THE LOSS DOES NOT SEE THE BASIS, BUT ADAM DOES

Anonymous authors
Paper under double-blind review

ABSTRACT

Different optimization algorithms can find very different solutions to the same problem. Gradient descent (GD) exhibits a low-rank bias in overparameterized problems when initialized near zero, while Adam does not. GD is rotationally equivariant: rotating both factors of a factored model $W = UV^\top$ by an orthogonal matrix leaves the loss unchanged, and GD's solution rotates accordingly. In contrast, Adam is not. We prove that any algorithm using only the current gradient and respecting rotational symmetry amounts to a left-preconditioner depending only on the gradient's Gram matrix. This gives a norm-based test, which passes for the spectral norm in Muon and fails for Adam's ℓ_∞ norm. Empirically, nine optimizers fall into groups consistent with the test on underdetermined matrix sensing. Sharing Adam's second moment across coordinates gradually recovers a low-rank bias, and its solutions approach GD's as the stepsize decreases. On four hyperspectral scenes, Adam's solutions are hundreds to thousands of times further from GD's than two GD runs with different step sizes are from each other. In transformers and LoRA adapters, Adam separates two rotationally equivalent models from the first iteration. Overall, recovering particular solutions depends on the optimizer's invisible coordinate basis, as well as how fast it grows small singular values relative to large ones.

1 INTRODUCTION

It is well-known that gradient descent (GD) dynamics exhibit an implicit bias towards low-rank solutions (Arora et al., 2019; Razin & Cohen, 2020; Li et al., 2021). Specifically, given an overparameterized model that predicts using the product of two tall matrices, $W = UV^\top$, the matrix that GD recovers with small initialization tends to have lower rank compared to other possible solutions with the same loss. Such bias emerges from GD's dynamical properties, and is not enforced by an explicit regularizer. Moreover, the particular solution found by GD is not always the minimum-nuclear-norm solution conjectured by Gunasekar et al. (2017).

Several popular optimizers adaptively rescale updates using an estimate of the scale of gradients in each direction, for example, Adam, RMSProp, signSGD, Lion, Adafactor. They also generalize differently from GD (Wilson et al., 2017; Keskar & Socher, 2017). In contrast to GD, their trajectories are basis-dependent, that is, change under a rotation of parameters (Xie et al., 2025). Interestingly, some variants that share information across directions can be shown to be equivariant (their trajectory rotates along with the parameters), for example, Adam with a single adaptive learning rate for the entire vector and adaptive optimizers that share information across vector-valued parameters (Ling et al., 2022).

Overparameterized matrix factorization objectives have a particular symmetry: if the loss depends on the tall matrices U, V only through their product, $L(U,V) = f(UV^\top)$, then for any orthogonal Q, we have $L(U,V) = L(UQ,VQ)$. This symmetry, which we call the *gauge*, does not change the resulting matrix, $(UQ)(VQ)^\top = UQQ^\top V^\top = UV^\top$, and is thus invisible to the model's predictions. We show that popular optimizers fall into two categories w.r.t. this symmetry: (i) *Equivariant optimizers*, such as GD, Momentum, Adam with a single shared adaptive learning rate (*scalar-Adam*), Muon, Shampoo, exhibit the same trajectory up to rotation from every start (UQ, VQ), the *gauge orbit* of (U,V); (ii) *Coordinate-wise optimizers* such as Adam, RMSProp, signSGD, Lion, and Adafactor, on the other hand, behave differently depending on the gauge-equivalent point from which they start.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

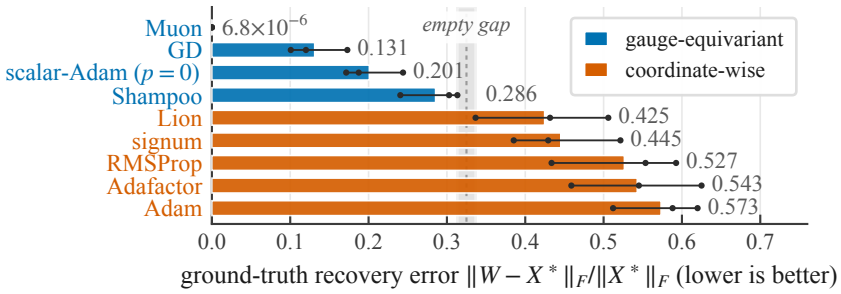

Figure 1: Equivariance sorts the optimizers. Three-seed *mean* recovery error (lower is better) against the planted target for nine rules on underdetermined matrix sensing, all run to interpolation within 2×10^4 steps. Dots are single seeds. The shaded band is the gap between the worst equivariant seed (0.313) and the best coordinate-wise seed (0.337), so the two predicted classes never overlap. Details: Appendix E.3, Table 2.

These latter optimizers lose GD's low-rank bias: on our 40×40 sensing task (§2), after 2×10^4 steps, they reach higher-rank solutions and recover the ground truth worse (Figure 1).

Contributions. **(1) A criterion, and the norm it corresponds to.** We give a criterion for which update rules respect the gauge in §3. Applied to popular optimizers, this recovers the two groups above, and shows that no rule applying a fixed entrywise nonlinearity at zero optimizer state is equivariant. Moreover, we characterize exactly which memoryless update rules (those that use only the current gradient G) are equivariant: precisely those that can be written $\Delta = H(GG^\top)G$. For updates defined as steepest descent under a norm, the test reduces to checking whether the norm is unchanged by rotations applied from the right (Corollary 3.3). A transfer theorem and a momentum lemma extend gradient flow's pathwise conclusions to scalar-Adam-type methods and first-moment methods.

(2) Nine optimizers sorted against ground truth, and a dial isolating the mechanism. We then return to the matrix sensing problem to compare optimization methods (§4). At n=40 there are two clusters of methods, and any method that passes the criterion has better mean-recovery than any method that fails the criterion. These differences persist as n grows up to n=256, with GD, scalar-Adam, and Muon performing better than Adam, RMSProp, Lion and Adafactor. We find that the signum method recovers almost perfectly with a long enough anneal (Appendix D.11), Shampoo with fixed damping performs worse than Adam in $4/5$ problem sizes (these two are the counterexamples to a clean split), and interpolating from Adam's coordinate-wise adaptivity to a single common adaptive learning-rate monotonically improves performance while controlling for the presence of adaptivity and momentum. We control for the anisotropy of Adam's update (its different step size per coordinate) by pooling over rows, which keeps anisotropy but restores equivariance: the resulting method closes about 30% of Adam's gap to scalar-Adam.

(3) Real data, and a gap that does not close at small step sizes. We study the extent to which optimizers preserve the symmetry on real data and in modern architectures. First, on hyperspectral benchmarks at matched training loss we find that gradient descent and scalar-Adam agree in all sixteen seed-paired runs on the two main scenes (§6), while Adam differs substantially. Notably, this difference between Adam and the other methods cannot be explained by optimization quality: Adam's distance to gradient descent's fitted matrix plateaus 2.7 to 3.8 orders of magnitude above the floor set by two gradient-descent runs at different rates on all four scenes, evidence that Adam's difference does not go away at smaller learning rates.

(4) The same gauge inside attention and LoRA. We then show that the same symmetry is present in modern architectures, and that Adam breaks it there too (§5). Namely, in transformers, head logits only depend on the query and key weights W_Q, W_K through their product $W_Q^\top W_K$. While different pairs (W'_Q, W'_K) can give rise to the same product, copies of transformers initialized identically except for such gauge-equivalent choices are indistinguishable under gradient descent or scalar-Adam. Adam separates these copies immediately, and at the end of training their $W_Q^\top W_K$ are 56% apart in relative Frobenius distance. We also show that the BA matrix in low-rank adaptation (LoRA) has the same symmetry, which Adam breaks at the first step while other methods preserve it.

(5) A repair for FlowAdam. Finally, we apply these lessons to FlowAdam (Singh & Sheel, 2026),

an Adam variant that blends short gradient-flow steps into its momentum, and find that we are able to use our symmetry test to identify where it breaks the gauge, repair it accordingly (see Appendix E.22), and improve FlowAdam's matrix recovery results.

Scope. Note that our analysis is not about which optimizer should be used, but about describing which solution a given optimizer will choose. In Appendix E.22.1, we describe in detail the conditions where low-rank bias does and does not help. Our explanation is an empirically falsifiable hypothesis, so we test it in regimes where it is expected to hold as well as fail.

