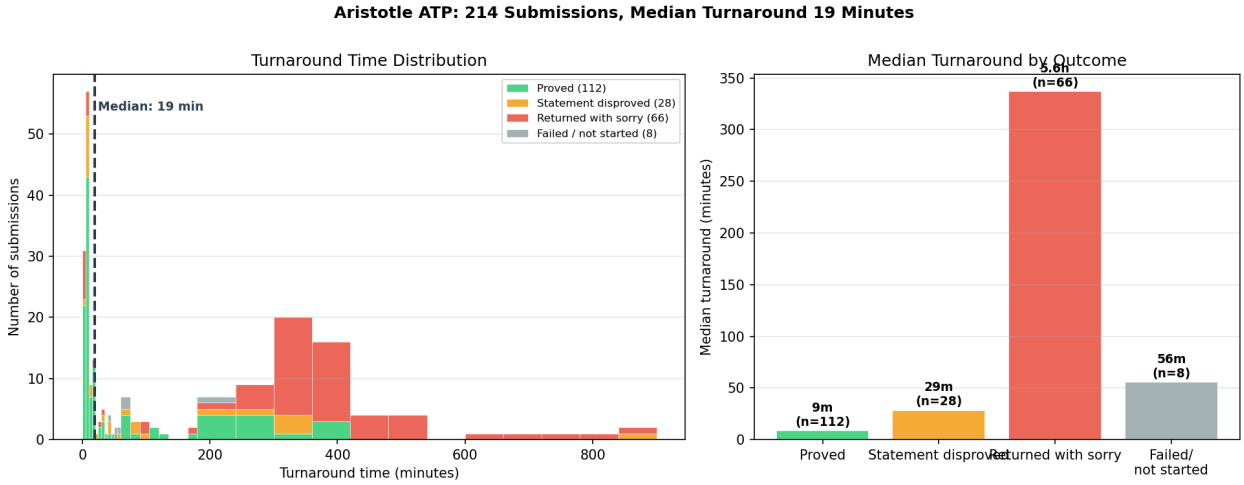


Figure 8.10: Aristotle turnaround times for 214 submissions, colored by outcome. Left: histogram. Proved lemmas (green) cluster in the first bins (median 9 minutes), while "returned with sorry" (red) dominates the long tail. Right: median turnaround by category. The overall median was 19 minutes. Disproved 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.

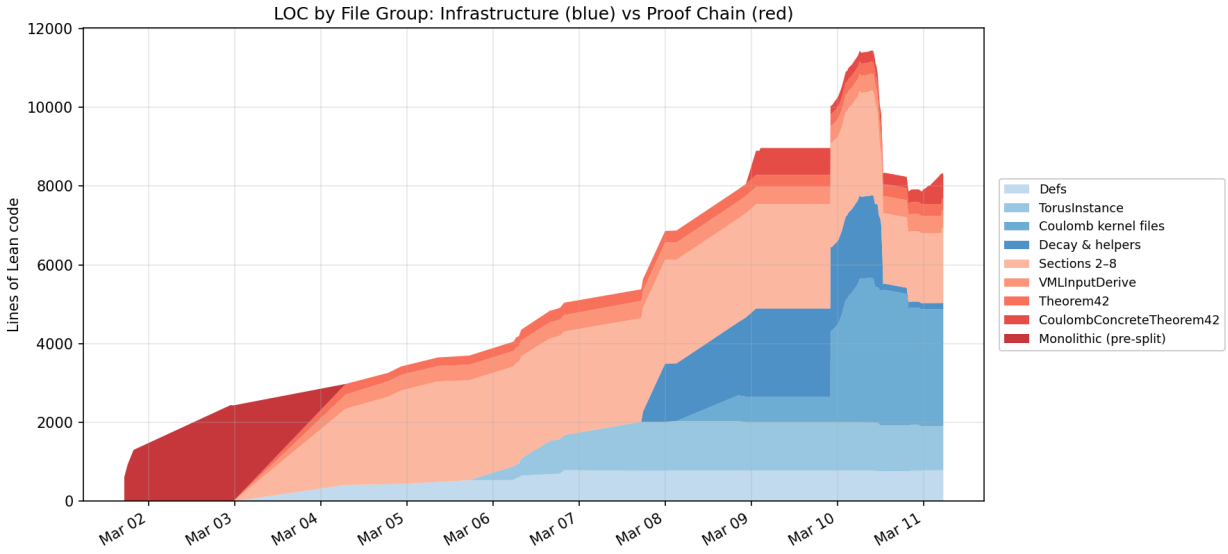


Figure 8.11: LOC by file group over time. Red shades are the mathematical proof chain (Sections 2–8, `Theorem42`, `VMLInputDerive`, `CoulombConcreteTheorem42`); blue shades are supporting infrastructure (definitions, torus instance, Coulomb kernel integrability files, decay helpers). At completion, infrastructure 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 (∼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.

