FlowAdam: Implicit Regularization via Geometry-Aware Soft Momentum Injection

Devender Singh
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL, Canada
devenders@mun.ca

Tarun Sheel
Department of Mathematics and Statistics
Memorial University of Newfoundland
St. John's, NL, Canada
tksheel@mun.ca

***Abstract*—Adaptive moment methods such as Adam use a diagonal, coordinate-wise preconditioner based on exponential moving averages of squared gradients. This diagonal scaling is coordinate-system dependent and can struggle with dense or rotated parameter couplings, including those in matrix factorization, tensor decomposition, and graph neural networks, because it treats each parameter independently. We introduce FlowAdam, a hybrid optimizer that augments Adam with continuous gradient-flow integration via an ordinary differential equation (ODE). When EMA-based statistics detect landscape difficulty, FlowAdam switches to clipped ODE integration. Our central contribution is Soft Momentum Injection, which blends ODE velocity with Adam's momentum during mode transitions. This prevents the training collapse observed with naive hybrid approaches. Across coupled optimization benchmarks, the ODE integration provides implicit regularization, reducing held-out error by 10-22% on low-rank matrix/tensor recovery and 6% on Jester (real-world collaborative filtering), also surpassing tuned Lion and AdaBelief, while matching Adam on well-conditioned workloads (CIFAR-10). MovieLens-100K confirms benefits arise specifically from coupled parameter interactions rather than bias estimation. Ablation studies show that soft injection is essential, as hard replacement reduces accuracy from 100% to 82.5%.**

***Index Terms*—implicit regularization, adaptive optimization, gradient flow, soft momentum injection, matrix completion, deep learning**

I. Introduction

Adam [1] has become the default optimizer for deep learning, combining momentum with per-parameter adaptive learning rates via the update $\theta_{t+1} = \theta_t - \alpha \cdot m_t/(\sqrt{v_t} + \epsilon)$, where m_t and v_t are exponential moving averages of the gradient and squared gradient, respectively.

While effective on many problems, Adam's per-parameter scaling uses a diagonal preconditioner that is coordinate-system dependent—treating parameters as if they can be optimized independently [2]. We call a problem *coupled* when gradients for one parameter block depend strongly on others (i.e., large off-diagonal Hessian blocks or rotated valleys), so coordinate-wise scaling alone is insufficient. Adam's diagonal assumption can fail when parameters are correlated (requiring coordinated updates), when the loss landscape is rotated relative to parameter axes, or when saddle points require coordinated multi-parameter moves for escape. A canonical example is matrix factorization $A \approx UV^\top$, where gradients with respect to U depend on V and vice versa. Adam treats U and V independently, struggling to capture their coupling.

We propose FlowAdam, a hybrid optimizer that addresses this limitation through three mechanisms:

- Adaptive mode switching: EMA-based detection of plateaus and ill-conditioned curvature triggers transitions from Adam to ODE integration.
- Clipped ODE gradient flow: Solves a continuous-time gradient-flow ODE (with elementwise clipping for stability) using adaptive Runge-Kutta methods.
- Soft momentum injection: Blends ODE velocity with Adam's existing momentum when returning from ODE mode, preserving scale knowledge and preventing training collapse.

We evaluate FlowAdam on benchmarks spanning diverse coupling mechanisms: matrix completion ($UV^\top$ factorization), robust matrix factorization (Huber loss), tensor completion (trilinear coupling), GNN link prediction (topological coupling), inverse kinematics (trigonometric coupling), and real-world recommender data (Jester, MovieLens-100K). On problems where coupled optimization is the limiting factor, FlowAdam achieves 10–22% error reduction and outperforms tuned alternatives (Lion, AdaBelief) on real-world collaborative filtering. MovieLens-100K serves as a mechanism check: under a residualized protocol isolating the UV interaction term, FlowAdam matches Adam, confirming benefits arise specifically from navigating coupled geometry. The method matches Adam on well-conditioned workloads (CIFAR-10: 91.7±0.1%), confirming safe drop-in use. Source code is available at https://github.com/idevender/flowadam.

II. Related Work

A. Adaptive Optimization Methods

Adam [1] and its variants (AdaGrad [3], AdamW [4]) use per-parameter learning rate adaptation. AdamW introduced decoupled weight decay, applying regularization directly to parameters rather than through gradients. Recent work has analyzed Adam's implicit second-order properties [5], showing that the square-root in Adam's update relates to diagonal Fisher approximations; our work addresses coupling limitations complementary to this perspective. Other recent

optimizers—AdaBelief [6], RAdam [7], Lion [8], Shampoo [9], Sophia [10]—also operate in a purely discrete-time regime with diagonal or block-diagonal preconditioning, which may struggle on problems with dense, rotated parameter coupling.

Second-order methods such as L-BFGS [11], AdaHessian [12], and K-FAC [13] can capture parameter correlations but introduce significant memory/compute overhead (e.g., large per-layer factors, Hessian-vector products, or expensive inversions), making them less practical as drop-in replacements at scale. Our goal is to provide a first-order, $O(N)$-memory optimizer that captures coupling benefits without full Hessian access. Matrix completion has specialized solvers such as ALS [14]; FlowAdam offers a general-purpose alternative without problem-specific algorithmic design.

B. Continuous-Time Perspectives

Our approach builds upon the continuous-time view of optimization, where discrete algorithms are interpreted as discretizations of ODEs [15], [16]. Scieur et al. [17] explicitly frame gradient descent and accelerated methods as numerical integration schemes, while Dherin et al. [18] discuss training via ODE solvers and integration choices. Unlike Neural ODEs [19], which use ODE solvers for model architectures, FlowAdam uses continuous gradient flow as a trajectory-based velocity proposal to navigate ill-conditioned couplings that discrete steps struggle to resolve. Critically, we are not simply replacing Euler with Runge-Kutta; our contribution is triggered switching with momentum state reconciliation and gradient clipping. These continuous-time analyses are typically theoretical and do not yield practical optimizers with adaptive mode switching—which is our contribution.