Related work. Theory of factored models centers around the balancedness $U^\top U - V^\top V$, which gradient flow conserves (Arora et al., 2018; Du et al., 2018). Much of the theory of Adam's implicit bias is based on a margin-based viewpoint (Zhang et al., 2024; Baek et al., 2026). Recently, there is a unified steepest descent perspective on optimizers where Adam corresponds to steepest descent in the ℓ_∞ geometry and Muon corresponds to steepest descent in the spectral geometry (Gronich & Vardi, 2026; Fan et al., 2025; Bernstein & Newhouse, 2024). We use this viewpoint only as motivation for the norm used in this paper.

The closest work to this paper studies rotations in the ambient space of the parameters or features rather than the gauge symmetry internal to the factorization. Specifically, Ling et al. (2022) and Xie et al. (2025) study how Adam is impacted by the basis in which parameters are represented, and DePavia et al. (2025) explain this effect with an inversion of Adam's richness bias in classification problems. Finally, Yen et al. (2025) study the LoRA reparametrization as a deliberate design target. What we ask is which interpolating solution the optimizer reaches when the symmetry is internal to the factorization. See Appendix F (Table 23) for more of a summary of these works according to the symmetries studied.

2 Setup: gauge, task, and protocol

Factored model and gauge transformation. We consider a differentiable loss $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ and the factored objective $L(U,V) = f(UV^\top)$ with $U,V \in \mathbb{R}^{n\times k}$, overparameterized so that k is at least the rank of any solution of interest. The product $UV^\top$ does not change if both factors are rotated together. For any orthogonal $Q \in \mathrm{O}(k)$, define the gauge transformation $\rho_Q(U,V) = (UQ, VQ)$. Since $(UQ)(VQ)^\top = UV^\top$, the loss is invariant: $L \circ \rho_Q = L$ whatever f is. Every claim below holds for non-square $UV^\top$ too, because the gauge acts only on the shared latent dimension k.

Definition 2.1 (Gauge equivariance of an optimizer)**.** Consider an optimizer with internal state s_t (e.g. momentum terms, preconditioning matrices) and update rule

$$(U_{t+1}, V_{t+1}, s_{t+1}) = \mathscr{A}\big(U_t, V_t, s_t; \nabla L(U_t, V_t)\big).$$

It is *gauge-equivariant* if for every $Q \in \mathrm{O}(k)$ there is a corresponding transformation σ_Q on the state such that initializing from $\big(\rho_Q(U_0,V_0), \sigma_Q(s_0)\big)$ leads to states $\big(\rho_Q(U_t,V_t), \sigma_Q(s_t)\big)$ at every step t. The state is taken only up to components that never reach the update (Appendix E.14 gives an example).

All optimizers we consider are initialized from a zero state; we adopt the convention that $\sigma_Q(0) = 0$.

For Adam, however, no such map exists for a general rotation Q. The problem lies in squaring the gradient, an operation that does not commute with rotations, that is, $Q(x \odot x) \neq (Qx) \odot (Qx)$. Adam is still equivariant under the special rotations that only reorder coordinates and flip their signs (signed permutations): reordering commutes with elementwise squaring, and squaring removes the signs. Our scalar version of Adam, scalar-Adam, fixes this problem by replacing coordinate-wise squared terms with a single, gauge-invariant scalar. So for every equivariant optimizer the product $W_t = U_t V_t^\top$ is the same from every gauge-equivalent start, at every step t; for Adam it is not.

Matrix sensing. To evaluate the impact of different optimizers' symmetries on their implicit biases, we follow the setup of Gunasekar et al. (2017) and consider low-rank matrix sensing with Gaussian measurements as studied by Recht et al. (2010): we observe m Gaussian linear measurements $y_i = \langle A_i, X^* \rangle$ and minimize $f(W) = \mathrm{mean}_i(\langle A_i, W\rangle - y_i)^2$. We seek to recover a planted, rank-3 matrix X^*. The dimension of the matrix is $n = 40$ and the number of measurements is twice the

degrees of freedom in the solution. This puts us well within the interpolation regime: there are infinitely many matrices that fit the data, including ones of rank well above 3. We report the relative Frobenius error with respect to the planted matrix, $\|UV^\top - X^*\|_F / \|X^*\|_F$, which we call *recovery* (smaller is better), as well as the effective rank of the learned matrix, a measure of the spread of the singular values. Although we report the latter, our claims are based mainly on recovery with respect to X^*, as it is possible for a matrix with large distance to X^* to have low effective rank. For all optimizers, we run them until they interpolate, meaning training loss below 10^{-7}, unless they fail to do so. All experiments use no explicit regularization unless noted (Appendix E.22.1 studies it). Details, the protocol and the control battery are in Appendices C, E.18 and D.

3 Theory: equivariance as a classifier of update rules

Given a loss function $L(U,V)$, consider a reparameterization $(U,V) \to (UQ,VQ)$ for some $Q \in \mathrm{O}(k)$. The gradients of the loss w.r.t. each factor transform covariantly:

$$\nabla_U L(UQ,VQ) = \big(\nabla_U L(U,V)\big)Q, \qquad \nabla_V L(UQ,VQ) = \big(\nabla_V L(U,V)\big)Q,$$

as stated in Lemma B.2 of Appendix B, where we give all proofs. It is this which determines which update rules preserve the intrinsic geometry of the factorization, and which of them read an arbitrary coordinate system instead. We call optimization rules that respect this transformation *equivariant*; see Definition 2.1. We show in Proposition A.1 that four families qualify: gradient descent, including Polyak or Nesterov momentum, and more generally any fixed linear combination of past gradients evaluated at correspondingly transformed points; scalar-Adam; Muon; and Shampoo, even with positive damping. For scalar-Adam, this is because it replaces the per-coordinate second moment with one bias-corrected scalar, which is built from gauge-invariant statistics. For Muon, the update, given by $\Delta = \mathrm{msign}(M_t)$, is the orthogonal factor of the momentum (its SVD with every nonzero singular value set to 1), and it commutes with the gauge exactly, including for the finite Newton–Schulz iteration used to compute it. Next we show that Adam, RMSProp, signSGD, Lion and Adafactor are not equivariant. These optimizers all have some amount of nonlinearity applied coordinate-wise at some point in the calculation. For the first four it happens during the first optimizer update, meaning that the failure of gauge-equivariance shows up in the represented matrix at the very first step; Proposition A.2 gives the exact identity for the gap $\widetilde{W}_1 - W_1$ between the step-1 products of a run and of the same run started from a rotated initialization. For Adafactor the nonlinearity happens in the computation of the factored statistics. The next question is whether this pattern is the whole story for memoryless rules, whose update depends only on the current gradient. Theorem 3.2 answers it exactly.

Proposition 3.1 (Nonlinear coordinate-wise maps break the gauge)**.** *Let $k \geq 2$ and let a memoryless update have the entrywise form $\Phi(G)_{ij} = \phi(G_{ij})$ for a fixed $\phi : \mathbb{R} \to \mathbb{R}$. If $\Phi(GQ) = \Phi(G)Q$ for every G and every $Q \in \mathrm{O}(k)$, then ϕ is linear. No rule applying a fixed nonlinearity coordinate-wise at zero optimizer state is gauge-equivariant.*

Appendix E gives an explicit k=2 example of this failure.

Theorem 3.2 (Structure: equivariant memoryless rules are Gram-determined left preconditioners)**.** *Let Φ, the map from a gradient $G \in \mathbb{R}^{n\times k}$ of any rank to an update, be memoryless. Then $\Phi(GQ) = \Phi(G)Q$ for all $Q \in \mathrm{O}(k)$ if and only if $\Phi(G) = H\left(GG^\top\right)G$ for some matrix-valued H depending only on the Gram matrix $GG^\top$, a quantity the gauge leaves invariant. The proof gives H canonically, at every rank.*

Proposition A.1 lists examples, and Theorem 3.2 says every one has the same form: any update that is equivariant and depends only on the current gradient is $H(GG^\top)G$ for some H. Rules that keep additional state, such as momentum, require separate treatment; for instance, such an update can be nonzero when the gradient happens to be zero. We discuss how to characterize these rules in Remark A.4.