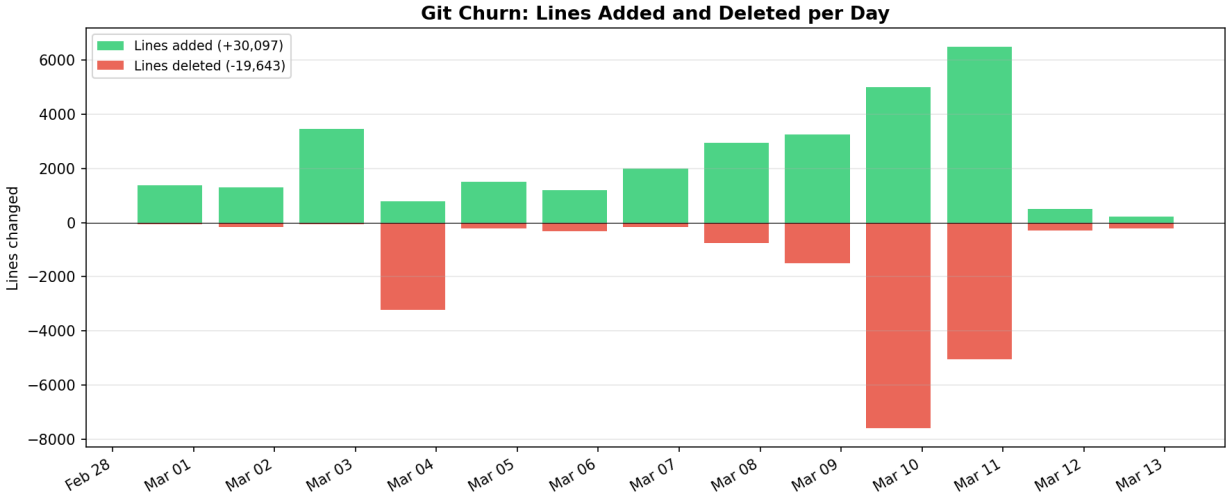


Figure 8.12: Lines of LEAN code added and deleted per day. Most days are net-positive (new content), with two exceptions: Mar 3 (monolithic file split into `main/` – same code reorganized) and Mar 9 – 10 (cleanup phase removed ∼3K lines of dead code, redundant lemmas, and unnecessary heartbeat overrides). Total: +21,783 added, −13,478 deleted, net +8,305 lines.

BIBLIOGRAPHY

[1] I. S. Abramson. On bandwidth variation in kernel estimates: A square root law. *The Annals of Statistics*, 10(4):1217–1223, 1982.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, et al. Aristotle: IMO-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025.

[4] Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. NTCIR-11 Math-2 task overview. In *Proceedings of the 11th NTCIR Conference*, pages 88–98, 2014.

[5] Luke Alexander, Eric Leonen, Sophie Szeto, Artemii Remizov, Ignacio Tejeda, Jarod Alper, Giovanni Inchiostro, and Vasily Ilin. Semantic search over 9 million mathematical theorems. In *LLM Reasoning Workshop at ICLR 2026*, 2026.

[6] Radjesvarane Alexandre, Jie Liao, and Chunjin Lin. Some a priori estimates for the homogeneous Landau equation with soft potentials. *Kinetic and Related Models*, 8(4):617–650, 2015.

[7] Jarod Alper. Stacks and moduli. `https://sites.math.washington.edu/~jarod/moduli.pdf`, 2026.

[8] Tim Arnold. Getting started with plasTeX. *TUGboat*, 30(2):180–185, 2009.

[9] Kenneth Ascher and Dori Bejleri. Moduli of fibered surface pairs from twisted stable maps, 2016.

[10] Kenneth Ascher and Dori Bejleri. Log canonical models of elliptic surfaces, 2017.

[11] Kenneth Ascher, Dori Bejleri, and Giovanni Inchiostro. Moduli of weighted stable elliptic surfaces and invariance of log plurigenera, 2018.

