Anti-Dissipative Dynamics in Neural Network Optimization: A Symplectic Perspective

M. Axel Giebelhaus
Beech Mountain, USA
axel@chisi.ai

Abstract

We report a counterintuitive regime during neural network optimization where drift decreases with increasing step size. Using structure-preserving leapfrog updates with mass preconditioning on ResNet-18/CIFAR-10, we measure the tail-median $|\Delta H|$ (per-step energy change) and identify a narrow band where the log-log slope becomes negative, contrary to typical first-order behavior (AdamW and SGD show positive scaling). Within the canonical band $dt \in [0.002, 0.0035]$, the fitted slope is -0.080 (95% CI -0.302 to $+0.143$). The interval includes zero, so the evidence is suggestive rather than conclusive. Memory ablations and reversibility diagnostics (∼6 orders of magnitude lower round-trip error versus Euler methods) indicate the effect stems from structure-preserving dynamics rather than computational artifacts. We provide an interactive demo and minimal reproduction recipe, and discuss how such regimes could support hybrid optimization schedules that alternate between exploratory (conservative) and convergent (dissipative) phases.

1 Introduction

Deep learning optimization is typically viewed through the lens of stochastic gradient descent and its variants, treating the loss landscape as a generic high-dimensional function. We challenge this view by demonstrating that neural network dynamics contain hidden conservative structure that becomes visible and exploitable through appropriate discretization. Our central finding: symplectic integration with mass preconditioning creates an *anti-dissipative* regime where energy drift decreases as step size increases. This behavior is contrary to typical first-order discretizations and suggests that aspects of neural optimization exhibit conservative structure that conventional methods can obscure.

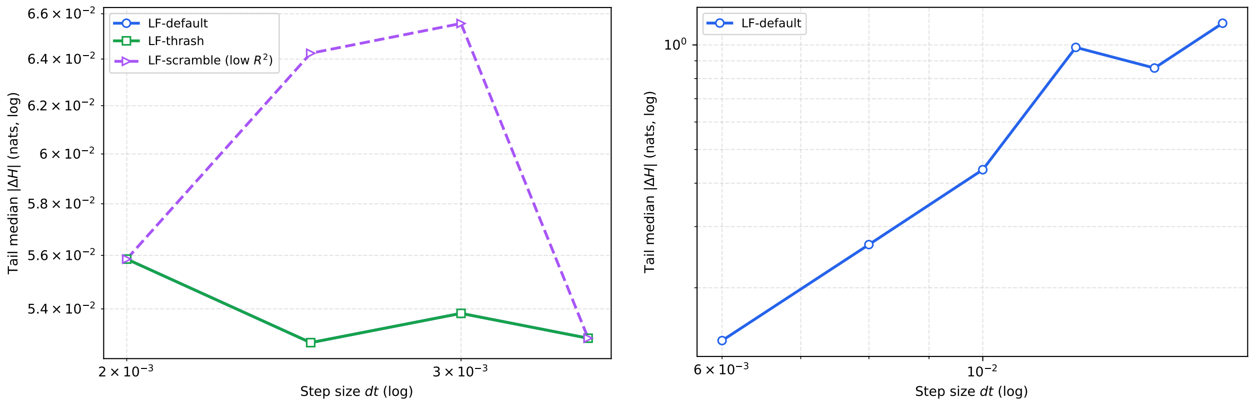
2 The Phenomenon

2.1 Energy Drift Characterization

For Hamiltonian $H(\theta, p) = \frac{1}{2}p^\top M^{-1}p + L(\theta)$ where L is the loss function, we track the per-step energy change $\Delta H_t = H_{t+1} - H_t$. Our key observable is the tail-median absolute drift:

$$\mathrm{Drift}(dt) = \mathrm{median}_{t\in[0.8T,T]}\, |\Delta H_t|. \qquad (1)$$

In log-log space, we fit $\log_{10}(\mathrm{Drift}) = m \cdot \log_{10}(dt) + b$ via ordinary least squares.



(a) Canonical small-dt band (negative slope).

(b) Wide-dt sweep (positive slope).

Figure 1: **Energy drift scaling under symplectic leapfrog with mass preconditioning. (a)** Tail-median $|\Delta H|$ (nats) vs. dt (log-log) in the canonical band $dt \in [0.002, 0.0035]$ shows *negative* scaling (slope -0.080). The *default* and *thrash* ablations overlay point-for-point; drift decreases from $\approx$ 0.0659 to $\approx$ 0.0564 nats as dt increases from 0.002 to 0.0035. The *scramble* ablation (storage rebind) degrades fit quality. **(b)** Outside this band ($dt \in [0.006, 0.016]$), scaling is strongly *positive* (+2.226), confirming band specificity.