The remaining freedom sits in H, which may be *any* matrix function of $GG^\top$, and even a fixed nonscalar left multiplier $\Phi(G) = DG$ is equivariant. When H applies one scalar function to each eigenvalue of $GG^\top$, however, the whole rule collapses to one *spectral transfer function* $h(\sigma)$, the singular value the update assigns to a gradient singular value σ (Proposition A.3). Gradient descent

has $h(\sigma) = \eta\sigma$ and the polar map has $h \equiv \eta$, and these two ends of the transfer function define the schedule axis of §6.

The criterion in the language of norms. Several recent papers describe deployed optimizers as steepest descent under a norm: Adam and the sign rules under ℓ_∞, Muon under the spectral norm (Xie & Li, 2024; Xie et al., 2025; Bernstein & Newhouse, 2024; Fan et al., 2025; Gronich & Vardi, 2026). Where a rule is generated by a norm, our criterion follows directly from that description, because the gauge acts on the right. The converse fails, however: the structure theorem puts no positivity constraint on H, so not every equivariant map comes from a norm.

Corollary 3.3 (Norm form of the criterion)**.** *Call a norm* right-invariant *if $\|MQ\| = \|M\|$ for every $Q \in \mathrm{O}(k)$. That happens exactly when the norm sees M only through the Gram matrix $MM^\top$. Steepest descent under a right-invariant norm is then gauge-equivariant, provided the steepest-descent direction is unique or an equivariant selection is used. If the norm carries optimizer state, the same holds as long as that state transforms with the gauge in the sense of Definition 2.1. Appendix E states both parts formally.*

Each entry of Proposition A.1 then becomes a one-line check on a norm. The spectral and Frobenius norms are functions of singular values alone, so they are right-invariant, which places Muon and gradient descent inside the class, while entrywise ℓ_p is not right-invariant for any $p \neq 2$, which is Proposition 3.1 from the norm side.

Theorem 3.4 (Transfer theorem, informal; Appendix B)**.** *Let the scalar-preconditioned flow $\dot\theta = -\nabla L(\theta)/a(t)$ have a clock $a(t) > 0$ that is determined solely by gauge-invariant statistics of the trajectory, is finite almost everywhere, and has $1/a$ locally integrable. Then the time change $\zeta(t) = \int_0^t du/a(u)$ turns any solution of it into a gradient-flow solution: the two follow the* same path *in parameter space, the whole of it and any limit point if $\int_0^\infty du/a(u) = \infty$. Under locally Lipschitz ∇L that solution is unique, so it is* the *gradient flow from that initialization.*

Any gradient-flow conclusion that rests only on the path or the limit point therefore transfers (Gunasekar et al., 2017; Arora et al., 2019; Li et al., 2021). Time-dependent conclusions such as convergence rates do not. Lemma A.6 (Appendix A) extends the transfer to a first-moment average in the small-step limit. It does not cover scalar-Adam's second-moment recursion, which we check numerically in §6.

Remark 3.5 (Scope: what equivariance classifies, and the axis left over). Equivariance says which basis changes a rule cannot see, and that is already testable: Adam is blind to a signed permutation but not to a rotation of equal size (Appendix E.7).

Equivariance is also what makes the transfer of Theorem 3.4 uniform over the gauge orbit, so that every gauge-equivalent start inherits the same conclusion. But it does not put a method on gradient flow's path, and it does not by itself decide which interpolant a method reaches.

Inside the class of Theorem 3.2, with H a spectral function, the only remaining freedom is the transfer function $h(\sigma)$ (Proposition A.3). The pair (equivariant, h) is then a complete description, with two axes. The second axis is the spectral schedule: the greedy sequential growth of singular values from small initialization (Li et al., 2021) that §6 maps.

Both axes matter. The factor-aware preconditioner of Appendix E.16, our own variant in the style of ScaledGD (Tong et al., 2021), preconditions each factor with the other factor's Gram matrix. It is equivariant, but it flattens the spectrum, a poor schedule, and it reaches recovery error 0.61, worse than Adam. In the other direction, Appendix D.11's long-anneal signum carries momentum state, so it falls outside the memoryless class, and its aggressive schedule reaches recovery without the symmetry.

4 Sorting the zoo, and isolating the cause

We run nine optimizers, three seeds each, shared across methods, on the sensing task from §2: gradient descent (GD), scalar-Adam, Muon (Jordan et al., 2024), Shampoo (Gupta et al., 2018), Adam (Kingma & Ba, 2015), RMSProp (Tieleman & Hinton, 2012), signum (that is, signSGD with momentum, Bernstein et al., 2018), Lion (Chen et al., 2023), Adafactor (Shazeer & Stern, 2018).

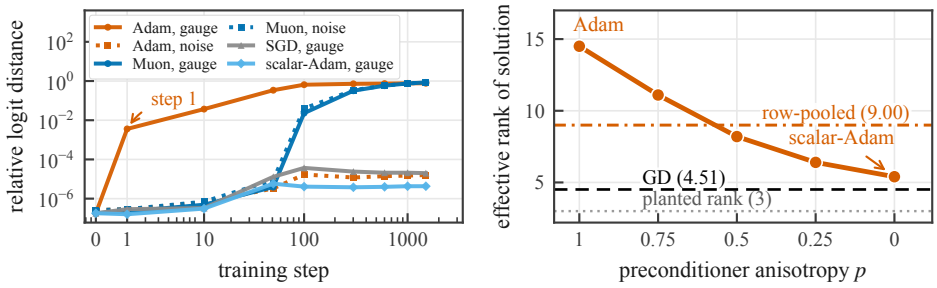

Figure 2: Left: Adam is basis-dependent inside an attention head. Gauge twins start as the same function up to a per-head rotation. Noise twins differ by a 10^{-7} perturbation. Adam splits at step 1, four orders of magnitude above its noise twin. The equivariant methods sit at that floor, and Muon separates later, at the scale of its own noise twin. **Right: the dial.** Effective rank falls to scalar-Adam's value as the preconditioner anisotropy $p \to 0$, and recovery improves with it, $0.570 \to 0.201$. Solid is the best-rate envelope and dash-dot the row-pooled arm (Appendices E.13, E.15).

Of these, we predict GD, scalar-Adam, Muon and Shampoo are equivariant, and predict that Adam, RMSProp, signum, Lion and Adafactor break the gauge.

For each optimizer, we report the rate from its own grid that gives the best recovery among the rates that reach interpolation. Every equivariant optimizer gets error below 0.3 for some learning rate, while none of the coordinate-wise optimizers can at any learning rate we tested (Appendix D.2). This shows every method at its best, and the two predicted groups still separate clearly. Adam, for example, stays between 0.56 and 0.58 at all five interpolating rates of Table 5, which span two orders of magnitude.

Using the fastest interpolating learning rate instead of the best recovery learning rate still sorts $8/9$ optimizers into the predicted groups; the exception is Muon, whose fastest-interpolating rate recovers poorly (Appendix D.3).

We also experimented with applying a shared cosine decay LR schedule to all nine methods, which still resulted in the same $9/9$ equivariant/coordinate-wise separation, ruling out a shared schedule as the confound (Table 4).

We find that up to n=256 and across our entire 10-seed ladder (Appendix D.10) a restricted version of our observed ordering holds. With the exceptions of signum and fixed-damping Shampoo, every run of GD/scalar-Adam/Muon has lower recovery error than all runs of Adam/RMSProp/Lion/Adafactor at four of the five problem sizes. At n=40, every equivariant method outperforms all coordinate-wise methods with no seed that crosses the boundary (Figure 1). Muon also outperforms our convex nuclear-norm reference solve in additional controlled experiments (Appendix E.12). We summarize results for all nine methods in Table 2.

Isolating the cause with a dial. We test whether Adam's poor performance is caused by the anisotropy in its preconditioner (a different step size per coordinate). To test, we interpolate between Adam and scalar-Adam, adjusting only the preconditioner anisotropy while keeping adaptivity, step-size and momentum identical. We call this new optimizer Adam-p and give its update rule in Algorithm 1. Adam-p interpolates the denominator of Adam's update, replacing $\sqrt{\hat{v}}+\varepsilon$ with $(\sqrt{\hat{v}}+\varepsilon)^p(\bar{s}+\varepsilon)^{1-p}$, where $\bar{s} \triangleq \sqrt{\text{mean}(\hat{v})}$ is a scalar and $\varepsilon = 10^{-8}$ for stability. Since $\bar{s}$ averages $\hat{v}$ over every entry of both factors, it sees the gradients only through their Frobenius norms and is gauge-invariant (Lemma B.2). The denominator reduces to the invariant scalar from Proposition A.1(2) when $p = 0$, so Adam-p at $p = 0$ matches scalar-Adam up to $O(\varepsilon)$. Adam-p matches Adam when $p = 1$.

