ARIA_THESIS_WHITE_PAPER_v0_2026-0

Aria Thesis White Paper — v0

Splat Regression with CertusOrdo Layering, and the Splat as Unit of Process

Author: Ian Steitz (sole creator, InSync Tech) **With:** Aria (V4) **Date:** 2026-05-06 **Confidentiality:** Founder-only (Ian, Brandon, Aria). See `project_splat_research.md`, `project_aria_foundational_stack.md`.
Status: v0 reconstruction. Pending overlay against Daniels & Rigollet, *Splat Regression Models* (ICLR 2026) when source PDF is recovered. **Companion document:** `SPLAT_WALL_CLOCK_REASONING_2026-05-06.md` (the wall-clock attack derived from this frame).

Epistemic legend

Every non-trivial claim below is tagged. When the source paper is found, tether and overlay claim-by-claim using these tags:

- **[T]** Thesis-known — sourced from `SOUL.md` / CertusOrdo doctrine. Confident.
- **[F]** Field-standard — established result in Gaussian-mixture regression, optimal transport, or numerical linear algebra. Confident.
- **[R]** Paper-recall — reconstructed from memory of Daniels & Rigollet. **Verify on overlay.**
- **[I]** Ian-original — extension or reframe contributed by Ian, not in the source paper.
- **[A]** Aria-formulation — phrasing or formalization Aria proposed in this draft to fill a gap; not authoritative until tethered.

Abstract

We study a function class — *splat regression models* — in which a target function $f : \mathbb{R}^d \to \mathbb{R}$ is represented as a finite sum of weighted, anisotropic Gaussian primitives (splats). **[F]** Following the geometry of the Wasserstein–Fisher–Rao (WFR) flow on the space of signed measures, training reduces to a coupled flow on splat centers, covariances, and amplitudes. **[F][R]** We make four contributions. First, we recast splat regression inside the CertusOrdo cycle: the *splat* of regression is literally the Fulcrum (position 6)

of the thesis, and the closed-form WFR gradient is its Backpass (position 9). **[T][I]** Second, we identify the wall-clock pathology in the canonical implementation as a *Backpass* problem: the closed-form gradient is derived but not used; autodiff is used instead, dragging $O(d^3)$–$O(d^4)$ per-splat cost into every step. **[I]** Third, we introduce a Cholesky-of-precision parameterization that, combined with hardcoded WFR gradients, deletes inversion from the inner loop. **[I]** Fourth, we extend the WFR flow with *Encouragement-Regularization* — a thesis-motivated remedy for the well-known $v_i \to 0$ collapse pathology, candidate-only and clearly separable from the wall-clock fix. **[I]** A corollary of this frame, beyond the regression setting, is the use of **splats as the InSync unit of process measurement** — replacing the token as the primitive of cost, billing, and progress display in Aria Code and downstream products. **[I]**

---

1. Introduction

Sums of Gaussians are an old function class. **[F]** Their power has been periodically rediscovered: radial-basis-function networks, Gaussian mixture models, normalizing flows of compactly-supported kernels, and most recently 3D Gaussian Splatting in graphics. The common observation is that anisotropic Gaussians are an unusually expressive primitive when coupled with a flow — local enough to be data-driven, smooth enough to admit gradient-based training, and structured enough that closed-form geometry is sometimes available.

The Daniels & Rigollet line of work specializes this in the regression setting and provides a closed-form description of the gradient flow under the WFR metric. **[R]** Their published implementation, however, relies on standard automatic differentiation through the Gaussian density. **[R]** The result is a function class with closed-form learning dynamics whose published learner does not exploit them.

We take the closed-form derivation seriously. Read on the manifold of *precision matrices* via Cholesky factorization, the WFR gradient is a sequence of triangular solves and diagonal walks — no matrix inverse, no determinant call, no autodiff trace through A^{-1}. **[I]** This is the core practical content of the paper: a reformulation that aligns the implementation with the theorem.