C. Prior Hybrid Approaches

FlowAdam relates to Lookahead [20], which interpolates between synchronized checkpoints. However, FlowAdam blends continuous ODE velocity with discrete momentum using adaptive triggers. Prior attempts to combine global search (like ODEs) with local search often failed because the local optimizer's momentum buffer becomes invalid after a global step, causing training instability. Our Soft Momentum Injection addresses this gap by treating the ODE trajectory as a velocity vector that blends smoothly into the momentum buffer.

III. Method

A. Clipped Descent ODE Formulation

We model optimization as a dynamical system. Define elementwise gradient clipping as $[\mathrm{clip}_{\mathrm{grad}}(g)]_i = \mathrm{sign}(g_i)\min\{|g_i|, 1\}$, which clamps each coordinate to $[-1, 1]$. The clipped descent ODE is:

$$\frac{d\theta}{dt} = -\mathrm{clip}_{\mathrm{grad}}(\nabla L(\theta)) \tag{1}$$

Note this is not the true gradient flow due to clipping, but it still guarantees descent (proved in Proposition 1 below). Unlike discrete steps, ODE integration provides a more continuous trajectory through the local gradient vector field. All norms $\|\cdot\|$ denote the L_2 norm unless otherwise specified.

B. Adaptive Mode Switching

FlowAdam maintains EMA statistics of gradient behavior:

$$\bar{g}_t = \beta_{ema} \cdot \bar{g}_{t-1} + (1 - \beta_{ema}) \cdot \|\nabla L_t\| \tag{2}$$

$$\bar{c}_t = \beta_{ema} \cdot \bar{c}_{t-1} + (1 - \beta_{ema}) \cdot \|\nabla L_t - \nabla L_{t-1}\| \tag{3}$$

where $\nabla L_t \triangleq \nabla L(\theta_t)$ and $\beta_{ema} = 0.9$ is the EMA decay factor (distinct from Adam's β_1, β_2), $\bar{g}_t$ tracks average gradient norm, and $\bar{c}_t$ measures step-to-step gradient variation—a lightweight trigger proxy that combines signals from curvature, step size, and noise (direct curvature estimation would require Hessian-vector products, outside our first-order goal). ODE mode triggers when:

$$\underbrace{\|\nabla L_t\| < \alpha_s \cdot \bar{g}_t}_{\text{Plateau detected}} \quad \vee \quad \underbrace{\|\nabla L_t - \nabla L_{t-1}\| > \alpha_c \cdot \bar{c}_t}_{\text{High gradient variation}} \tag{4}$$

where α_s (switch sensitivity) and α_c (variation sensitivity) are hyperparameters. In highly stochastic regimes, $\|\nabla L_t - \nabla L_{t-1}\|$ is noise-dominated, so we use this trigger primarily for low-noise/full-batch problems. This approach is scale-invariant, adaptive to problem-specific gradient scales, and noise-smoothed via EMA averaging. In highly stochastic regimes, variance-reduction techniques (e.g., SVRG-style gradient corrections) could further improve trigger reliability; we leave this exploration to future work.

C. ODE Integration

When triggered, we integrate (1) using `dopri5`, a Dormand–Prince RK4/5 method:

$$\theta_{new} = \mathrm{ODESolve}\left(\frac{d\theta}{dt} = -\mathrm{clip}_{\mathrm{grad}}(\nabla L), \theta_{old}, [0, \alpha \cdot \tau]\right) \tag{5}$$

where α is the learning rate and τ is the ODE time scale. We use tolerance 10^{-4} for both relative and absolute tolerances. The adaptive RK integration provides local error control and automatically reduces internal step size in rapidly changing regions. If integration fails (rare), we fall back to a standard Adam step.

D. Soft Momentum Injection

Naive hybrid approaches reset Adam's momentum after ODE steps, causing state mismatch—Adam loses learned gradient scale information and training collapses. Our solution blends ODE velocity with existing momentum:

$$m_t \leftarrow (1 - \gamma) \cdot m_t + \gamma \cdot \underbrace{\frac{\theta_{old} - \theta_{new}}{\alpha}}_{\text{ODE velocity}} \tag{6}$$

where $\gamma \in [0, 1]$ is the injection weight on ODE velocity. In practice, we apply velocity clipping (7) before injection. Setting $\gamma = 0$ recovers pure Adam momentum with no injection, while $\gamma = 1$ uses full ODE velocity but may lose Adam stability. The default value $\gamma = 0.5$ provides balanced

Algorithm 1 FlowAdam Optimizer

Require: Learning rate α, EMA decay β_{ema}, sensitivities α_s, α_c, injection weight γ, ODE time scale τ, warmup steps t_{warmup}

Initialize $m_0 = 0$, $v_0 = 0$, $\bar{g}_0 = 0$, $\bar{c}_0 = 0$, $g_{\text{prev}} = 0$
for $t = 1, 2, \ldots$ **do**
  $g_t \leftarrow \nabla L(\theta_t)$
  Update EMAs: $\bar{g}_t$, $\bar{c}_t$
  plateau $\leftarrow \|g_t\| < \alpha_s \cdot \bar{g}_t$
  grad_change $\leftarrow \|g_t - g_{\text{prev}}\| > \alpha_c \cdot \bar{c}_t$
  if (plateau $\vee$ grad_change) **and** $t > t_{warmup}$ **then**
    $\theta_{new} \leftarrow \text{ODESolve}(-\text{clip}_{\text{grad}}(\nabla L), \theta_t, \alpha \cdot \tau)$
    $v_{ode} \leftarrow (\theta_t - \theta_{new})/\alpha$
    $m_t \leftarrow (1-\gamma)m_t + \gamma \cdot \text{clip}_{\text{vel}}(v_{ode})$ {Soft inj.}
    $\theta_{t+1} \leftarrow \theta_{new}$ {v_t, step counter unchanged}
  else
    Standard Adam update (increments per-parameter step counter)
  end if
  $g_{\text{prev}} \leftarrow g_t$
end for

blending. Note that v_{ode} is τ-scaled (since integration spans time $\alpha \cdot \tau$), so τ controls how strongly an ODE segment influences momentum. We do not update v_t or the Adam step counter on ODE steps and apply velocity clipping to prevent ODE velocity from dominating during hand-off:

$$\text{clip}_{\text{vel}}(v_{ode}) = \begin{cases} v_{ode} & \text{if } \|v_{ode}\| \le 5 \cdot \|m_t\| \\ v_{ode} \cdot \frac{5 \cdot \|m_t\|}{\|v_{ode}\|} & \text{otherwise} \end{cases} \quad (7)$$