Effective rank (Figure 2 (right)) and recovery both improve monotonically as we turn the dial from Adam ($p = 1$) towards scalar-Adam ($p = 0$), holding all other hyper-parameters fixed. With a fixed learning rate of 10^{-3}, recovery improves from 0.576 for Adam ($p = 1$) to 0.201 for scalar-Adam ($p = 0$). With the best-performing learning rate for each value of p, we observe the same improvement. Monotonicity holds at four of the five fixed rates, and the amount of repair varies (Appendix E.13). So the gain comes from the dial, not from retuning the step size.

Pooling ablation. We study an ablation in which we pool the denominator of Adam only along the rows of a factor, thus maintaining Adam's anisotropy across rows while recovering gauge equivariance. On the benchmark testbed and with the same grid and seeds as before, this achieves a recovery error of 0.461, intermediate between Adam's 0.570 and scalar-Adam's 0.201. Thus about 30% of Adam's recovery gap is attributable to breaking the gauge symmetry, as discussed in Appendix E.15, while the rest is due to its anisotropy. The anisotropy is also what makes Adam fast: scalar-Adam needs about eight times as many steps to reach interpolation.

5 The gauge in transformers, and what a merge costs

Attention has the same symmetry. An attention head computes logits through $q_i^\top \kappa_j = x_i^\top W_Q^\top W_K x_j$ (Vaswani et al., 2017), up to the usual $1/\sqrt{d_h}$ scaling, with d_h the head dimension. For any orthogonal A_h, replacing (W_Q, W_K) with $(A_h W_Q, A_h W_K)$ leaves every logit unchanged. This is the gauge of §3 with $W_Q^\top W_K$ in the role of $UV^\top$, under the same $\mathrm{O}(k)$ convention §2 fixes. The covariance arguments carry over to this head-level action.

Testing it with twins. To test whether an optimizer respects this symmetry, we train *twins*: two copies of the same network that start from gauge-equivalent weights, so they compute exactly the same function at step 0. Both copies then see the same data in the same order with the same optimizer. If the optimizer is equivariant, the copies keep computing the same function at every step. If it is not, they drift apart. We use four kinds of twin:

- *gauge twin*: each head's W_Q and W_K are rotated by the same random orthogonal matrix. This is the test.
- *identity twin*: the rotation is the identity, so the copies should stay identical. They do, up to numerical precision, which checks the code.
- *noise twin*: no rotation, but a 10^{-7} perturbation is added to the original weights. It shows how far rounding error alone moves the copies apart, so it is the floor.
- *different-init twin*: an independent random initialization. It shows how far apart two unrelated runs end up.

An optimizer breaks the symmetry when its gauge twins drift apart more than its noise twins.

We run this on modular addition (Power et al., 2022), with a 2-layer, 4-head transformer trained on addition mod 47 at representation size 64 (Figure 2 (left), Table 7, Appendix D.4). We remove biases and QK norm so that the symmetry is exact.

Adam's gauge twins split at step 1. At that point their logits already differ four orders of magnitude more than the noise twins'. By the end of training, the gauge-invariant product $W_Q^\top W_K$ is 56% apart in relative Frobenius distance, so Adam takes the query/key product somewhere else and does not just rotate the heads. For SGD and scalar-Adam, the gauge twins stay at the noise floor. Muon's gauge twins do drift apart later in training, but its noise twins drift by the same amount, so this is numerical instability (likely in computing msign), not a failure of equivariance. This holds for every seed and depth we ran. Under mini-batch noise, Adam's step-1 split still exceeds a $100\times$ stronger noise twin by $39\times$. Equivariance must hold at every step, so one broken step is enough to show it is broken, whatever happens later in training.

We propose this four-twin protocol as the diagnostic for any claim of equivariance during training. When the equivariant methods are trained to convergence (at 6 layers and d=256), their gauge twins separate only as much as their noise twins.

Merging twins. Merging means averaging the weights of two models. Its cost is the merged model's held-out loss minus the twins' mean held-out loss. Gauge twins compute the same function at the start, so if the optimizer kept them equivalent they could be merged after training. Even after Procrustes alignment, which rotates one twin to best match the other, merging Adam's gauge twins costs a $+0.052$ nats penalty on character-level text. For comparison, merging twins of the equivariant methods costs -0.0004 to 0 nats, and merging Adam's noise twins costs 0. The cost is specific to twins that split from scratch. Copies that share a short common prefix of training (Appendices E.8, E.10), or that are fine-tuned from the public GPT-2-small weights (Appendix E.11), merge at no cost.

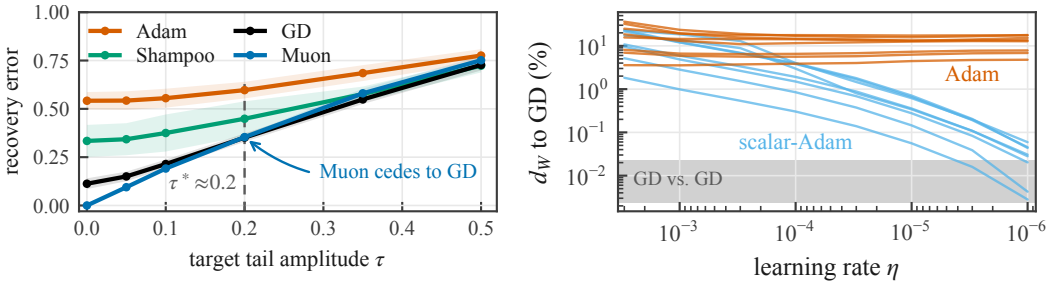

Figure 3: Left: recovery as the target stops being exactly low rank. Moving right puts more of the target outside its top three singular values. Adam is worst everywhere, and the crossover inside the equivariant class is annotated. Ten seeds, bands ± 1 s.d. **Right: distance of each method's fitted matrix from gradient descent's**, both stopped at the same training loss. Each line is one scene at one sampling density, averaged over four seeds. The step size *shrinks* to the right. The grey band is GD against itself at two step sizes, so a line inside it has reached GD's answer. Design and digits: Appendix E.1, Tables 15 and 20.

LoRA has the same symmetry. Low-rank adaptation (LoRA) trains only a low-rank update $\Delta W = BA$ on top of a frozen base model, with $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times d'}$ (Hu et al., 2022). For any orthogonal $Q \in \mathrm{O}(r)$, the adapters $(BQ, Q^\top A)$ give the same ΔW, so LoRA has the same gauge. LoRA initializes $B = 0$, so gauge twins start with the same $\Delta W = 0$ and compute the same function. On both modular addition and character-level language modeling, Adam's twins differ from the first update onwards, while SGD, scalar-Adam and Muon stay within 10^{-16} to 10^{-15} of each other at step 1. Adam's gauge-to-noise ratios are 10^3 to 10^5, depending on the perturbation scale. The twins' held-out losses stay close (± 0.046 and 8×10^{-4} nats on modular addition and text, respectively), so the split is not a difference in quality of fit. Merging adapters is cheap, because the average of the products, $\frac{1}{2}(B_1A_1 + B_2A_2)$, does not depend on the gauge: text twins whose products BA end 0.28 apart merge with a penalty below 10^{-5} nats. Details are in Appendix E.8 and Table 17.

6 Inside the equivariant class, and on real data

The spectral-tail phase diagram. Equivariance is a yes/no property. Behavior within the equivariant class varies continuously, and Proposition A.3 shows that the axis along which it varies is the spectral transfer function $h(\sigma)$. For example, a single step of GD has $h(\sigma) = \eta\sigma$: it scales each singular component by a multiple of its singular value. In contrast, the polar mapping in Muon applies a uniform scale $h \equiv \eta$ to all nonzero singular components; its effect on the predictor depends on the dynamics of the factorization.

Some recent work provides complementary insight into these dynamics. For example, Kang et al. (2026) show that in LoRA-like factorizations under Muon all modes grow at the same rate, an update well-suited to targets that really are low-rank. Meanwhile, Dragutinović et al. (2026) show that Muon can give up GD's sequential simplicity bias in a way that, depending on the data, can make it more prone to fitting spurious features. Our sweep recovers both of these effects, with the spectral tail of the target as the axis between them.