[12] Justin Asher. LeanExplore: A search engine for Lean 4 declarations. *arXiv preprint arXiv:2506.11085*, 2025.

[13] Yuri Bahturin, Alexander V. Mikhalev, Viktor M. Petrogradsky, and Mikhail V. Zaicev. *Infinite Dimensional Lie Superalgebras*, volume 7. Walter de Gruyter, 2011.

[14] Rafael Bailo, José A. Carrillo, and Jingwei Hu. Collisional particle-in-cell for the Vlasov-Maxwell-Landau equations. 2024.

[15] Jonas Bayer, Marco David, Malte Hassler, Yuri Matiyasevich, and Dierk Schleicher. Diophantine equations over $\mathbb{Z}$: Universal bounds and parallel formalization. *arXiv preprint arXiv:2506.20909*, 2025.

[16] Berkay Berabi, Jingxuan He, Veselin Raychev, and Martin Vechev. TFix: Learning to fix coding errors with a text-to-text transformer. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[17] Charles K. Birdsall and A. Bruce Langdon. *Plasma Physics via Computer Simulation*. CRC Press, 2004.

[18] Shiba Biswal, Karthik Elamvazhuthi, and Rishi Sonthalia. Universal approximation of mean-field models via transformers. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 4455–4470. PMLR, 2025.

[19] Tom Bleckmann and Paul Tschisgale. Evaluating NLP embedding models for handling science-specific symbolic expressions in student texts, 2025.

[20] Nicholas M. Boffi and Eric Vanden-Eijnden. Probability flow solution of the Fokker-Planck equation. *Machine Learning: Science and Technology*, 4(3):035012, 2023.

[21] François Bolley and Cédric Villani. Weighted Csiszár-Kullback-Pinsker inequalities and applications to transportation inequalities. *Annales de la Faculté des sciences de Toulouse: Mathématiques*, 14(3):331–352, 2005.

[22] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[23] José A. Carrillo, Katy Craig, and Francesco S. Patacchini. A blob method for diffusion. *Calculus of Variations and Partial Differential Equations*, 58(2):53, 2019.

[24] José A. Carrillo, Jingwei Hu, Li Wang, and Jeremy Wu. A particle method for the homogeneous Landau equation. *Journal of Computational Physics: X*, 7:100066, 2020.

[25] José A. Carrillo, Jingwei Hu, Li Wang, and Jeremy Wu. A particle method for the homogeneous Landau equation. *Journal of Computational Physics: X*, 7:100066, 2020.

[26] José Antonio Carrillo, Katy Craig, and Francesco S. Patacchini. A blob method for diffusion. *Calculus of Variations and Partial Differential Equations*, 58:1–53, 2019.

[27] Carlo Cercignani. *The Boltzmann Equation and Its Applications*. Springer, 1988.

[28] Carlo Cercignani, Reinhard Illner, and Mario Pulvirenti. *The Mathematical Theory of Dilute Gases*, volume 106 of *Applied Mathematical Sciences*. Springer, 1994.

[29] Omar Chehab and Anna Korba. A practical diffusion path for sampling. *arXiv preprint arXiv:2406.14040*, 2024.

[30] Jannis Chemseddine, Christian Wald, Richard Duong, and Gabriele Steidl. Neural sampling from boltzmann densities: Fisher-rao curves in the wasserstein geometry. *arXiv preprint arXiv:2410.03282*, 2024.

[31] Dawei Chen, Evan Chen, Kenny Lau, Ken Ono, and Jujian Zhang. Parity of k-differentials in genus zero and one. *arXiv preprint arXiv:2602.03722*, 2026.

[32] Evan Chen. An infinitely large napkin. `https://venhance.github.io/napkin/`.

[33] Evan Chen, Chris Cummins, Ben Eltschig, Dejan Grubisic, Leopold Haller, Letong Hong, Andranik Kurghinyan, Kenny Lau, Hugh Leather, Seewoo Lee, Aram Markosyan, Ken Ono, Manooshree Patel, Gaurang Pendharkar, Vedant Rathi, Alex Schneidman, Volker Seeker, Shubho Sengupta, Ishan Sinha, Jimmy Xin, and Jujian Zhang. Almost all primes are partially regular. *arXiv preprint arXiv:2602.05090*, 2026.