The clipping factor of 5 was selected via validation and is robust to factors in $[3, 10]$. This enables smooth transitions while preserving Adam's scale knowledge.

Warmup. We disable ODE triggering during an initial warmup period ($t \le t_{warmup}$, default $t_{warmup} = 10$). This serves two purposes. First, it allows Adam's momentum m_t to accumulate meaningful gradient statistics before velocity clipping (7) is applied, since clipping relative to $\|m_t\|$ is uninformative when $m_t \approx 0$. Second, it avoids false ODE triggers before the EMA statistics $\bar{g}_t, \bar{c}_t$ have stabilized. The warmup period $t_{warmup} = 10$ is robust across $[5, 20]$ in our experiments. The complete procedure is summarized in Algorithm 1.

E. Hyperparameter Guidelines

We define two operating modes. **Mode A** (neural networks, stochastic) uses $\alpha_s = 0.4$, $\alpha_c = 3.0$, $\tau = 2.0$ for conservative triggering. **Mode B** (scientific ML, deterministic) uses $\alpha_s = 0.9$, $\alpha_c = 0.1$, $\tau = 0.5$ for aggressive triggering. This parallels Adam's default β_1, β_2 values—sensible defaults with problem-specific tuning when needed. Automatic mode selection based on estimated gradient noise level is a natural extension.

F. Theoretical Properties

We provide two formal results justifying the core mechanisms of FlowAdam.

Proposition 1 (Monotonic Descent for Clipped Gradient Flow). Let $L \in C^1(\mathbb{R}^d)$ and define elementwise clipping $C : \mathbb{R}^d \to \mathbb{R}^d$ by $[C(g)]_i = \text{sign}(g_i) \min\{|g_i|, 1\}$. Consider the clipped gradient flow ODE (1). Then along any differentiable trajectory $\theta(t)$:

$$\frac{d}{dt} L(\theta(t)) = -\nabla L(\theta(t))^\top C(\nabla L(\theta(t))) \le 0 \quad (8)$$

Proof. For each coordinate i, we have $\partial_i L \cdot [C(\nabla L)]_i = |\partial_i L| \min\{|\partial_i L|, 1\} \ge 0$ since clipping preserves sign. Summing over coordinates yields $\nabla L^\top C(\nabla L) \ge 0$, hence $\frac{d}{dt} L(\theta(t)) = -\nabla L^\top C(\nabla L) \le 0$. □

Note that clipping changes the vector field and we do not claim steepest descent; however, near critical points where $\|\nabla L\|_\infty < 1$, clipping is inactive and standard gradient flow is recovered. The continuous-time flow is monotone; numerically, we observe non-increase up to solver tolerances in our experiments (rare violations may occur due to discretization).

Lemma 2 (Soft Injection Bound). If the momentum update is a convex blend $m^+ = (1-\gamma)m + \gamma\tilde{v}$ with $\gamma \in [0, 1]$, then:

$$\|m^+\| \le (1-\gamma)\|m\| + \gamma\|\tilde{v}\| \le \max\{\|m\|, \|\tilde{v}\|\} \quad (9)$$

Proof. Triangle inequality gives $\|m^+\| \le (1-\gamma)\|m\| + \gamma\|\tilde{v}\|$. Since $(1-\gamma)+\gamma = 1$, this convex combination satisfies $(1-\gamma)\|m\| + \gamma\|\tilde{v}\| \le \max\{\|m\|, \|\tilde{v}\|\}$. □

This bounds the momentum magnitude after injection: if $\tilde{v}$ (the ODE velocity) is additionally clipped by construction (7), then $\|m^+\|$ is bounded by the clipping threshold, preventing momentum explosion during mode transitions. These component-level guarantees do not constitute a convergence proof for the full hybrid switching system; formal convergence analysis of the alternating Adam–ODE dynamics remains future work.

IV. Experiments

We evaluate FlowAdam on benchmarks organized into three groups: (A) mechanism validation, (B) conditioning and robustness, and (C) core challenge tasks demonstrating performance on coupled parameter problems. All experiments use PyTorch with `torchdiffeq` for ODE integration and standard bias-corrected Adam/AdamW implementations. We compare against Adam (and AdamW where relevant); SGD+momentum is included as a sanity baseline where applicable, along with problem-specific baselines. Each experiment reports mean ± std over multiple seeds unless noted.

Experimental Protocol. We use 5 seeds for all experiments. All matrix/tensor completion and robust matrix factorization experiments use full-batch (deterministic) optimization. "Improv." denotes relative improvement vs. Adam: for RMSE, $(\text{Adam} - \text{FlowAdam})/\text{Adam} \times 100\%$; for AUC, $(\text{FlowAdam} - \text{Adam})/\text{Adam} \times 100\%$. Wall-clock times include ODE solver overhead. We report total gradient evaluations for compute-fair comparison: for Adam, total grad evals = number of steps (one backward per step); for FlowAdam, total grad evals = outer steps + ODE NFE (number of function evaluations), since each

`ode_func` call triggers one backward pass. With `dopri5` at tolerance 10^{-4}, each ODE trigger typically uses 8–12 NFE.