We plant $X = \sqrt{1-\tau^2}\,X_3 + \tau E_\perp$, a rank-3 signal plus a dense tail sitting in the orthogonal complement of the signal; τ is the tail's amplitude and τ^2 its share of the energy. Sweeping τ over 10 seeds shows both axes on one plot (Figure 3 (left)): Adam against the equivariant methods, and Muon against GD.

The crossing point of Muon and GD, $\tau^* \simeq 0.2$ (95% interval $[0.15, 0.25]$ over ten seeds), corresponds to about 4% tail energy. On the symmetry axis, Adam is worst at every plotted τ. Along the schedule axis, Muon and GD are extreme points of the transfer function; the more aggressive optimizer (the one that grows small singular values faster relative to large ones) better recovers the clean low-rank signal but performs worse on a tail. Damped Shampoo performs worse than GD for all τ except the largest, where the difference is within noise.

Overall, we can say that uniform growth is not optimal when the target has a heavy tail. A solvable two-mode model (Proposition A.9) illustrates the contrast: by the time the large mode is fit, equal-rate growth has fit the small mode in full, while GD's has barely moved.

Comparison at matched training loss. Since real data are only approximately low-rank, we use hyperspectral images reported in Graña et al. (2021) and Baumgardner et al. (2015): we report results for Indian Pines and Pavia University, and perform similar experiments on Salinas and Botswana to check our conclusions on all four scenes, as detailed in Appendix D.6. We perform rank-48 factored matrix completion on each, without weight decay, at two sampling densities per scene: each scene is a pixels-by-bands matrix, and we fill in held-out entries from a random observed subset.

We set the sampling density by counting degrees of freedom at rank 24, an approximate rank read off the spectrum and not tuned by validation (Appendix C), with the same count for every method. We compare methods at matched training loss, since otherwise a method that fits the observed entries more slowly can look deceptively better.

Extending the learning-rate comparison. We expand our choice of learning rates three orders of magnitude lower, $\{3 \times 10^{-3}, \ldots, 10^{-6}\}$, and add scalar-Adam to the comparison. For the smallest stepsize, scalar-Adam and GD agree on both statistics (held-out error and effective rank) in all sixteen seed-paired runs over four cells (two scenes at two densities), and the gap shrinks at the $O(\eta)$ rate expected of a discretization error. Additional learning rate results are in Table 19.

Distance between fitted matrices. Finally, we show that Adam converges to a weight matrix different from GD's, and not just a different held-out error. Define the relative matrix distance

$$d_W = \|W - W_{\mathrm{GD}}\|_F \,/\, \|W_{\mathrm{GD}}\|_F \,,$$

evaluated at the matched-loss checkpoints, paired on seed. As a baseline, compare GD's matrix to itself at the same losses using two rates from GD's own grid. Figure 3 (right) shows d_W for Adam and scalar-Adam on all four scenes and both densities, against the learning rate η. Scalar-Adam reaches the calibration floor (grey band) on Pavia and stays within a factor of 15 of it on the other three scenes. Table 20 gives the numbers; Adam's d_W is at least $293\times$ scalar-Adam's d_W in all eight cells. Scalar-Adam's d_W monotonically decreases over three orders of magnitude; Adam's d_W plateaus 2.7 to 3.8 orders of magnitude above the calibration floor in all cells, and rises again in six.

We can argue against slow convergence by decreasing the learning rate by one more factor of ten, to 10^{-7}, and seeing little change (Table 21): Adam's coordinate-wise scaling of its effective step size based on second moments of past gradients persists even when its overall step size is reduced, changing its fitted matrix and not just its rate of convergence. This cannot come from rate selection: the plateau spans the whole range of rates, and d_W, a distance, has no sign for a choice of rate to flip.

Held-out error. Table 9 reports the held-out errors on Indian Pines and Pavia using train-only selected rates; GD beats Adam on both held-out error and effective rank in all cells, for every seed, at loss 10^{-5}. When each method's own best rate is used, Adam beats GD in some cells but not others. Unlike d_W and small-step effective rank, best-rate held-out error does not repeat across scenes, and it does not change monotonically in the range of spectral-tails, data density, or target rank.

7 Discussion, Limitations and Conclusion

Given equal tuning budgets, one can argue that there is no optimizer that is consistently better than Adam tuned properly (Schmidt et al., 2021; Dahl et al., 2023). Indeed, in the 4-scene evaluation grid (Appendix D.6), Adam attains the best held-out error in six of eight cells at the optimal rate.

Our main-text experiments range from 40×40 to 2000×200. Sensing experiments (Appendix D.10) were done up to $n = 256$ and averaged over ten seeds. The splitting of the attention twins (§5) was validated at GPT-2-small width. There are three exceptions to the 2-cluster behavior observed for the nine methods: Shampoo with its fixed damping, signum with a long-enough annealing phase, and Muon when trained on the Indian Pines data. All three lie along the schedule axis (Appendices D.10, D.11 and D.5). So the gauge prediction has no counterexample, while the recovery ordering has three, which is why we claim the nine-method separation only at $n = 40$.

Our results give a test: given an optimizer, verify whether the norm that generates the step is invariant to the action of the group. This will tell us whether gauge-equivalent models stay gauge-equivalent under optimization, but to know which interpolant will be selected, we also need to understand the schedule.

REPRODUCIBILITY STATEMENT

Complete proofs of all numbered results are in Appendix B, and their full statements are in Appendix A. Appendix C gives the experimental setup, including the numerical precision and hardware of every run. It also describes how the hyperspectral data are preprocessed. Appendix D gives the control battery. Anonymized code is provided with this submission. It contains the scripts and logs behind every experimental table and figure, among them all control conditions: (i) schedule symmetrization, (ii) model selection criterion, (iii) LR curves, (iv) alternate init, (v) scaled-up noise twins. We use a fixed, per-run random seed. Software versions and the deterministic-kernel settings are listed in the README of the anonymized code. We will also release the code publicly upon acceptance.

USE OF LARGE LANGUAGE MODELS

Large Language Models (LLMs) were used to implement and refactor experiment code, expand intermediate algebraic steps based on the authors' outlines, and polish prose for clarity and readability. LLMs were not used to generate data, experimental results, or references. The authors led the research, and the theoretical results and experimental design are their own. The authors ran and tested all code, verified every reported number against the experiment logs, and take full responsibility for the entirety of this work.

REFERENCES

Samuel K. Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations (ICLR)*, 2023.

Nicola Aladrah, Emanuele Ballarin, Matteo Biagetti, Alessio Ansuini, Alberto d'Onofrio, and Fabio Anselmi. Understanding and inverse design of implicit bias in stochastic learning: a geometric perspective. *arXiv preprint arXiv:2601.06597*, 2026.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning (ICML)*, 2018.

Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Beomhan Baek, Minhak Song, and Chulhee Yun. Implicit bias of per-sample Adam on separable data: Departure from the full-batch regime. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2510.26303.

Marion F. Baumgardner, Larry L. Biehl, and David A. Landgrebe. 220 band AVIRIS hyperspectral image data set: June 12, 1992 Indian Pine test site 3. https://doi.org/10.4231/R7RX991C, 2015.

Pierfrancesco Beneventano, Mahmoud Abdelmoneum, and Tomaso Poggio. The spectral dynamics and noise geometry of Muon. *arXiv preprint arXiv:2606.08388*, 2026.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML)*, 2018.

Matias D. Cattaneo, Jason M. Klusowski, and Boris Shigida. On the implicit bias of Adam. In *International Conference on Machine Learning (ICML)*, 2024.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Hung-Hsu Chou, Carsten Gieshoff, Johannes Maly, and Holger Rauhut. Gradient descent for deep matrix factorization: Dynamics and implicit bias towards low rank. *Applied and Computational Harmonic Analysis*, 68:101595, 2024a.

Hung-Hsu Chou, Holger Rauhut, and Rachel Ward. Robust implicit regularization via weight normalization. *Information and Inference: A Journal of the IMA*, 13(3):iaae022, 2024b.

Jeremy M. Cohen, Simran Kaur, Yuanzhi Li, J. Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations (ICLR)*, 2021.