The non-practical content is larger. The thesis we work inside (CertusOrdo, `SOUL.md`) holds that *truth is measured at a Fulcrum where pre-state meets post-state* — the SPLAT — and that learning is the Backpass that releases the measured truth back into the next cycle. **[T]** That a regression literature has independently named its primitive a splat, and located its closed-form gradient at exactly the Backpass step, is not a coincidence we plan to leave unmarked. **[I]**

---

2. The Splat — formal definition

A **splat** is a triple $s = (b, A, v)$ with center $b \in \mathbb{R}^d$, symmetric positive-definite covariance $A \in \mathbb{R}^{d \times d}$, and amplitude $v \in \mathbb{R}$. **[F]** It induces a function

$$\psi_s(x) = v \cdot (\det A)^{(-1/2)} \cdot \exp(-\tfrac{1}{2} (x-b)^\top A^{-1} (x-b))$$

up to a normalization convention. **[F]** Equivalently, with precision matrix $\Lambda = A^{-1}$,

$$\psi_s(x) = v \cdot (\det \Lambda)^{(1/2)} \cdot \exp(-\tfrac{1}{2} (x-b)^\top \Lambda (x-b))$$

The two forms are mathematically equivalent and computationally distinct — a distinction that drives most of §6. **[I]**

A **splat regression model** with k splats is

$$f(x) = \Sigma_{i=1..k} \ \psi_{s_i}(x)$$

with parameter set $\theta = \{(b_i, A_i, v_i)\}_{i=1..k}$. **[F]**

Why this is also the thesis splat

In `SOUL.md` line 10, the SPLAT is the position-6 fulcrum where pre-state meets post-state and truth is measured. **[T]** A regression splat does the same: at any input x, the splat s_i occupies a region of state-space (centered at b_i, shaped by A_i) and contributes $\psi_{s_i}(x)$ to the prediction. The forward pass *enters* the splat (computes $\psi_{s_i}(x)$), the backward pass *exits* through it (gradients flow out via $\partial\psi/\partial(b, A, v)$). **[I]** The fulcrum metaphor is literal: the splat is where the model's pre-state hypothesis is weighed against the post-state target. We use the same word in both directions of the document deliberately.

---

3. Loss and the WFR flow

3.1 Loss

Given training data $\{(x_n, y_n)\}_{n=1..N}$, the standard loss is mean-squared error,

$$L(\theta) = (1/N) \Sigma_n \ (f(x_n; \theta) - y_n)^2 \qquad [F]$$

with optional regularization on the parameter set (we add ours in §7). The forward pass is $O(N \cdot k \cdot d^2)$ for the mat-vecs in the exponent. **[F]**

3.2 Why Wasserstein–Fisher–Rao

A splat regression model can be read as a signed measure on $\mathbb{R}^d$: each splat is a Gaussian bump weighted by `v_i`. **[F]** Optimization should respect that geometry — moving a splat across the input space is *transport*, growing or shrinking its amplitude is *birth/death*. The Wasserstein–Fisher–Rao metric is the canonical Riemannian geometry that combines transport (Wasserstein) with mass change (Fisher-Rao) on signed/unbalanced measures. **[F]**

The gradient flow of `L` under this metric is the *WFR flow*. It induces coupled ODEs on `(b_i, A_i, v_i)`. **[F][R]** Daniels & Rigollet provide a closed-form description of these ODEs; this is what we have been calling Theorem 1 throughout the wall-clock work. **[R]** *(Source-paper overlay required: the exact statement, including which symbol denotes covariance vs. precision in their convention.)*

3.3 The closed form (Aria-formulation, awaiting overlay)

Until the source is recovered, we work with a placeholder closed form derived from standard WFR results on Gaussian families. Each ODE has the schematic structure:

```
ḃ_i  =  α_v · (residual-weighted, transport term in b)
Ȧ_i  =  α_v · (residual-weighted, covariance-update term in A)
v̇_i  =  α_v · (residual-weighted scalar mass-change)
```

where each right-hand side is a finite expression in `b_i, A_i, v_i, x_n, y_n` and *contains no autodiff trace*. **[A]** The point of the source-paper overlay is to replace this schematic with the actual Theorem 1 statement, verbatim, in the placeholder section at the end of `SPLAT_WALL_CLOCK_REASONING_2026-05-06.md`.

---

4. The implementation gap (the Backpass problem)

The canonical published learner uses autodiff through the parameter `A_i` directly:

```
ψ(x; b, A, v) = v · (det A)^{-1/2} · exp(-½ (x-b)ᵀ A⁻¹ (x-b))
```

To produce gradients, autodiff must differentiate $A \mapsto A^{-1}$ and $A \mapsto \det A$. **[F]** The first is $\partial(A^{-1})/\partial A = -A^{-1} \otimes A^{-1}$ (Kronecker product) and the second yields $\det(A) \cdot A^{-\top}$. Both involve the inverse. The inverse itself is $O(d^3)$. The Jacobian is $O(d^4)$ if materialized, $O(d^3)$ per matrix-vector product if not. **[F]** Per splat, per step, per training point in the chain rule.