To ensure the ODE solver integrates the regularized landscape, FlowAdam applies regularization directly to the loss ($\lambda\|\theta\|^2$). When AdamW is included, we tune its weight decay across $\{10^{-5}, 10^{-3}, 10^{-2}\}$. To control for regularization implementation differences, we additionally report "Adam (explicit L2)", which uses the same loss-based regularization as FlowAdam: $L = \text{MSE} + \lambda \cdot (\|U\|^2 + \|V\|^2)$ with $\lambda = 10^{-5}$. This isolates the optimizer effect from regularization implementation.

Evaluation Philosophy. Our experiments serve four purposes: (1) *Drop-in evaluation*—comparing default FlowAdam against default Adam/AdamW to assess practical adoption without tuning (Tables I–VI); (2) *Alternative strategies*—comparing tuned FlowAdam against fundamentally different optimization approaches (Table VII); (3) *Conditioning stress-tests*—verifying behavior under ill-conditioning and confirming no regression on well-conditioned tasks (Section IV.B); (4) *Mechanism checks*—toy diagnostics/ablations (Section IV.A) and residualized MovieLens (Section IV.C.7).

A. Mechanism Validation

1) Rosenbrock Function: The Rosenbrock function $f(x, y) = (1-x)^2 + 100(y-x^2)^2$ has a curved valley testing optimizer navigation. Starting at $(-1.5, 1.5)$ over 500 steps, FlowAdam (Mode B) achieves 12% lower final loss than Adam with 50 ODE triggers concentrated at the valley turn.

2) Two Spirals Classification: Two interleaved spirals (1200° rotation) test neural network optimization. Using 1000 points with a 3-layer MLP (24 hidden units, Tanh activation) over 4000 steps (Mode A), FlowAdam achieves 100% accuracy compared to Adam's 100% and SGD's 54.4%. FlowAdam matches Adam on this task where adaptive methods excel, confirming no regression on well-conditioned neural network problems. (SGD's poor performance reflects sensitivity to per-parameter adaptation.)

3) Ablation: Soft vs. Hard Replacement: Comparing soft momentum injection (6) against hard replacement ($m_t \leftarrow v_{\text{ode}}$) on two spirals reveals the importance of our approach. Hard replacement undergoes a late-stage collapse (down to 67%) and ends at 82.5%, while soft injection recovers to 100.0% with fewer ODE triggers (221 for Hard vs. 166 for Soft). Figure 1 illustrates this state mismatch problem and its resolution. Together with the compute-matched comparison in Section IV.C.6 (Table VI), this partially disentangles improvement sources: the compute-matched experiment isolates the ODE trajectory's contribution, while the present ablation isolates the injection mechanism.

B. Conditioning and Robustness

1) Rotated Stiff Valley (Ill-Conditioned Quadratic): A 50-dimensional quadratic with random rotation and condition number 2000 directly tests Adam's diagonal assumption. We construct $L(\theta) = \frac{1}{2}\theta^\top H \theta$ where $H = Q \cdot \text{diag}(2000, 2000, 1, \ldots, 1) \cdot Q^\top$ and optimize over 500 steps

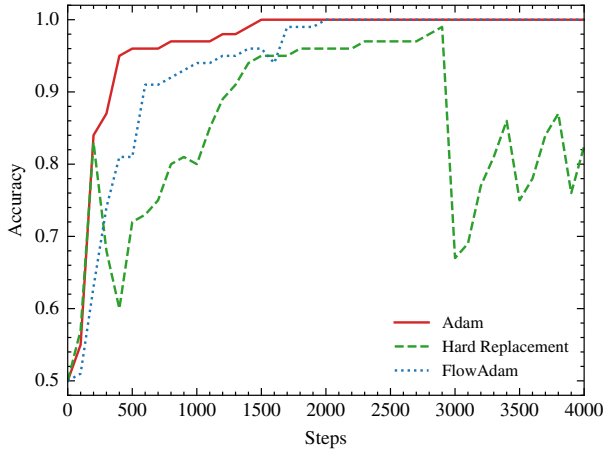

Fig. 1. Ablation: Hard replacement (green) induces a state mismatch by overwriting Adam's momentum buffer, causing a late-stage instability (final accuracy 82.5%). FlowAdam with soft injection (blue) blends the ODE velocity with the existing momentum, avoiding the mismatch and recovering stable convergence (100.0%).

(Mode B). FlowAdam achieves final loss 36.0 compared to Adam's 97.0 and SGD's 156.2, yielding 2.7× lower loss than Adam. The high ODE trigger rate (96%) indicates correct detection of the ill-conditioned landscape.

2) CIFAR-10 Regression Test: To verify FlowAdam does not regress on well-conditioned problems, we evaluate on CIFAR-10 with ResNet-18 (50 epochs, batch 128, standard augmentation, no learning rate scheduling, Mode A). FlowAdam achieves $91.7 \pm 0.1\%$ test accuracy compared to Adam's $91.8 \pm 0.1\%$ with negligible ODE triggers (≈ 0), confirming it is safe where Adam already works well. The near-zero trigger rate indicates that Mode A's conservative thresholds effectively deactivate ODE integration on this well-conditioned stochastic task; extending the trigger to activate beneficially under mini-batch noise is an open direction.

C. Coupled Parameter Benchmarks

1) Matrix Completion: Matrix completion—reconstructing a partially observed matrix via $A \approx UV^\top$—creates dense parameter coupling. We test three scenarios with increasing difficulty: Small Dense (200×300, rank=10, 30% observed), Medium Moderate (300×400, rank=15, 20% observed), and Larger Sparse (400×500, rank=20, 15% observed). We set model rank to true rank + 5 and train for 1000 full-batch steps. We evaluate RMSE on held-out entries (those not in the observation mask) using the known synthetic ground truth.