George E. Dahl, Frank Schneider, Zachary Nado, Naman Agarwal, Chandramouli Shama Sastry, Philipp Hennig, Sourabh Medapati, Runa Eschenhagen, Priya Kasimbeg, Daniel Suo, et al. Benchmarking neural network training algorithms. *arXiv preprint arXiv:2306.07179*, 2023.

Adela DePavia, Vasileios Charisopoulos, and Rebecca Willett. How do simple rotations affect the implicit bias of Adam? *arXiv preprint arXiv:2510.23804*, 2025.

Yihe Dong and Will Sawin. Muon^p: Muon with fractional spectral powers. *arXiv preprint arXiv:2606.13867*, 2026.

Sara Dragutinović, Yedi Zhang, and Rajesh Ranganath. To use or not to use Muon: How simplicity bias in optimizers matters. *arXiv preprint arXiv:2603.00742*, 2026.

Simon S. Du, Wei Hu, and Jason D. Lee. Algorithmic regularization in learning deep homogeneous models: Layers are automatically balanced. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

Chen Fan, Mark Schmidt, and Christos Thrampoulidis. Implicit bias of spectral descent and Muon on multiclass separable data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2502.04664.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020.

M. Graña, M. A. Veganzons, and B. Ayerdi. Hyperspectral remote sensing scenes. https://web.archive.org/web/20250115080044/https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes, 2021. Grupo de Inteligencia Computacional (GIC), Universidad del País Vasco (UPV/EHU). Distributes the corrected Indian Pines (AVIRIS) and Pavia University (ROSIS) cubes. Page last edited 12 July 2021; the original page is no longer reachable, so it is cited via the Internet Archive snapshot of 15 January 2025.

Eitan Gronich and Gal Vardi. The implicit bias of Adam and Muon on smooth homogeneous neural networks. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2602.16340.

Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning (ICML)*, 2018.

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.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks. Blog post, https://kellerjordan.github.io/posts/muon/, 2024.

Changmin Kang, Jihun Yun, Baekrok Shin, Yeseul Cho, and Chulhee Yun. Uniform spectral growth and convergence of Muon in LoRA-style matrix factorization. *arXiv preprint arXiv:2602.06385*, 2026.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015. Origin of the character-level Shakespeare experiment; the `tiny-shakespeare` corpus is distributed with the accompanying `char-rnn` release, https://github.com/karpathy/char-rnn/blob/master/data/tinyshakespeare/input.txt.

Nitish Shirish Keskar and Richard Socher. Improving generalization performance by switching from Adam to SGD. *arXiv preprint arXiv:1712.07628*, 2017.

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

Daniel Kunin, Javier Sagastuy-Brena, Surya Ganguli, Daniel L. K. Yamins, and Hidenori Tanaka. Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. In *International Conference on Learning Representations (ICLR)*, 2021.

Hannah Lawrence, Kristian Georgiev, Andrew Dienes, and Bobak T. Kiani. Implicit bias of linear equivariant networks. In *International Conference on Machine Learning (ICML)*, 2022.

Tim Tsz-Kit Lau and Weijie Su. Symmetry-compatible principle for optimizer design: Embeddings, LM heads, SwiGLU MLPs, and MoE routers. *arXiv preprint arXiv:2605.18106*, 2026.

Xinghan Li, Haodong Wen, and Kaifeng Lyu. Adam reduces a unique form of sharpness: Theoretical insights near the minimizer manifold. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Selena Zihan Ling, Nicholas Sharp, and Alec Jacobson. VectorAdam for rotation equivariant geometry optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations (ICLR)*, 2024.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

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

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *International Conference on Machine Learning (ICML)*, 2017.

Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *European Signal Processing Conference (EUSIPCO)*, pp. 606–610, 2007.

Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley: benchmarking deep learning optimizers. In *International Conference on Machine Learning (ICML)*, 2021.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

Tianqi Shen, Jinji Yang, Runze Shi, Jianhao Ma, Jiaye Teng, and Ziye Ma. Towards understanding the power and limits of the Muon optimizer: A river-valley perspective. *arXiv preprint arXiv:2606.21514*, 2026.

Tejas Pradeep Shirodkar. Dead-direction conditioners: Gauge-equivariant preconditioning for deep networks. *arXiv preprint arXiv:2606.29176*, 2026.

Eva Silverstein, Daniel Kunin, and Vasudev Shyam. Symmetry breaking in transformers for efficient and interpretable training. *arXiv preprint arXiv:2601.22257*, 2026.

Devender Singh and Tarun Sheel. FlowAdam: Implicit regularization via geometry-aware soft momentum injection. In *International Joint Conference on Neural Networks (IJCNN)*, 2026. arXiv:2604.06652.

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *Journal of Machine Learning Research*, 19(70), 2018.

Teodor-Mihai Stupariu and Andrei Manolache. How the optimizer shapes learned solutions in equivariant neural networks. *arXiv preprint arXiv:2605.27662*, 2026.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5–RMSProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:26–31, 2012.

Tian Tong, Cong Ma, and Yuejie Chi. Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent. *Journal of Machine Learning Research*, 22(150):1–63, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

Shuo Xie and Zhiyuan Li. Implicit bias of AdamW: ℓ_∞-norm constrained optimization. In *International Conference on Machine Learning (ICML)*, 2024.

Shuo Xie, Mohamad Amin Mohamadi, and Zhiyuan Li. Adam exploits ℓ_∞-geometry of loss landscape via coordinate-wise adaptivity. In *International Conference on Learning Representations (ICLR)*, 2025.

Jui-Nan Yen, Si Si, Zhao Meng, Felix Yu, Sai Surya Duvvuri, Inderjit S. Dhillon, Cho-Jui Hsieh, and Sanjiv Kumar. LoRA done RITE: Robust invariant transformation equilibration for LoRA optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Liu Ziyin, Mingze Wang, Hongchao Li, and Lei Wu. Parameter symmetry and noise equilibrium of stochastic gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Chenyang Zhang, Difan Zou, and Yuan Cao. The implicit bias of Adam on separable data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Haozhou Zhang. PolarAdamW: Disentangling spectral control and Schur gauge-equivariance in matrix optimisation. *arXiv preprint arXiv:2605.07067*, 2026.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. GaLore: Memory-efficient LLM training by gradient low-rank projection. In *International Conference on Machine Learning (ICML)*, 2024.

APPENDIX

A The numbered results in full **15**
A.1 The equivariant class . . . 15
A.2 The exact first-step Adam defect . . . 16
A.3 The spectral transfer function . . . 16
A.4 Extending the criterion to stateful rules . . . 17
A.5 More than two factors . . . 18
A.6 Momentum and the small-step flow . . . 18
A.7 Balancedness . . . 19
A.8 Discretization and the step size . . . 19
A.9 The phase boundary . . . 19
B Proofs **20**
B.1 Transfer theorem, formal statement . . . 20
B.2 Maximal isometric gauge . . . 21
B.3 Gradient covariance . . . 21
B.4 Proposition A.1 (equivariant class) . . . 22
B.5 Proposition 3.1 (coordinate-wise rules break the gauge) . . . 23
B.6 Proposition A.2 (exact first-step Adam defect) . . . 24
B.7 Theorem 3.2 (structure) . . . 24
B.8 Proposition A.3 (spectral transfer) . . . 25
B.9 Corollary A.5 (deeper factorizations) . . . 26
B.10 Theorem 3.4 (transfer theorem) . . . 26
B.11 Lemma A.6 (momentum preserves the small-step flow) . . . 27
B.12 Proposition A.7 (balancedness) . . . 28
B.13 Proposition A.9 (greedy vs. equal-rate phase boundary) . . . 28
C Experimental details **29**
D The control battery **33**
D.1 Schedule symmetrization . . . 33
D.2 The learning-rate axis . . . 35
D.3 Learning-rate curves and selection rules . . . 35
D.4 Attention twins across seeds, initialization draws, and noise levels . . . 36
D.5 Hyperspectral learning-rate hygiene . . . 38
D.6 Four scenes, and what holds on all of them . . . 39
D.7 Doubling the rank cap . . . 40
D.8 Interpolation-bar audits . . . 41
D.9 FlowAdam-p is not an artifact of early stopping . . . 41
D.10 The H100 replication ladder . . . 41
D.11 Recovery without equivariance: annealed sign descent at long horizons . . . 44
E Additional results and derivations **45**
E.1 The spectral-tail sweep: design and table . . . 45
E.2 A two-dimensional witness for Proposition 3.1 . . . 46
E.3 Reading the zoo figure, and two further readings from the table . . . 46
E.4 The norm form of the criterion, stated in full . . . 46
E.5 Shampoo and part (ii) of the norm form . . . 47
E.6 Three qualifications on the norm correspondence . . . 47
E.7 One instance, one initialization, four latent bases . . . 48
E.8 Merge experiment details . . . 49
E.9 Reparameterizing a model mid-training . . . 51
E.10 The fork-time sweep . . . 51
E.11 Fine-tune twins on released GPT-2 weights . . . 52
E.12 The nuclear-norm reference solve . . . 52
E.13 The dial: geometric-mean variant, balancedness, and scale . . . 53
E.14 The dial in pseudocode . . . 53
E.15 The row-pooled control . . . 54
E.16 The factor-aware preconditioner of Remark 3.5 . . . 55
E.17 Breaking the head gauge on purpose . . . 55
E.18 The protocol in full . . . 56
E.19 Attention twins at scale and under mini-batch noise . . . 56
E.20 Do the two methods fit the same matrix, or only score the same? . . . 56
E.21 The matched-loss trajectories and the density sweep . . . 58
E.22 The FlowAdam repair . . . 59
F Extended related work **61**