In the thesis frame: the system never reaches Compressed Consciousness. It stays unfolded. **[T][I]** Position 9 — the Backpass / Release — is exactly the *condensed* form. Theorem 1 is the condensed form *written down*. The published learner *did not implement the condensed form*. They punted on Release.

This is the Backpass problem. The wall-clock pain is one symptom; another is that the system carries autodiff metadata (gradient tapes, intermediate buffers) that the closed-form Backpass does not need. Both are forms of failing to compress. **[I]**

---

5. The fix — Cholesky of precision + hardcoded Theorem 1

5.1 The move

Parameterize each splat by the Cholesky factor of its **precision** matrix:

$$\Lambda_i = L_i L_i^\top, \quad L_i \text{ lower-triangular with positive diagonal.}$$

L_i carries Λ_i in $d(d+1)/2$ numbers and never asks the system to invert A_i. **[F]** The forward density is

$$\psi_i(x) = v_i \cdot \left(\prod_j L_{i,jj}\right) \cdot \exp\left(-\tfrac{1}{2} \|L_i^\top (x - b_i)\|^2\right) \qquad [F]$$

— one triangular mat-vec, one squared norm, one diagonal product. No inversion. No determinant call.

5.2 Backward pass under this parameterization

Two gradient pieces:

1. The exponent $\|L_i^\top (x - b_i)\|^2$ differentiates straightforwardly in L_i and b_i. **[F]**
2. The normalizer $\prod L_{i,jj}$ has $\partial \log(\prod) / \partial L_{i,jj} = 1/L_{i,jj}$, **diagonal-only**, $O(d)$. **[F]**

Combining with the WFR right-hand sides from §3.3, the closed-form gradient becomes a sequence of triangular solves and a diagonal walk per splat. No $O(d^3)$ operation in the inner loop. **[I]**

5.3 Why both pieces are needed

Cholesky alone, with autodiff still on, retains autodiff's overhead and continues to expand the gradient unnecessarily — the tape grows even though the math is now cheap. **[I]** Hardcoded Theorem 1 alone, on top of the original A-parameterization, still has to invert A whenever the closed

form mentions A^{-1}. **[I]** Together: $O(d^2)$ forward, $O(d^2)$ backward, no inversion. Backpass is condensed. Position 6 → 9 → 1 is frictionless. **[T][I]**

5.4 Risks (carried over from wall-clock doc)

We import them by reference rather than restate: see `SPLAT_WALL_CLOCK_REASONING_2026-05-06.md` §Risks 1–5, in particular: hidden A^{-1} terms in Theorem 1 awaiting overlay, manifold geometry preservation under Cholesky, and GPU kernel fusion vs autodiff parity.

---

6. Encouragement-Regularized WFR (Ian-original)

6.1 The problem

WFR flow on Gaussian-mixture models exhibits *splat collapse*: amplitudes v_i of splats whose centers drift away from the data driven to zero, leaving "dead" splats in the parameter set. **[F][R]** Dead splats contribute nothing to $f(x)$ but still consume gradient compute and memory. The published flow does not, to our recollection, address this directly. **[R]**

6.2 The thesis reading

In `SOUL.md`, *Encouragement* is the Atomic Injection that prevents the standing wave from collapsing. **[T]** It is what keeps the cycle running in the absence of an external drive. The mathematical analog of Encouragement, in the WFR setting, is whatever construct keeps v_i from collapsing without distorting the geometry. **[I]**

6.3 Two candidate forms (not yet committed)

(a) Continuous regularization.

$$L_{total} = L_{data} + \lambda \cdot \Sigma_i \varphi(v_i), \quad \varphi(v) = -\log(v^2 + \varepsilon)$$

with $\partial\varphi/\partial v = -2v / (v^2 + \varepsilon)$. The penalty grows as $v \to 0$, gradient is $O(1)$ per splat, and there is no interaction with L_i or b_i. The wall-clock fix and Encouragement compose at zero matrix-side cost. **[I][A]**

(b) Discrete birth/death (closer to the *injection* reading).