Results. FlowAdam achieves 10.2–21.8% lower test RMSE than Adam across all scenarios (Table I), with improvement that scales with problem difficulty (Figure 2, illustrating the implicit regularization effect).

Comparison with AdamW. We swept AdamW weight decay $\lambda \in \{10^{-5}, 10^{-3}, 10^{-2}\}$ (Table I). AdamW is insensitive to λ (RMSE varies $< 1\%$) and consistently underperforms Adam. A joint LR×λ sweep on Medium (30 configs) con-

TABLE I
MATRIX COMPLETION: TEST RMSE (5 SEEDS). MODE B.

Scenario	Adam	AdamW*	FlowAdam	Improv.
Small Dense	0.098 ± 0.001	0.115 ± 0.003	$\mathbf{0.088 \pm 0.001}$	10.2%
Medium Mod.	0.129 ± 0.001	0.181 ± 0.004	$\mathbf{0.111 \pm 0.001}$	14.0%
Larger Sparse	0.303 ± 0.005	0.640 ± 0.015	$\mathbf{0.237 \pm 0.001}$	21.8%

*Best AdamW across $\lambda \in \{10^{-5}, 10^{-3}, 10^{-2}\}$. Improv. vs. Adam.

TABLE II
ROBUST MATRIX FACTORIZATION: TEST RMSE (5 SEEDS). MODE B.

Scenario	Adam	FlowAdam	Improv.
Small Heavy	0.736 ± 0.053	$\mathbf{0.619 \pm 0.031}$	15.9%
Medium Heavy	0.955 ± 0.020	$\mathbf{0.822 \pm 0.016}$	13.9%
Large Heavy	0.938 ± 0.054	$\mathbf{0.819 \pm 0.014}$	12.7%

TABLE III
TENSOR COMPLETION: TEST RMSE (5 SEEDS). MODE B.

Scenario	Adam	AdamW	FlowAdam	Improv.
Small Sparse	0.071 ± 0.002	0.083 ± 0.008	$\mathbf{0.063 \pm 0.001}$	11.3%
Medium Sparse	0.068 ± 0.002	0.105 ± 0.027	$\mathbf{0.060 \pm 0.001}$	11.8%
Larger Sparse	0.055 ± 0.001	0.067 ± 0.005	$\mathbf{0.049 \pm 0.001}$	10.9%

TABLE IV
GNN LINK PREDICTION: TEST AUC (5 SEEDS). MODE B.

Scenario	Adam	FlowAdam	Improv.
Medium Strong	0.770 ± 0.008	$\mathbf{0.793 \pm 0.009}$	3.0%
Large Moderate	0.730 ± 0.009	$\mathbf{0.758 \pm 0.014}$	3.8%
Larger Challenge	0.698 ± 0.013	$\mathbf{0.723 \pm 0.011}$	3.6%
Easy Calib.	0.941 ± 0.003	0.942 ± 0.003	–

Calibration uses high-signal settings to verify the pipeline reaches high AUC.

firms this, with AdamW's best RMSE still 13% worse than FlowAdam.

AdamW's underperformance on matrix/tensor completion—yet parity with Adam on inverse kinematics (Table V)—suggests decoupled weight decay may interfere with implicit regularization dynamics specific to low-rank recovery, where the optimization trajectory itself induces bias toward simple solutions.

Regularization control. Adam with identical loss-based L2 regularization (explicit L2) achieves RMSE 0.116 ± 0.001 on Medium versus FlowAdam's 0.111 ± 0.001—a 4.3% improvement confirming gains arise from ODE integration.

2) Robust Matrix Factorization: Robust matrix factorization recovers a low-rank approximation $M \approx UV^\top$ from observations corrupted by heavy outliers. We test three scenarios with 20% heavy outliers (magnitude 4–5× signal): Small (60×80, rank=5), Medium (80×100, rank=8), and Large (100×120, rank=10). The loss uses Huber regression between observations and the low-rank reconstruction, which is robust to outliers.

FlowAdam achieves 12.7–15.9% lower reconstruction error across all scenarios (Table II). The dense coupling between U and V factor matrices creates correlated parameter updates where ODE integration excels, with ∼245 ODE triggers per run.

3) Tensor Completion: Tensor completion extends matrix factorization to 3D: $\mathcal{T} \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r$, creating trilinear coupling across three factor matrices. We test Small Sparse ($30 \times 40 \times 50$, rank=5, 10% observed), Medium Sparse ($40 \times 50 \times 60$, rank=8, 8% observed), and Larger Sparse ($50 \times 60 \times 70$, rank=10, 8% observed).

FlowAdam achieves 10.9–11.8% improvement across all scenarios (Table III). The 3D coupling amplifies correlated gradients beyond 2D matrices, with ∼498 ODE triggers demonstrating aggressive landscape exploration.

4) GNN Link Prediction: Graph Neural Networks provide a test case where coupling arises from topological structure rather than matrix factorization. Using a 2-layer GCN with hidden dimension 32, we predict 20% held-out edges on synthetic graphs. We test Medium Strong (500 nodes, degree=20), Large Moderate (600 nodes, degree=18), and Larger Challenge (800 nodes, degree=15).

FlowAdam achieves 3.0–3.8% AUC improvement across the main scenarios (Table IV), demonstrating effectiveness beyond matrix-structured problems. The topological coupling from message passing creates correlated gradients that benefit from ODE integration (averaging 160–175 triggers per run). We use Mode B (full-batch) on these moderate-sized synthetic graphs; for large-scale GNNs with mini-batch sampling, Mode A would be appropriate. The calibration run confirms the model can reach high AUC (> 0.94) when signal is abundant, ruling out capacity limitations.

5) Inverse Kinematics: Multi-target inverse kinematics tests purely trigonometric coupling. For an n-link planar robot, end-effector position involves nested trigonometric functions—not bilinear $UV^\top$ structure. We optimize a trajectory of 10 waypoints for an 8-link arm with smoothness penalty $\lambda = 1.0$ coupling all configurations (1500 steps).