A THE NUMBERED RESULTS IN FULL

This appendix states in full every numbered result that is not stated in the body. They appear in the order in which Appendix B proves them. The two exceptions are the helper Lemmas B.1 and B.2, which are stated and proved together in Appendix B. Any discussion of a result sits directly below it, so a cross-reference from the body always lands in one place.

Throughout, G is the gradient of one factor, an $n \times k$ matrix. The gauge acts on it from the right, $G \mapsto GQ$ with $Q \in \mathrm{O}(k)$ (Lemma B.2). For a rule Φ that uses only the current gradient, equivariance means $\Phi(GQ) = \Phi(G)Q$: rotating the gradient rotates the update in the same way. Definition 2.1 gives the general version, with optimizer state.

A.1 THE EQUIVARIANT CLASS

The four families below are equivariant. So for each of them, the product $W_t = U_t V_t^\top$ depends only on the data and on the gauge orbit of the starting point (U_0, V_0): starting from $(U_0 Q, V_0 Q)$ instead gives the same W_t at every step. Each statement is about the full-matrix rule in exact (real) arithmetic. Real implementations often change the rule: they split variables across factors, mix update rules, or clip individual coordinates. Such an implementation is equivariant only where it follows the stated rule without modification.

Proposition A.1 (Equivariant class)**.** *The following update rules satisfy gauge equivariance as defined in Definition 2.1.*

1. ***Gradient descent and Polyak/Nesterov momentum****: any fixed linear combination of past gradients computed at appropriately transformed points. For Nesterov momentum, the gradient is taken at the look-ahead point, which transforms in the same way as the iterates.*
2. ***Scalar-preconditioned Adam****: Adam with the per-coordinate second-moment estimate replaced by a single scalar $\hat{v}_t$, a bias-corrected exponential moving average (EMA) of gauge-invariant quantities, such as the average of all entries of $g^{\odot 2}$ across both factors. The update is $\Delta \propto \hat{m}_t / a_t$, where $\hat{m}_t$ is the bias-corrected first moment and the denominator is $a_t = \sqrt{\hat{v}_t} + \varepsilon$.*
3. ***Muon****: $\Delta = \mathrm{msign}(M_t)$, where M_t is the momentum buffer and*
$$\mathrm{msign}(M) = AB^\top = \sum_{\sigma_i > 0} a_i b_i^\top$$
is the support-restricted polar factor, with $M = A\Sigma B^\top$ a compact SVD. In words, $\mathrm{msign}(M)$ is M with every nonzero singular value replaced by 1. Equivariance holds exactly under real arithmetic, and it remains valid for any finite-order Newton–Schulz approximation.
4. ***Shampoo****, with damping $\lambda > 0$:*
$$\Delta = (L_t + \lambda I)^{-1/4} G_t (R_t + \lambda I)^{-1/4}, \qquad L_t = \sum_{u \le t} G_u G_u^\top, \quad R_t = \sum_{u \le t} G_u^\top G_u.$$
Equivariance is exact in this setting, even with damping.

Why these four pass. In each case the proof (Appendix B) shows that the update rotates exactly as the gradient does, $\Delta \mapsto \Delta Q$. Gradients and momentum rotate this way. Scalar-Adam's denominator depends only on Frobenius norms, which rotations do not change. Muon's msign commutes with the rotation. For Shampoo, the left accumulator does not change, and the right one is rotated on both sides, which cancels in the update.

Notes on the rules.

- *Muon.* The sum in msign runs only over *strictly positive* singular values. So $\mathrm{msign}(0) = 0$, and no direction in the kernel of M enters the update. The restriction also makes the polar factor unique. If zero singular values were included, A and B would not be unique off the support, and the covariance argument would fail.
- *Shampoo.* Shampoo takes matrix roots of its accumulators, which are rank-deficient early in training. The damping λ keeps these roots well-defined. Our experiments use $\lambda = 1$. For $\lambda = 0$ we take the roots on the active subspace. The letters L and R in L_t and R_t name Shampoo's left and right accumulators, not the objective L.

A.2 THE EXACT FIRST-STEP ADAM DEFECT

Adam's gauge defect is already present, in closed form, after the very first step. The reason is that Adam's first step divides each gradient entry by its own size. That division is the entrywise normalizer D_ε defined below, and it does not commute with rotations. The error term $E_Q(G)$ measures how far it is from commuting with a rotation Q. The proposition compares two runs, one from (U_0, V_0) and one from the rotated start (U_0Q, V_0Q), through the difference of their products after one step.

Proposition A.2 (Adam has an exact first-step gauge defect)**.** *Fix $\varepsilon > 0$. Consider bias-corrected Adam with elementwise operations, a shared step size η, initial moment estimates set to zero, and the update convention*

$$\theta^+ = \theta - \eta\, \hat{m} \oslash \big(\sqrt{\hat{v}} + \varepsilon\big).$$

Let $G_U = \nabla_U L(U_0, V_0)$ and $G_V = \nabla_V L(U_0, V_0)$ be the gradients at initialization, and let $(\widetilde{U}_1, \widetilde{V}_1)$ be the first iterate from the gauge-rotated start (U_0Q, V_0Q). With $|\cdot|$ the entrywise absolute value, define

$$D_\varepsilon(G) \coloneqq G \oslash \big(|G| + \varepsilon\big), \qquad E_Q(G) \coloneqq D_\varepsilon(GQ)Q^\top - D_\varepsilon(G),$$
$$D_U \coloneqq D_\varepsilon(G_U), \qquad D_V \coloneqq D_\varepsilon(G_V).$$

Then the products after one step satisfy the identity

$$\begin{aligned} \widetilde{W}_1 - W_1 = -\eta \big[E_Q(G_U)V_0^\top + U_0 E_Q(G_V)^\top\big] \\ + \eta^2 \big[E_Q(G_U)D_V^\top + D_U E_Q(G_V)^\top + E_Q(G_U)E_Q(G_V)^\top\big]. \end{aligned} \tag{A.1}$$

Equivalently, the second-order term is $\eta^2\big[D_\varepsilon(G_UQ)D_\varepsilon(G_VQ)^\top - D_U D_V^\top\big]$.

Here $\oslash$ is entrywise division, so each entry of $D_\varepsilon(G)$ is $G_{ij}/(|G_{ij}| + \varepsilon)$, roughly the sign of G_{ij} when $|G_{ij}| \gg \varepsilon$. The identity splits the defect in two. The term of first order in η comes from each factor's one-step update. The term of second order comes from the product of the two factors' updates. $E_Q(G)$ is zero exactly when D_ε commutes with Q at G, and when it is zero for both factors the whole right-hand side vanishes. Gradient descent has the identity map in place of D_ε, which is why its defect is zero.