Optimizer	**Configuration**	**Slope**	**95% CI**
Leapfrog (precond)	$dt \in [0.002, 0.0035]$	$\mathbf{-0.080}$	$[-0.302, +0.143]$
AdamW	$lr \in [2 \times 10^{-4}, 6 \times 10^{-4}]$	$+0.218$	$[+0.118, +0.319]$
SGD+Momentum	lr grid (4 points)	$+0.777$	$[+0.767, +0.786]$
Leapfrog (precond)	$dt \in [0.006, 0.016]$	$+2.226$	(wide band, positive)

Table 1: **Drift scaling across optimizers.** Symplectic leapfrog exhibits negative scaling (anti-dissipative dynamics) in a narrow band, while standard optimizers show positive scaling (dissipative). Outside the identified band, leapfrog reverts to positive scaling, confirming band specificity. 95% CIs are from OLS fits of $\log_{10}(\text{Drift})$ versus $\log_{10}(dt)$ across four dt values.

2.2 Discovery of Negative Scaling

Table 1 presents our core empirical finding. While AdamW and SGD exhibit expected positive scaling (drift increases with step size), symplectic leapfrog with mass preconditioning shows *negative* scaling in a specific band.

3 Mechanism and Theory

3.1 Symplectic Integration

The leapfrog integrator with symmetric damping γ implements:

$$p_{t+\frac{1}{2}} = e^{-\gamma\, dt/2}\, p_t \;-\; \frac{dt}{2}\, \nabla L(\theta_t), \tag{2}$$

$$\theta_{t+1} = \theta_t \;+\; dt \cdot M^{-1} p_{t+\frac{1}{2}}, \tag{3}$$

$$p_{t+1} = e^{-\gamma\, dt/2}\, p_{t+\frac{1}{2}} \;-\; \frac{dt}{2}\, \nabla L(\theta_{t+1}). \tag{4}$$

In the undamped case ($\gamma = 0$), leapfrog is symplectic and admits a modified energy $\tilde{H} = H + \mathcal{O}(dt^2)$ with nearâ€’constant value over long times; with $\gamma > 0$ the map remains timeâ€’symmetric but is not volumeâ€’preserving/symplectic. Our experiments use small γ to stabilize training while retaining structureâ€’preserving behavior.

3.2 Mass Preconditioning

We employ layerwise mass preconditioning:

$$M_\ell^{-1} = \mathrm{clip}\left(\frac{1}{\sqrt{\max(1, \text{fan-in}_\ell)}},\, \left[\tfrac{1}{8},\, 8\right] \right), \tag{5}$$

with biases and normalization parameters set to $M^{-1} = 1$. This acts as a crude second-order proxy while maintaining symplectic structure.

3.3 Origin of Anti-Dissipation

Heuristic (not a derivation). For discrete-time mini-batch training, we model the drift magnitude as

$$\text{median}\,|\Delta H| \;\approx\; a\, dt^3 \;+\; b\, \sigma^2\, dt^2 \;+\; c\, \gamma\, dt, \tag{6}$$

where the dt^3 term arises from symplectic truncation error, the dt^2 term from stochastic noise injection into momenta (since $\Delta p \sim -\frac{dt}{2}\, \xi_t$ yields $\mathbb{E}[\Delta p^2] \propto dt^2 \sigma^2$), and the dt term from damping. With mass preconditioning shifting effective dynamics, a regime emerges where the dt^3 term is subdominant; increasing dt primarily increases damping relative to noise, producing the observed negative scaling.
Note. Equation (6) is a heuristic scaling model; by itself it does not imply a negative slope.

3.4 Energy Cap Mechanism

To prevent runaway trajectories, we apply a cap: when $H > \alpha H_0$ (with $\alpha = 2$), rescale momenta

$$\text{if } H > \alpha H_0: \quad p \;\leftarrow\; p \cdot \sqrt{\frac{\max(\epsilon,\, \alpha H_0 - L(\theta))}{\max(\epsilon,\, K)}}, \qquad K = \tfrac{1}{2} \sum_i M_i^{-1}\, \|p_i\|^2, \tag{7}$$

with $\epsilon = 10^{-12}$ used only to guard against division by zero.