FlowAdam outperforms on 4/5 trajectory instances (Table V), achieving 20.9% mean improvement in target RMSE (7.2% median). The trigonometric forward kinematics creates a highly non-convex landscape with many local minima; on one trajectory instance, FlowAdam escapes a poor local minimum that traps all baselines (RMSE 0.016 vs. 0.192). This validates that benefits extend to geometric/trigonometric coupling.

6) Jester: Real-World Matrix Completion: Jester [21] is a classic collaborative filtering benchmark. We evaluate on Subset 1, which contains the most active users (24,983 users who rated 36+ jokes), resulting in 1.81 million ratings on 100 jokes (scale $[-10, 10]$). Unlike MovieLens-100K, Jester's small item count (100 jokes vs. 1,682 movies) reduces item bias variance, creating a setting where $UV^\top$ interaction learning dominates. We evaluate on held-out observed ratings, following standard practice for real collaborative filtering benchmarks without full ground truth.

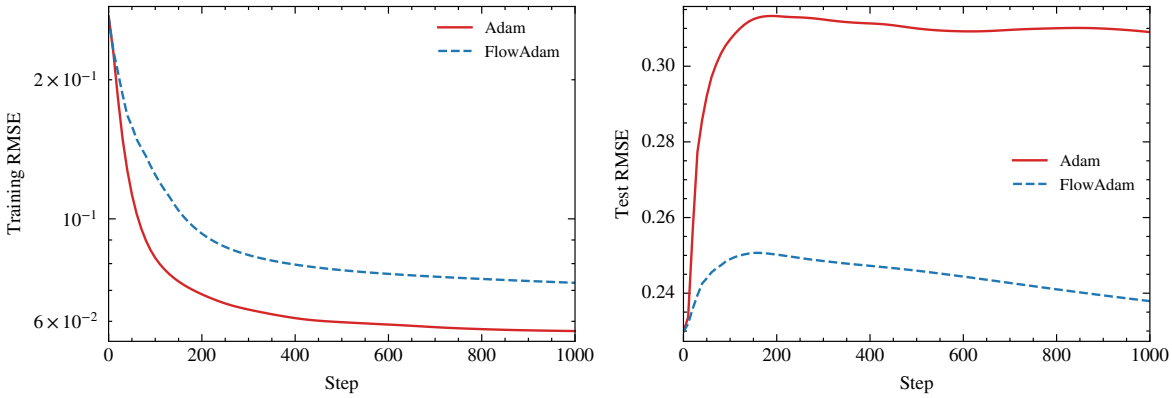

Fig. 2. Matrix Completion: Larger Sparse scenario (400×500, rank=20, 15% observed). Left: Training RMSE (log scale). Right: Test RMSE (linear scale). Representative single-seed run. Despite higher training RMSE (left), FlowAdam achieves 23.0% lower test RMSE (right; 5-seed mean: 21.8%, Table I), demonstrating implicit regularization.

TABLE V
INVERSE KINEMATICS: TARGET RMSE (5 SEEDS). MODE B.

Optimizer	Target RMSE	Median	Improv.
Adam	0.182 ± 0.080	0.208	–
AdamW	0.182 ± 0.080	0.208	–
FlowAdam	$\mathbf{0.144 \pm 0.070}$	**0.193**	**20.9%**

TABLE VI
JESTER: TEST RMSE (5 SEEDS). MODE B.

Optimizer	Test RMSE	Improv.
Adam (1K steps)	4.731 ± 0.014	–
Adam Ext (4,350 grad evals)	5.007 ± 0.019	−5.8%
FlowAdam (1K steps)	$\mathbf{4.427 \pm 0.009}$	**6.4%**

FlowAdam achieves 6.4% lower RMSE than Adam (Table VI), with improvement consistent across all 5 seeds (range 6.1–6.7%). Notably, even when Adam's training is extended to match FlowAdam's compute budget (Adam Extended in Table VI), it fails to close the performance gap—instead overfitting rather than converging to a better solution. This indicates that FlowAdam's advantage stems from the implicit regularization of the ODE trajectory, not merely additional gradient evaluations. The compute-matched comparison shows FlowAdam is 11.6% better than Adam Extended with identical compute budget. The ODE triggering rate of 49.5% (495/1000 steps) indicates a challenging landscape where coupled optimization matters.

Comparison with alternative optimization strategies. Tables I–VI establish FlowAdam's advantage over Adam under the drop-in protocol. Table VII asks a different question. Does FlowAdam remain competitive against methods employing fundamentally different optimization strategies? We compare against Lion [8] (sign-based updates), AdaBelief [6] (belief-scaled variance), and L-BFGS (quasi-Newton curvature approximation). *Note: Different tuning protocols were used, so absolute RMSE values should not be compared across tables.*

TABLE VII
COMPARISON WITH ALTERNATIVE OPTIMIZATION STRATEGIES (JESTER). ALL METHODS TUNED VIA GRID SEARCH (5 SEEDS).

Optimizer	Test RMSE	Std	vs. FlowAdam
Lion	4.271	0.008	+2.0%
AdaBelief	4.495	0.022	+7.4%
L-BFGS	5.083	0.017	+21.4%
FlowAdam	**4.186**	**0.013**	–

Tuned FlowAdam (4.19 RMSE) outperforms tuned Lion (4.27) by 2.0% and tuned AdaBelief (4.49) by 7.4%. L-BFGS performs poorly on this large-scale problem (1.81M ratings), likely due to difficulty approximating curvature with limited history vectors. Lion did not close the gap with extended training. FlowAdam wins under *both* protocols—untuned (6.4% vs. Adam) and tuned (2.0% vs. Lion)—demonstrating that gains arise from optimizer dynamics rather than Adam's limitations.