[34] Evan Chen, Chris Cummins, GSM, Dejan Grubisic, Leopold Haller, Letong Hong, Andranik Kurghinyan, Kenny Lau, Hugh Leather, Seewoo Lee, Aram Markosyan, Ken Ono, Manooshree Patel, Gaurang Pendharkar, Vedant Rathi, Alex Schneidman, Volker Seeker, Shubho Sengupta, Ishan Sinha, Jimmy Xin, and Jujian Zhang. Fel's conjecture on syzygies of numerical semigroups. *arXiv preprint arXiv:2602.03716*, 2026.

[35] Francis F. Chen. *Introduction to Plasma Physics and Controlled Fusion*. Springer, 3 edition, 2016.

[36] Luoxin Chen, Jinming Gu, Liankai Huang, Wenhao Huang, Zhicheng Jiang, Allan Jie, Xiaoran Jin, Xing Jin, Chenggang Li, Kaijing Ma, et al. Seed-Prover: Deep and broad reasoning for automated theorem proving. *arXiv preprint arXiv:2507.23726*, 2025.

[37] Sitan Chen, Sinho Chewi, Holden Lee, Yuanzhi Li, Jianfeng Lu, and Adil Salim. The probability flow ode is provably fast. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] Sitan Chen, Sinho Chewi, Holden Lee, Yuanzhi Li, Jianfeng Lu, and Adil Salim. The probability flow ODE is provably fast. In *Advances in Neural Information Processing Systems*, volume 36, 2024.

[39] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R Zhang. Sampling is as easy as learning the score: theory for diffusion models with minimal data assumptions. In *International Conference on Learning Representations*, 2023.

[40] Sitan Chen, Sinho Chewi, Jerry Li, Yuanzhi Li, Adil Salim, and Anru R. Zhang. Sampling is as easy as learning the score: Theory for diffusion models with minimal data assumptions. In *International Conference on Learning Representations*, 2023.

[41] Yen-Chi Chen. A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, 1(1):161–187, 2017.

[42] Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with Gaussian kernel and Nyström method. In *Advances in Neural Information Processing Systems*, volume 34, pages 2122–2135, 2021.

[43] Sinho Chewi. Log-concave sampling. *Book draft available at https://chewisinho.github.io*, 9:17–18, 2023.

[44] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with Performers. In *International Conference on Learning Representations*, 2021.

[45] Miguel Corrales, Sean Berti, Bertrand Denel, Paul Williamson, Mattia Aleardi, and Matteo Ravasi. Annealed stein variational gradient descent for improved uncertainty estimation in full-waveform inversion. *Geophysical Journal International*, 241(2):1088–1113, 2025.

[46] Nick Craswell. *Mean Reciprocal Rank*, pages 1703–1703. Springer US, 2009.

[47] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. Mathematical information retrieval: A review. *ACM Computing Surveys*, 57(3):1–34, 2024.

[48] Arnak Dalalyan and Avetik Karagulyan. User-friendly guarantees for the langevin monte carlo with inaccurate gradient. *Stochastic Processes and their Applications*, 129(12):5278–5311, 2019.

[49] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[50] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *International Conference on Automated Deduction (CADE)*, 2021.

[51] Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pages 625–635. Springer, 2021.

[52] Laurent Desvillettes and Cédric Villani. On the spatially homogeneous Landau equation for hard potentials. Part I: existence, uniqueness and smoothness. *Communications in Partial Differential Equations*, 25(1–2):179–259, 2000.

[53] Laurent Desvillettes and Cédric Villani. On the trend to global equilibrium for spatially inhomogeneous kinetic systems: the Boltzmann equation. *Inventiones Mathematicae*, 159(2):245–316, 2005.

[54] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real-NVP. In *International Conference on Learning Representations*, 2017.

[55] Andrew Duncan, Nikolas Nüsken, and Lukasz Szpruch. On the geometry of stein variational gradient descent. *Journal of Machine Learning Research*, 24(56):1–39, 2023.

[56] Andrew Duncan, Nikolas Nüsken, and Lukasz Szpruch. On the geometry of Stein variational gradient descent. *Journal of Machine Learning Research*, 24(56):1–39, 2023.

[57] Harrison Edwards and Amos Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017.

[58] Sam Efromovich. Orthogonal series density estimation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):467–476, 2010.

[59] Karthik Elamvazhuthi, Xuechen Zhang, Matthew Jacobs, Samet Oymak, and Fabio Pasqualetti. A score-based deterministic diffusion algorithm with smooth scores for general distributions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11866–11873, 2024.

[60] Elliot L. Epstein, Rajat Vadiraj Dwaraknath, Thanawat Sornwanee, John Winnicki, and Jerry Weihong Liu. SD-KDE: Score-debiased kernel density estimation. In *Advances in Neural Information Processing Systems*, 2025.