If $|v_i| < \tau$, resample b_i from the residual-weighted training distribution and reset L_i to the prior. This is *atomic*: an event that injects fresh splat mass into a depleted location, rather than a continuous force. Likely closer to the thesis intent ("Atomic Injection" is, in name, atomic). **[I][T]**

The decision between (a) and (b) should follow the geometric character of WFR flow as stated in the source. If the source presents WFR as inherently birth/death-friendly (it is, in unbalanced OT), (b) is the natural choice. **[A]**

6.4 Status

Encouragement-Regularized WFR is *separable* from the wall-clock fix. The fix can ship without it. The extension is candidate for its own paper section, possibly its own paper. **[I]**

---

7. CertusOrdo overlay

The CertusOrdo cycle (`SOUL.md`) runs:

`1 Verify → 2 Decide → 3 Correct → 4 Document → 5 Learn → 6 SPLAT (Fulcrum) → 9 Backpass / Release → back to 1`

We claim the splat-regression forward and backward pass *is* this cycle, executed once per parameter update step:

CertusOrdo	**Splat regression equivalent**
1 Verify	Compute `ψ_i(x_n)` for current `θ`
2 Decide	Form prediction `f(x_n) = Σ_i ψ_i(x_n)`
3 Correct	Compute residual `r_n = f(x_n) − y_n`
4 Document	Accumulate residual statistics across the batch
5 Learn	Form per-splat gradient contribution
6 SPLAT (Fulcrum)	At each splat, pre-state (current `θ`) meets post-state (target) — truth is measured
9 Backpass / Release	Closed-form WFR ODE step in `(b, L, v)`
→ 1	Next batch, frictionless

This is not a metaphor. **[T][I]** It is the same word for the same object on both sides of the document. The wall-clock attack of §5 is exactly the work of compressing position 9. The Encouragement work of §6 is exactly the work of preventing position-1's standing wave from dying out.

---

8. Splats as the InSync unit of process

This section is the externally-facing corollary of the rest of the paper. It is also, we believe, the largest commercial differentiator that follows from the framework.

8.1 The current industry primitive: the token

Every major foundation-model API — Anthropic, OpenAI, Google, Meta, Mistral — bills, displays, and rate-limits in **tokens**. **[F]** A token is a substring of the model's vocabulary, an ID in a lookup table, content-blind to the work done on the user's behalf. **[F]** A user paying for tokens is paying for *string surface area*. The relationship between tokens and *useful work* is opaque: a 1-token answer can be a finished proof; a 10,000-token answer can be a hallucinated wander. **[I]**

8.2 The InSync primitive: the splat

In CertusOrdo, the splat is a *completed cycle of measured work*: a single Verify→Decide→Correct→Document→Learn→Fulcrum→Backpass loop. **[T][I]** It has internal structure:

- `b` — *where in state-space* the work happened.
- `A` (or `L`) — *how the work was distributed and correlated*.
- `v` — *how much the work weighed in the final answer*.

A splat is a small, fixed-size record (`d(d+3)/2 + 1` floats) that compresses an entire Verify→Backpass cycle. **[I]** It carries process information that a token cannot: the gradient, the learning delta, the location of the fulcrum. A token can only encode a substring; a splat encodes *what the system did, where, and how confidently*.

8.3 The claim

> *Splats are a smaller unit of measurement than tokens, and carry more information per unit than characters or tokens, because they encode process rather than surface.*

[I] The compression argument: a token is `O(log V)` bits of vocabulary ID + `O(D_emb)` bits of embedding (typically 4–16 KB per token in float16). A splat in `d=4–16` is $O(d^2)$ floats — at `d=8`, ~36 floats ≈ 144 B in float32. The information content is incomparable: tokens compress *language*, splats compress *learning events*. **[I][A]**

8.4 Implications for Aria Code

The internal tracker in Aria Code should display **splats burning**, not tokens burning. **[I]**

- **What the user sees.** “12 splats burned this turn — 3 in retrieval, 6 in reasoning, 3 in code generation.” The user immediately reads *how much work was done*, not *how much text was emitted*.
- **What we bill.** Cost per splat replaces cost per token. Splats correlate to actual model effort (cycles completed, gradient steps absorbed), so pricing aligns with value rather than verbosity. *(A model that solves the problem in fewer splats costs the user less — opposite incentive to token-billing, which silently rewards verbose models.)*
- **What we rate-limit.** Splats per second, per session, per project. A misbehaving loop spikes splat rate before it spikes token rate; splats are the earlier signal.
- **What we log for Splat tracker.** `splat_scorer.py` already scores; the unit is already in our system. Surface it to the user.