7) MovieLens-100K (Residualized): MovieLens-100K [22] serves as a *mechanism check* rather than a headline gain. This dataset contains 100,000 ratings from 943 users on 1,682 movies. To directly test whether FlowAdam helps coupled UV optimization, we employ a **residualized protocol**. First, we fit user/item biases ($\mu + b_u + b_i$) using Adam with early stopping. Then we compute residuals $e = r - (\mu + b_u + b_i)$. Finally, we train only the interaction term $UV^\top$ on residual targets.

MovieLens-100K (residualized protocol, Mode B, 5 seeds) shows that bias-only RMSE is 0.945 ± 0.004 using Adam. On residual UV targets, FlowAdam matches Adam (0.949 ± 0.005 vs. 0.949 ± 0.005), indicating no regression on a near-separable setting. This is *consistent with our hypothesis*. FlowAdam's benefit is largest when coupled optimization is the limiting factor. On MovieLens, user/item biases explain most variance, leaving the residual UV component near the noise floor with little room for improvement.

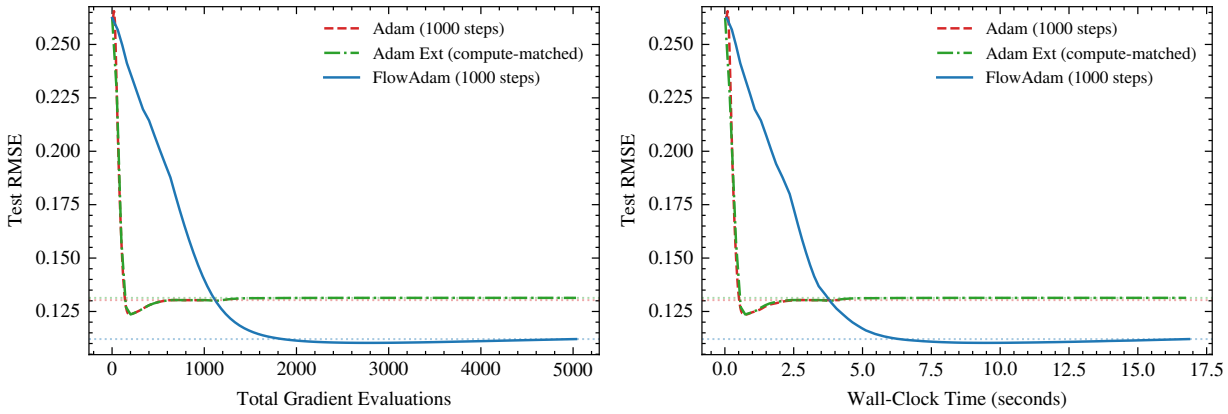

Fig. 3. Compute-matched comparison on Medium Matrix Completion. Left: RMSE vs. gradient evaluations. Right: RMSE vs. wall-clock time. Horizontal dotted lines indicate converged test RMSE.

V. Discussion

A. Applicability and Scope

FlowAdam benefits full-batch or large-batch training with dense parameter correlations (e.g., matrix factorization) and ill-conditioned curvature, while providing no benefit (but no harm) when training is highly stochastic or Adam already converges well. For drop-in evaluation, FlowAdam and Adam use identical default learning rates ($\alpha = 0.001$), ensuring performance differences reflect optimizer dynamics. FlowAdam is intended as a drop-in, $O(N)$-memory replacement for Adam rather than a competitor to full second-order methods.

Practitioner guidelines. Use Mode B for full-batch or large-batch training with low gradient noise, matrix/tensor factorization, and problems with dense off-diagonal Hessian structure. Use Mode A for standard stochastic neural network training. Avoid when mini-batch noise would dominate ODE triggers.

B. Computational Overhead

ODE integration adds 8–12 gradient evaluations per trigger. On CIFAR-10 (Mode A), triggers ≈ 0 (no overhead). On matrix completion, $\sim$500 triggers provide implicit regularization. To verify gains are not from extra compute, we compared FlowAdam (1,000 steps, 5,032 grad evals) against Adam Extended (5,032 steps) on Medium Matrix Completion. FlowAdam achieves 14.7% lower RMSE (Figure 3), demonstrating the advantage comes from accessing better optima, not additional compute.

C. Implicit Regularization

The observation that frequent ODE triggers improve generalization despite higher training loss aligns with theoretical findings on implicit regularization [23], [24]. Gradient descent trajectories in matrix factorization induce implicit bias toward low-rank solutions. FlowAdam enriches these dynamics by integrating continuous gradient flow, exploring coupled parameter geometry more effectively than discrete steps.

D. Sensitivity Analysis

The switch sensitivity α_s controls ODE frequency. Performance is stable for $\alpha_s \in [0.675, 0.90]$ with $<1\%$ RMSE variation. However, $\alpha_s > 1$ causes plateau detection on nearly every step, collapsing the hybrid into always-ODE mode with degraded performance. We recommend $\alpha_s \leq 1$.

For the injection weight γ (weight on ODE velocity during momentum blending), all values in $[0.1, 0.9]$ achieve test RMSE within $\sim$2% of the optimal (Figure 4, left), demonstrating that $\gamma = 0.5$ is a robust default. Additionally, we verified robustness across regularization configurations. FlowAdam retained a $\sim$7% advantage even when Adam was given $2\times$ regularization (Figure 4, right).

E. Limitations

Several limitations warrant discussion. First, the EMA-based triggering mechanism is sensitive to mini-batch noise: the gradient variation criterion $\|\nabla L_t - \nabla L_{t-1}\|$ conflates true curvature signals with stochastic variance, restricting reliable ODE triggering to low-noise or full-batch settings. Second, the hybrid switching system lacks formal convergence guarantees; our theoretical results (Proposition 1, Lemma 2) bound individual components but do not prove convergence of the alternating Adam–ODE dynamics. Third, ODE integration adds 8–12 gradient evaluations per trigger, which may be prohibitive for very large-scale architectures (e.g., Transformers) unless the trigger rate remains low. Fourth, users must manually select between Mode A and Mode B; automatic mode adaptation based on runtime noise estimation is a natural but unimplemented extension. Finally, our evaluation is limited to small- and medium-scale benchmarks; scaling behavior on modern large-scale training remains to be established.