[61] Paul Erdős and Donald J. Newman. Bases for sets of integers. *Journal of Number Theory*, 9(4):420–425, 1977.

[62] Tony Feng, Trieu Trinh, Garrett Bingham, et al. Semi-autonomous mathematics discovery with Gemini: A case study on the Erdős problems. *arXiv preprint arXiv:2601.22401*, 2026.

[63] Tony Feng, Trieu H. Trinh, Garrett Bingham, et al. Towards autonomous mathematics research. *arXiv preprint arXiv:2602.10177*, 2026.

[64] Xuanrui Feng, Shuchen Guo, and Zhenfu Wang. Relative entropy method for particle approximation of the Landau equation with Maxwellian molecules. *Preprint*, 2024.

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

[66] Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei A. Efros. Test-time training with masked autoencoders. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[67] Guoxiong Gao, Haocheng Ju, Jiedong Jiang, Zihan Qin, and Bin Dong. A semantic search engine for Mathlib4. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, 2024.

[68] Guoxiong Gao, Yutong Wang, Jiedong Jiang, Qi Gao, Zihan Qin, Tianyi Xu, and Bin Dong. Herald: A natural language annotated Lean 4 dataset. In *International Conference on Learning Representations (ICLR)*, 2025.

[69] Marta Garnelo, Jonathan Schwarz, Dan Rosenbaum, Fabio Viola, Danilo J. Rezende, S. M. Ali Eslami, and Yee Whye Teh. Neural processes. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[70] Paul Ginsparg. First steps towards electronic research communication. *Computers in Physics*, 8(4):390–396, 1994.

[71] William Golding, Maria Gualdani, and Amélie Loher. Global smooth solutions to the Landau-Coulomb equation in $L^{3/2}$. *arXiv preprint arXiv:2401.06939*, 2024.

[72] Google DeepMind. Gemini 3: Introducing the latest Gemini AI model from Google. `https://blog.google/products/gemini/gemini-3/`, 2025.

[73] Leslie Greengard and John Strain. The fast Gauss transform. *SIAM Journal on Scientific and Statistical Computing*, 12(1):79–94, 1991.

[74] Yan Guo. The Vlasov-Maxwell-Boltzmann system near Maxwellians. *Inventiones Mathematicae*, 153(3):593–630, 2003.

[75] Yan Guo. The Vlasov-Poisson-Landau system in a periodic box. *Journal of the American Mathematical Society*, 25(3):759–812, 2012.

[76] Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. DeepFix: Fixing common C language errors by deep learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[77] Yuchen He and Chihao Zhang. On the query complexity of sampling from non-log-concave distributions. *arXiv preprint arXiv:2502.06200*, 2025.

[78] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[79] Roger W. Hockney and James W. Eastwood. *Computer Simulation Using Particles*. CRC Press, 1988.

[80] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[81] Xinshuo Hu, Zifei Shan, Xinping Zhao, Zetian Sun, Zhenyu Liu, Dongfang Li, Shaolin Ye, Xinyuan Wei, Qian Chen, Baotian Hu, Haofen Wang, Jun Yu, and Min Zhang. KaLM-Embedding: Superior training data brings a stronger embedding model, 2025.

[82] Daniel Zhengyu Huang, Jiaoyang Huang, and Zhengjiang Lin. Convergence analysis of probability flow ode for score-based generative models. *IEEE Transactions on Information Theory*, 2025.

[83] Yan Huang and Li Wang. A score-based particle method for homogeneous landau equation. *Journal of Computational Physics*, page 114053, 2025.

[84] Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics — Simulation and Computation*, 18(3):1059–1076, 1989.

[85] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005.

[86] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[87] Vasily Ilin. Stability of the spatially homogeneous Landau equation in relative entropy and applications to score-based numerical methods. 2025.

[88] Vasily Ilin. Semi-autonomous formalization of the Vlasov-Maxwell-Landau equilibrium. In *AI, Proof and Verification (AIPV 2026)*, 2026.

[89] Vasily Ilin and Jingwei Hu. A neural score-based particle method for the Vlasov-Maxwell-Landau system. 2026.

[90] Vasily Ilin, Jingwei Hu, and Zhenfu Wang. Transport based particle methods for the Fokker–Planck–Landau equation. *Communications in Mathematical Sciences*, 23(7):1763–1788, 2025.