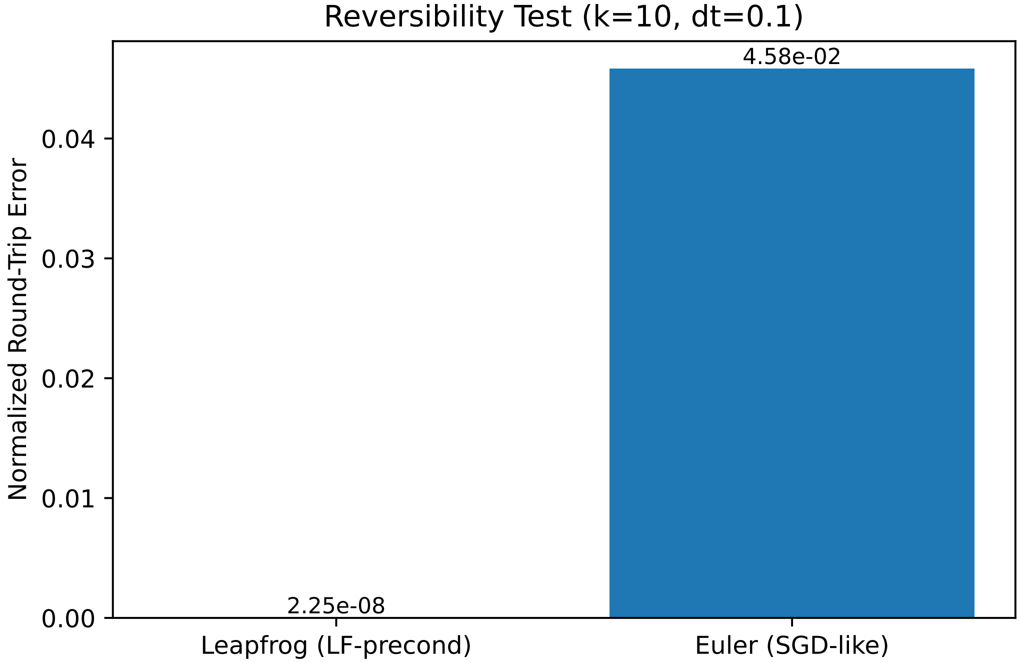


Figure 2: **Reversibility as evidence of conservation.** Symplectic leapfrog reverses its trajectory with normalized round-trip error 2.25×10^{-8}, while first-order methods accumulate 4.58×10^{-2}, a gap of about 6.3 orders of magnitude.

4 Validation and Properties

4.1 Memory Ablations

We tested three memory access patterns: **default** (standard execution), **thrash** (buffer touch between half-steps; cache stress), and **scramble** (storage rebind; breaks locality). Result: *default* and *thrash* produce identical drift curves (Fig. 1a), while *scramble* disrupts the fit, supporting an algorithmic (not hardware) origin.

4.2 Structure Preservation

A defining feature of symplectic integration is its ability to preserve hidden structure in the dynamics. We validate this in two ways:

1. **Reversibility test.** Integrating forward and then reversing momenta, leapfrog retraces its trajectory to machine precision ($\sim 10^{-8}$ normalized error), while first-order methods accumulate error of order 10^{-2} (Fig. 2).

2. **Cap engagement analysis.** We measure how often the energy cap (safety mechanism) activates. In the anti-dissipative band engagement is $\approx 0\%$, indicating the phenomenon is intrinsic rather than an artifact (Fig. 3).

These diagnostics reinforce that the negative scaling is linked to structure-preserving dynamics.