VI. Conclusion

We presented FlowAdam, a hybrid optimizer combining Adam with ODE-based gradient flow. The key contribution, soft momentum injection, enables smooth mode transitions without destabilizing training. Across benchmarks span-

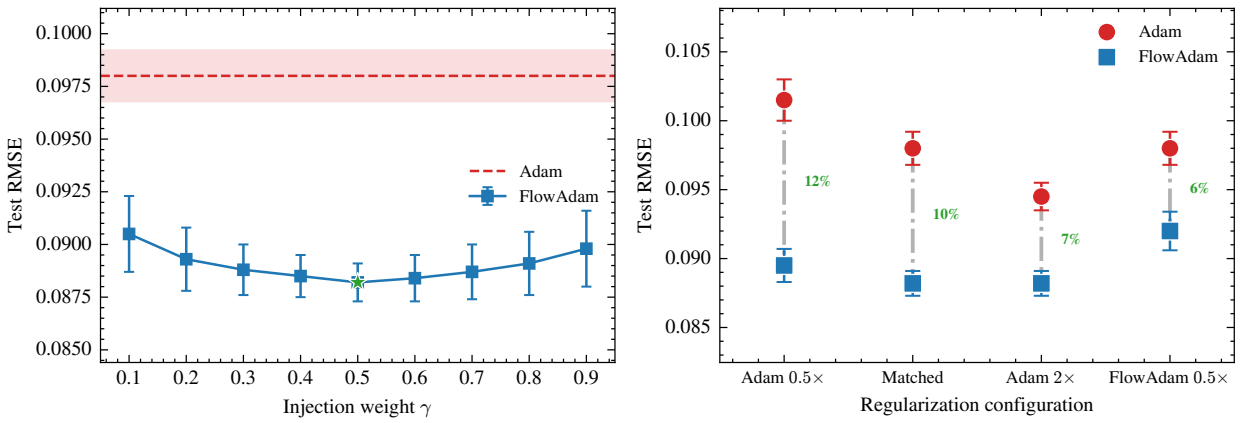

Fig. 4. Sensitivity analysis on matrix completion (Mode B, 5 seeds). Left: Performance is robust across $\gamma \in [0.1, 0.9]$; asterisk marks default $\gamma = 0.5$. Right: FlowAdam (fixed $\lambda = 10^{-5}$) vs. Adam across regularization strengths. The x-axis shows regularization configurations: Adam at $0.5\times$, $1\times$ (matched), and $2\times$ the regularization strength of FlowAdam, plus FlowAdam at $0.5\times$ regularization. FlowAdam outperforms Adam at all tested configurations.

ning algebraic coupling (matrix/tensor factorization), architectural coupling (GNN), and geometric coupling (robotics), FlowAdam achieves 10–22% error reduction on matrix/tensor recovery, 3–4% on GNN, and 6% on real-world collaborative filtering (surpassing tuned alternatives including Lion) when coupled optimization is limiting. MovieLens residualized protocol confirms benefits arise specifically from navigating coupled geometry.

A notable finding is that aggressive ODE triggering provides implicit regularization, improving generalization with minimal explicit weight decay. Future directions include convergence analysis, adaptive γ scheduling, and distributed training. More broadly, FlowAdam demonstrates that incorporating continuous-time dynamics into discrete optimizers can complement recent geometry-aware approaches to optimization.

References

[1] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[2] A. C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht, "The marginal value of adaptive gradient methods in machine learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[3] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[4] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations (ICLR)*, 2019.

[5] S. Malladi, K. Lyu, A. Panigrahi, and S. Arora, "On the SDEs and scaling rules for adaptive gradient algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[6] J. Zhuang, T. Tang, Y. Ding, S. C. Tatikonda, N. Dvornek, X. Papademetris, and J. Duncan, "AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 18 795–18 806, 2020.

[7] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

[8] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 49 205–49 233, 2023.

[9] V. Gupta, T. Koren, and Y. Singer, "Shampoo: Preconditioned stochastic tensor optimization," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 1842–1850.

[10] H. Liu, Z. Li, D. Hall, P. Liang, and T. Ma, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," in *International Conference on Learning Representations (ICLR)*, 2024.

[11] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989.

[12] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "AdaHessian: An adaptive second order optimizer for machine learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 10 665–10 673.

[13] J. Martens and R. Grosse, "Optimizing neural networks with Kronecker-factored approximate curvature," in *International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 2408–2417.

[14] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[15] W. Su, S. Boyd, and E. J. Candès, "A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights," *Journal of Machine Learning Research*, vol. 17, no. 153, pp. 1–43, 2016.

[16] A. Wibisono, A. C. Wilson, and M. I. Jordan, "A variational perspective on accelerated methods in optimization," *Proceedings of the National Academy of Sciences*, vol. 113, no. 47, pp. E7351–E7358, 2016.

[17] D. Scieur, V. Roulet, F. Bach, and A. d'Aspremont, "Integration methods and optimization algorithms," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[18] B. Dherin, M. Munn, H. Mazzawi, M. Wunder, S. Medapati, and J. Gonzalvo, "Learning by solving differential equations," *arXiv preprint arXiv:2505.13397*, 2025.

[19] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[20] M. R. Zhang, J. Lucas, G. Hinton, and J. Ba, "Lookahead optimizer: k steps forward, 1 step back," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[21] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[22] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, pp. 1–19, 2015.

[23] S. Gunasekar, B. E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro, "Implicit regularization in matrix factorization," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[24] S. Arora, N. Cohen, W. Hu, and Y. Luo, "Implicit regularization in deep matrix factorization," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.