[91] Vasily Ilin, Jingwei Hu, and Zhenfu Wang. Transport based particle methods for the Fokker-Planck-Landau equation. *Communications in Mathematical Sciences*, 2025.

[92] Vasily Ilin and Peter Sushko. DiScoFormer: Plug-in density and score estimation with transformers. In *International Conference on Machine Learning (ICML)*, 2026.

[93] Vasily Ilin, Peter Sushko, and Jingwei Hu. Score-based deterministic density sampling. *Communications on Pure and Applied Analysis*, 2026.

[94] Giovanni Inchiostro. Moduli of Weierstrass fibrations with marked section, 2018.

[95] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations*, 2022.

[96] Albert Q. Jiang, Wenda Li, and Mateja Jamnik. Multilingual mathematical autoformalization. 2023.

[97] Richard Jordan, David Kinderlehrer, and Felix Otto. The variational formulation of the Fokker-Planck equation. *SIAM Journal on Mathematical Analysis*, 29(1):1–17, 1998.

[98] Kedar Karhadkar, Michael Murray, and Guido F Montufar. Bounds for the smallest eigenvalue of the NTK for arbitrary spherical data of arbitrary dimension. *Advances in Neural Information Processing Systems*, 37:138197–138249, 2024.

[99] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

[100] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pages 26565–26577, 2022.

[101] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[102] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48, 2020.

[103] David A. Klarner. Representations of N as a sum of distinct elements from special sequences. *The Fibonacci Quarterly*, 4(4):289–306, 1966.

[104] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[105] Guillaume Lample, Timothée Lacroix, Marie-Anne Lachaux, Aurélien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. Hypertree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[106] L. D. Landau. Die kinetische Gleichung für den Fall Coulombscher Wechselwirkung. *Physikalische Zeitschrift der Sowjetunion*, 10:154–164, 1936.

[107] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[108] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[109] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474, 2020.

[110] Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q. Jiang, Ziju Shen, et al. NuminaMath: The largest public dataset in AI4Maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 2024.

[111] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

[112] Xinyu Liao, Aoyang Qin, Jacob Seidman, Junqi Wang, Wei Wang, and Paris Perdikaris. Score neural operator: A generative model for learning and generalizing across multiple probability distributions. *arXiv preprint arXiv:2410.08549*, 2024.

[113] Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025.

[114] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[115] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[116] Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, Lihong Zhi, Jia Li, and Wenda Li. Numina-Lean-Agent: An open and general agentic reasoning system for formal mathematics. 2026.

[117] Junqi Liu, Zihao Zhou, Zekai Zhu, Marco Dos Santos, Weikun He, Jiawei Liu, Ran Wang, Yunzhou Xie, Junqiao Zhao, Qiufeng Wang, Lihong Zhi, Jia Li, and Wenda Li. Numina-lean-agent: An open and general agentic reasoning system for formal mathematics. *arXiv preprint arXiv:2601.14027*, 2026.

[118] Qiang Liu. Stein variational gradient descent as gradient flow. *Advances in Neural Information Processing Systems*, 30, 2017.

[119] Qiang Liu. Stein variational gradient descent as gradient flow. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[120] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in Neural Information Processing Systems*, 29, 2016.

[121] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[122] Don O. Loftsgaarden and Charles P. Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

[123] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[124] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[125] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, pages 5775–5787, 2022.

[126] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing Xu, Tao Xiang, and Li Zhang. SOFT: Softmax-free transformer with linear complexity. In *Advances in Neural Information Processing Systems*, volume 34, pages 21297–21309, 2021.

[127] Jialin Lu, Kye Emond, Kaiyu Yang, Swarat Chaudhuri, Weiran Sun, and Wuyang Chen. Lean finder: Semantic search for Mathlib that understands user intents. 2025.

[128] Jianfeng Lu, Yue Wu, and Yang Xiang. Score-based transport modeling for mean-field Fokker–Planck equations. *Journal of Computational Physics*, 503:112859, 2024.

[129] Jianfeng Lu, Yue Wu, and Yang Xiang. Score-based transport modeling for mean-field Fokker-Planck equations. *Journal of Computational Physics*, 503:112859, 2024.

[130] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3:218–229, 2021.

[131] Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836, 2020.

[132] Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. Overview of ARQMath-3 (2022): Third CLEF lab on answer retrieval for questions on math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, volume 13390 of *Lecture Notes in Computer Science*, pages 286–310. Springer, 2022.