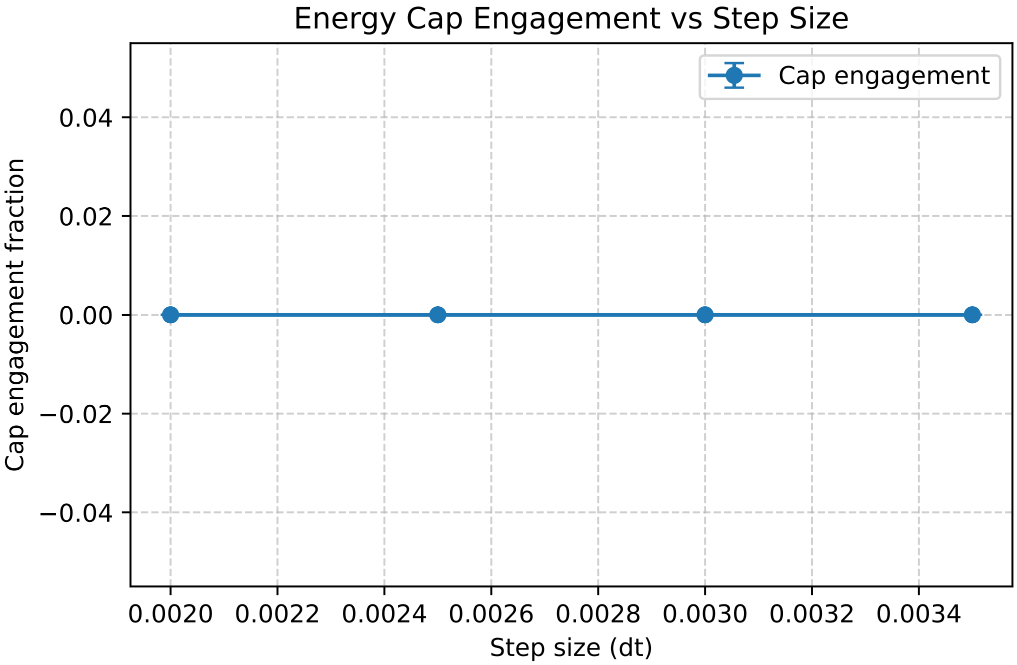


Figure 3: **Energy cap analysis (mean ± 95% CI across seeds).** Engagement is near 0% throughout the anti-dissipative band, indicating the effect is intrinsic rather than an artifact of the cap.

4.3 Experimental Details

- **Dataset**: CIFAR-10 (32×32), standard augmentation [7]
- **Architecture**: ResNet18 (8.7M parameters) [5]
- **Training**: 1 epoch (∼400 steps), batch size 128
- **Hardware**: RTX 4090, CUDA 12.9, PyTorch 2.5+
- **Configuration**: Damping $\gamma = 10^{-3}$, energy cap at $2H_0$
- **Seeds**: 3 independent runs per configuration
- **Artifact**: Interactive demo and code are included in the supplemental package (see `paper/demo/README.md`)

5 Implications

5.1 Theoretical Significance

The existence of anti-dissipative regimes challenges assumptions about optimization dynamics: (i) neural networks are not generic loss surfaces; (ii) discretization choices access qualitatively different dynamical regimes; (iii) the *trajectory* can be as important as the destination.

5.2 Toward Hybrid Optimization

Exploit both regimes: *exploration* (symplectic, small dt) for reversible, stable exploration; *convergence* (AdamW [10]) for rapid local optimization; possible *refinement* by returning to the anti-dissipative band.

5.3 Applications

Potential uses include continual learning (reversibility to mitigate forgetting), ensemble generation (stable exploration), and robustness contexts where conservative dynamics help control drift.

6 Related Work

Our use of structure-preserving updates follows geometric numerical integration and modified-energy perspectives for long-time stability [4, 9]. With damping ($\gamma > 0$), the appropriate lens is quasi-symplectic/Langevin splittings [8, 2]. Mass preconditioning and metric choices connect to work on HMC in Riemannian settings [3, 1]. We situate our observation—negative drift-stepsize scaling of median $|\Delta H|$ in a narrow band—within this literature and contrast it with standard first-order optimizers [5, 7, 6, 10, 11].

7 Limitations and Future Directions

Demonstrated on a single architecture/dataset; the band is narrow and sensitive to hyperparameters; overhead is 2–3×; a complete theory remains open. Future work: conditions across models/data, automated band discovery, efficiency improvements, and rigorous analysis of noise–damping–discretization interplay.

8 Conclusion

Structure-preserving discretization reveals anti-dissipative regimes where larger steps yield less drift. This exposes conservative structure in neural optimization that conventional methods obscure, suggesting new ways to design and stage optimizers around dynamical properties rather than only asymptotic performance.

References

[1] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo, 2017.

[2] Nawaf Bou-Rabee and Houman Owhadi. Long-time stability of variational integrators in the stochastic context. *SIAM Journal on Numerical Analysis*, 48(1):278–297, 2010.

[3] Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

[4] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31 of *Springer Series in Computational Mathematics*. Springer, 2 edition, 2006.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[7] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[8] Benedict Leimkuhler and Charles Matthews. Rational construction of stochastic numerical methods for molecular sampling. *Applied Mathematics Research eXpress*, 2013(1):34–56, 2013.

[9] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian Dynamics*, volume 14 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, 2004.

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[11] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.