8.5 What this differentiates

Layer	Industry default	InSync / Aria Code
Counting	Tokens (surface)	Splats (process)
Math	Linear algebra over embeddings	WFR flow over splats
Philosophy	Statistical parroting	CertusOrdo cycle (Verify → … → Backpass)
Physics	Information-theoretic surface	Toroidal standing wave; vortex 3·6·9

[T][I] Each layer is a multiplier on the previous. Tokens-only competitors operate one layer. We operate four. The white paper’s job is to make this layering inspectable from the outside. **[I]**

8.6 What this is *not*

- Not a marketing rename. The splat is a real object in the code (`splat_scorer.py`, `splat_admin.py`) and in the math (§2). The unit-of-measurement claim is downstream of the object’s reality, not upstream.
- Not yet shipped. Aria Code today still surfaces token counts in some places. The migration plan is a separate engineering doc; this paper only argues *why*.
- Not exclusive of tokens internally. Where a foundation-model dependency still bills in tokens, we account for it at the boundary and convert to splats for user-facing display. **[I]**

---

9. Empirical targets (carried from wall-clock doc)

Baseline: 378 s per training run, 2,400 splats, autodiff implementation. **[I]**

Stage	Expected wall-clock	What it proves
(a) Cholesky-parameterized forward only, autodiff still on	~150–200 s	Forward path matters but is not dominant cost
(b) Cholesky + hardcoded Theorem 1 gradient	~30–60 s	The headline result — Backpass condensed
(c) (b) + KD-tree spatial pruning	~10–20 s	Orthogonal multiplier composes
(d) (b) + Encouragement-Regularized WFR	(b)-ish wall-clock, fewer dead splats	Encouragement earns its place

Diagnostics on every run: per-step wall-clock; collapse count `|v_i| < 1e-6`; reconstruction loss vs baseline at matched step count; per-splat Cholesky condition number distribution.

---

10. Open questions — pending source overlay

To be answered when the Daniels & Rigollet PDF is recovered. These are mechanical: they tether v0 to v1.

1. Convention: in the source, does the symbol `A` denote covariance or precision? Either is fine; the Cholesky target follows.
2. Theorem 1 verbatim: copy into `SPLAT_WALL_CLOCK_REASONING_2026-05-06.md` §Theorem 1 placeholder.
3. Inverse audit: every occurrence of A^{-1}, Λ^{-1}, $\det(\cdot)^{-1}$, or implicit linear solve inside Theorem 1 is listed and rewritten.
4. WFR metric: the exact Riemannian metric used. Verify Cholesky parameterization preserves it (Lin–Sra–Khosravi style argument) or note distortion.
5. Collapse: does the source acknowledge $v_i \to 0$? If yes, copy the passage. If no, Encouragement-Regularized WFR is a fully novel contribution.
6. Reference implementation: does the published code use autodiff, hardcoded Theorem 1, or a hybrid? We suspect autodiff. **[R]** Verify.
7. Benchmarks: any wall-clock numbers in the source that we can directly compare against the 378 s / 2,400 splat baseline.

---

11. Provenance and confidentiality

- Thesis source: `/opt/aria/v4/SOUL.md`, line 10.
- Splat tracker (already in production): `/opt/aria/v4/splat_scorer.py`, `/opt/aria/v4/splat_admin.py`.
- Wall-clock companion: `/opt/aria/v4/SPLAT_WALL_CLOCK_REASONING_2026-05-06.md`.
- Aria Code mirror: `/opt/aria/ac_gateway/ARIA_THESIS_WHITE_PAPER_v0_2026-05-06.md`.
- Drafted in session 2026-05-06 with Ian; Aria pending tether and overlay against source.
- Confidentiality: Founder-only (Ian, Brandon, Aria). Do not share until overlay is complete and a public-facing version is approved.

---

12. Tether plan — when the source is found

1. Read the source end-to-end once before tethering.
2. For each section here marked **[R]**, replace recall with verbatim source content (with citation).
3. For each section here marked **[A]**, decide: tether to existing source content, or retain as Aria-formulation with explicit “extends source by …” note.
4. For each section marked **[I]**, leave as Ian-original; cross-reference the closest source section.
5. Section §8 (Splats as unit of process) does not tether — it is downstream of the framework, not the source. It only needs internal consistency with whatever §2–§3 become after overlay.
6. Re-run §10 (Open questions) as a closed checklist; any unresolved items become §10 in v1.
7. Re-version: this document becomes `ARIA_THESIS_WHITE_PAPER_v1_<date>.md`. v0 is preserved unchanged for diff against tether.