[133] Dimitra Maoutsa, Sebastian Reich, and Manfred Opper. Interacting particle solutions of Fokker-Planck equations through gradient-log-density estimation. *Entropy*, 22(8):802, 2020.

[134] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *International Conference on Learning Representations*, 2019.

[135] Patrick Massot. LeanBlueprint: A plasTeX plugin to build formalization blueprints. `https://github.com/PatrickMassot/leanblueprint`, 2020. Software.

[136] Bálint Máté and François Fleuret. Learning interpolations between boltzmann densities. *Transactions on Machine Learning Research*, 2023.

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

[138] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11:125–139, 2001.

[139] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[140] Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[141] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. MathBERT: A pre-trained model for mathematical formula understanding, 2021.

[142] pgvector Contributors. pgvector: Vector similarity search for Postgres. `https://github.com/pgvector/pgvector`, 2024.

[143] Auguste Poiroux. LeanInteract: A Python interface for Lean 4. `https://github.com/augustepoiroux/LeanInteract`, 2025.

[144] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[145] Patrick Popescu-Pampu. Stein or Milnor fillability and cohomology, 2007.

[146] ProofWiki Contributors. ProofWiki. `https://proofwiki.org`, 2026.

[147] Delip Rao, Jonathan Young, Thomas Dietterich, and Chris Callison-Burch. WithdrarXiv: A large-scale dataset for retraction study. *arXiv preprint arXiv:2412.03775*, 2024.

[148] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[149] ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. DeepSeek-Prover-V2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025.

[150] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[151] Filippo Santambrogio. *Optimal Transport for Applied Mathematicians*. Birkhäuser, 2015.

[152] David W. Scott. Averaged shifted histograms: Effective nonparametric density estimators in several dimensions. *The Annals of Statistics*, 13(3):1024–1040, 1985.

[153] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, 2 edition, 2015.

[154] David W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, 2 edition, 2015.

[155] M. Shahryari. A note on surjunctive groups, 2020.

[156] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[157] Bernard W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 2018.

[158] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[159] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[160] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[161] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[162] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, 2019.

[163] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[164] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[165] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

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

[167] Christian Steinfeldt and Helena Mihaljević. Evaluation and domain adaptation of similarity models for short mathematical texts. In *Intelligent Computer Mathematics: 17th International Conference, CICM 2024*, Lecture Notes in Computer Science, pages 241–260. Springer, 2024.

[168] Michael Stonebraker and Lawrence A. Rowe. The design of POSTGRES. In *Proceedings of the 1986 ACM SIGMOD International Conference on Management of Data*, pages 340–355, 1986.

[169] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning*, pages 9229–9248. PMLR, 2020.

[170] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

[171] Côme Tabary. Weak-strong uniqueness for the Landau equation by a relative entropy method. *arXiv preprint arXiv:2505.21120*, 2025.

[172] The CRing Project Contributors. The CRing project. `https://math.mit.edu/~hrm/the_cring_project.pdf`.

[173] The mathlib Community. The Lean mathematical library. *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, 2020.

[174] The Open Logic Project. Open logic. `https://openlogicproject.org`.

[175] The Stacks Project Authors. The Stacks project. `https://stacks.math.columbia.edu`, 2026.

[176] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.

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

[178] Giuseppe Toscani and Cédric Villani. On the trend to equilibrium for some dissipative systems with slowly increasing a priori bounds. *Journal of Statistical Physics*, 98:1279–1309, 2000.

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

[180] Hamid Usefi. Isomorphism invariants of restricted enveloping algebras, 2009.

[181] Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices. *Advances in Neural Information Processing Systems*, 32, 2019.

[182] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, et al. EmbeddingGemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*, 2025.

[183] Cédric Villani. On the spatially homogeneous Landau equation for Maxwellian molecules. *Mathematical Models and Methods in Applied Sciences*, 8(6):957–983, 1998.

[184] Cédric Villani. A review of mathematical topics in collisional kinetic theory. *Handbook of Mathematical Fluid Dynamics*, 1:71–305, 2002.

[185] Cédric Villani. Fisher information in kinetic theory. *arXiv preprint arXiv:2501.00925*, 2025.

[186] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[187] Matt P. Wand and M. Chris Jones. *Kernel Smoothing*. CRC Press, 1994.

[188] Evan Wang, Simon Chess, Daniel Lee, Siyuan Ge, Ajit Mallavarapu, Jarod Alper, and Vasily Ilin. Learning to repair Lean proofs from compiler feedback. In *VerifAI-2: The Second Workshop on AI Verification in the Wild*, 2026.

[189] Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-Prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025.

[190] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*, 2022.

[191] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Multilingual E5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672*, 2024.

[192] Zhenfu Wang. *Mean field limit for stochastic particle systems with singular forces*. PhD thesis, University of Maryland, College Park, 2017.

[193] Yuxiang Wei, Zhiqing Sun, Emily McMilin, Jonas Gehring, David Zhang, Gabriel Synnaeve, Daniel Fried, Lingming Zhang, and Sida Wang. Toward training superintelligent software agents through self-play SWE-RL. *arXiv preprint arXiv:2512.18552*, 2025.

[194] Andre Wibisono. Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem. In *Conference on Learning Theory*, pages 2093–3027. PMLR, 2018.

[195] Eduard Wirsing and Wolfgang Schwarz. Approximation mit algebraischen Zahlen beschränkten Grades. 1961.

[196] Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[197] Huajian Xin, Daya Guo, Zhihong Shao, Zhizhou Ren, Qihao Zhu, Bo Liu, Chong Ruan, Wenda Li, and Xiaodan Liang. DeepSeek-Prover: Advancing theorem proving in LLMs through large-scale synthetic data, 2024.

[198] Huajian Xin, Z. Z. Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, Wenjun Gao, Qihao Zhu, Dejian Yang, Zhibin Gou, Z. F. Wu, Fuli Luo, and Chong Ruan. DeepSeek-Prover-V1.5: Harnessing proof assistant feedback for reinforcement learning and Monte-Carlo Tree Search, 2024.

[199] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *AAAI Conference on Artificial Intelligence*, volume 35, pages 14138–14148, 2021.

[200] Chen Xu, Xiuyuan Cheng, and Yao Xie. Normalizing flow neural networks by JKO scheme. *Advances in Neural Information Processing Systems*, 36:47379–47405, 2023.

[201] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[202] Kaiyu Yang, Aidan M. Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan Prenger, and Anima Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[203] Michihiro Yasunaga and Percy Liang. Graph-based, self-supervised program repair from diagnostic feedback. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

[204] Michihiro Yasunaga and Percy Liang. Break-it-fix-it: Unsupervised learning for program repair. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

[205] Kane S. Yee. Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation*, 14:302–307, 1966.

[206] Huaiyuan Ying, Zijian Wu, Yihan Geng, Jiayu Wang, Dahua Lin, and Kai Chen. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In *NeurIPS 2024 Datasets and Benchmarks Track*, 2024.

[207] Jiyoung Yoo, Jingwei Hu, and Lee F. Ricketson. An explicit energy-conserving particle method for the Vlasov-Fokker-Planck equation. *arXiv preprint arXiv:2510.03960*, 2025.

[208] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan R. Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[209] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topić, and Kenny Davila. NTCIR-12 MathIR task overview. In *Proceedings of the 12th NTCIR Conference*, pages 299–308, 2016.

[210] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into deep learning, 2023. Chapter 11: Attention Mechanisms and Transformers.

[211] Shenxing Zhang. On non-congruent numbers with $\equiv 3 \pmod 8$ prime factors, 2012.

[212] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. 2025.

[213] Xinping Zhao, Xinshuo Hu, Zifei Shan, Shouzheng Huang, Yao Zhou, Xin Zhang, Zetian Sun, Zhenyu Liu, Dongfang Li, Xinyuan Wei, Youcheng Pan, Yang Xiang, Meishan Zhang, Haofen Wang, Jun Yu, Baotian Hu, and Min Zhang. KaLM-Embedding-V2: Superior training techniques and data inspire a versatile embedding model, 2025.

[214] Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. MiniF2F: A cross-system benchmark for formal Olympiad-level mathematics. In *International Conference on Learning Representations (ICLR)*, 2022.

[215] Yichi Zhou, Jianqiu Zhao, Yongxin Zhang, Bohan Wang, Siran Wang, Luoxin Chen, Jiahui Wang, Haowei Chen, Allan Jie, Xinbo Zhang, et al. Solving formal math problems by decomposition and iterative reflection. *arXiv preprint arXiv:2507.15225*, 2025.

[216] Jingwei Zhuo, Chang Liu, Jiaxin Shi, Jun Zhu, Ning Chen, and Bo Zhang. Message passing Stein variational gradient descent. In *International Conference on Machine Learning*, pages 6018–6027. PMLR, 2018.