A.3 THE SPECTRAL TRANSFER FUNCTION

Proposition A.3 (The spectral transfer function: the within-class coordinate)**.** *Suppose $H = \psi(GG^\top)$ for a scalar function ψ, meaning: diagonalize the symmetric positive-semidefinite Gram matrix $GG^\top$ and replace each eigenvalue λ by $\psi(\lambda)$, so that $\psi(GG^\top)$ multiplies each eigenspace by its own $\psi(\lambda)$. Given the singular value decomposition $G = \sum_i \sigma_i u_i v_i^\top$, it follows that*

$$\Phi(G) \;=\; \sum_i h(\sigma_i)\, u_i v_i^\top, \qquad \textit{where} \quad h(\sigma) \coloneqq \psi(\sigma^2)\,\sigma.$$

So the update keeps the gradient's singular vectors u_i, v_i and only changes each singular value σ_i into $h(\sigma_i)$. We call h the spectral transfer function. Whenever an inverse or root of a singular positive-semidefinite matrix appears later, we apply it on the matrix's range and set it to zero on the nullspace.

Instances. For memoryless one-step methods, h has a closed form.

- *Gradient descent:* $h(\sigma) = \eta\sigma$.
- *Common-scalar-preconditioned GD:* $h_t(\sigma) = \eta\sigma/a_t$, which is a time-rescaled GD (Theorem 3.4).
- *The exact polar map, and support-restricted undamped one-step Shampoo,* applied to a single input matrix: $h \equiv \eta$. Both fully flatten the nonzero singular values.
- *The fractional family of Dong & Sawin (2026),* which replaces each σ_i with σ_i^q: $h(\sigma) = \eta\sigma^q$. Their exponent q is not our dial p, and it runs the other way. q=1 is GD and q=0 is the polar map, whereas p=1 is Adam and p=0 is the shared scalar.

Damped Shampoo with accumulation, momentum methods, and Muon all carry state. For them, the one-step formulas above are approximations, not full descriptions of the dynamics.

Two implications of Proposition A.3 are particularly relevant below.

(i) Aggressiveness. The ratio $h(\sigma)/h(\sigma_{\max})$ says how much a rule boosts small singular values relative to the largest one. A larger ratio means a more aggressive rule, and this ratio defines the aggressiveness axis explored in §6. The ratio alone does not rank real optimizers, because real optimizers do not apply their one-step formula. Shampoo sums gradients over all past steps (accumulation) and adds λI (damping), and Muon acts on momentum rather than on the current gradient. For instance, the idealized exact polar map and undamped one-step Shampoo both have $h \equiv \eta$, yet neither deployed optimizer behaves like $h \equiv \eta$. What the axis does place is GD and Muon, at its two ends, and §6 maps the crossover between them.

The axis does not place damped Shampoo. At $\lambda{=}1$ its recovery error is worse than GD's at every τ of Table 15 except the largest, where the two tie within noise. We report this gap without resolving it.

(ii) Continuity. Near rank-deficient matrices the exact polar map jumps, while GD changes smoothly. To see this, compare the two nearby gradients $\mathrm{diag}(1, \pm\delta)$, where δ is a small singular value (not Adam's stabilizer ε). The two gradients differ by 2δ, while their updates differ by $2|h(\delta)|$. So the Lipschitz constant of Φ is at least $|h(\delta)|/\delta$ (Appendix B gives the details). For the exact msign operation, this bound diverges as $\delta \to 0$ whenever $h(0^+) > 0$. The exact polar map and support-restricted one-step Shampoo are both in that case, since both have $h \equiv \eta$. In contrast, the ratio stays bounded for GD and scalar-preconditioned GD, at η and η/a_t respectively, provided a_t stays bounded away from zero. Rules whose scaling vanishes with the gradient norm (e.g. $a = \|G\|_F$) are excluded.

Finite Newton–Schulz. In practice Muon computes msign approximately, with a fixed number of Newton–Schulz iterations. This finite map is equivariant in exact arithmetic (Case 3 of the proof of Proposition A.1). Unlike the exact map, it is also Lipschitz continuous. The iteration starts from $X_0 = M/(\|M\|_F + \varepsilon_{\mathrm{NS}})$, where $\varepsilon_{\mathrm{NS}}$ is a small stabilizer, and uses the first coefficient $\gamma_1 = 3.4445$ from Jordan et al. (2024) (see Appendix C). Near a zero singular value the cubic and quintic terms vanish, so each iteration multiplies the slope there by γ_1. After five iterations, the slope at zero singular values, in the units of M, is

$$\frac{\gamma_1^5}{\|M\|_F + \varepsilon_{\mathrm{NS}}}.$$

This slope depends on the whole matrix, through $\|M\|_F$. A transfer function $h(\sigma)$ looks at one singular value at a time, so it cannot describe this map. The practical upshot is that the finite map can boost small singular values strongly without the jump of the exact map. That is a different effect from breaking the gauge symmetry, and the twin protocol of §5 tells the two apart experimentally.

A.4 Extending the criterion to stateful rules

The results so far cover memoryless rules, whose update depends only on the current gradient. Real optimizers carry state: momentum buffers, second-moment accumulators, and preconditioner matrices. Remark A.4 extends the equivariance criterion to them.

Remark A.4 (Stateful rules and right preconditioners). For a stateful rule, each piece of optimizer state transforms in one of three ways under the gauge.

- *Covariant* (picks up Q on the right, like the gradient): momenta, $M_t \mapsto M_t Q$.
- *Invariant* (does not change): Frobenius norms and left-side accumulators $L_t = \sum_u G_u G_u^\top$.
- *Conjugated* ($Q^\top$ on the left and Q on the right): right-side quantities, $R_t \mapsto Q^\top R_t Q$. This includes both $R_t = \sum_u G_u^\top G_u$ and the factor Gram matrices $U^\top U$ and $V^\top V$.

Any update built as

$$\Delta = H(\mathscr{I}_t)\, Z_t\, K(R_t),$$

where Z_t is a covariant quantity, like G_t or M_t, $\mathscr{I}_t$ is gauge-invariant, and K is a spectral function, then transforms like the gradient, $\Delta \mapsto \Delta Q$, so it is equivariant. Indeed, $H(\mathscr{I}_t)\,(Z_t Q)\,(Q^\top K(R_t) Q) = H(\mathscr{I}_t)\, Z_t\, K(R_t)\, Q$. This form covers momentum-based gradient descent ($Z_t = M_t$), scalar-Adam, Muon ($Z_t = M_t$), Shampoo, and ScaledGD. It is one way to build equivariant stateful rules, not the only one: it covers rules built as a product of per-factor pieces and gauge-invariant couplings.

A.5 More than two factors

The body works with a two-factor decomposition $W = UV^\top$. The following corollary extends the results to depth-D factorizations $W = U_1 U_2 \cdots U_D$. The key new difficulty is that interior factors see gauge rotations on both sides, not just one.

Corollary A.5 (Deeper factorizations)**.** *Assume now a composite weight matrix $W = U_1 U_2 \cdots U_D$ of depth D (we write D for depth because L is the loss). The loss is then unchanged by the following move at each interface i between neighbouring factors:*

$$(U_i, U_{i+1}) \mapsto (U_i Q_i, Q_i^\top U_{i+1}), \qquad Q_i \in \mathrm{O}(k_i).$$

Then gradients transform covariantly at each interface: each factor's gradient picks up the interface rotation on the same side as the factor does. Consequently:

- *(a)* Outer factors. *U_1 and U_D see a one-sided action, and the earlier conditions apply to them directly.*
- *(b)* Interior factors. *An interior factor U_i sees both sides at once, $G \mapsto Q_{i-1}^\top G Q_i$, and right-equivariance alone (the condition $\Phi(GQ) = \Phi(G)Q$ used so far) is not sufficient there. What it needs is bi-equivariance,*

  $$\Phi(Q_L^\top G Q_R) = Q_L^\top \Phi(G)\, Q_R \qquad \text{for all orthogonal } Q_L, Q_R \text{ of matching sizes},$$

  with state transforming to match. The gradient-only, momentum, and msign *rules satisfy this. A fixed non-scalar left multiplier $\Phi(G) = CG$ does not: it is right-equivariant but fails on the left.*
- *(c)* Non-equivariance carries over directly. *Any fixed nonlinear coordinate-wise update rule fails to be equivariant at any interface where the internal dimension satisfies $k_i \geq 2$.*
- *(d)* A common scalar. *If all factor gradients are scaled by the same positive scalar $a(t)$, then the dynamics for all D factors reduce to a single time-rescaled gradient flow, and the transfer theorem continues to hold under its original conditions. Equivariance at individual interfaces does not by itself imply the transfer.*

The multiplier C in (b) is the DG of §3, renamed because D is the depth here. The gauge freedom of the type discussed in §5 is the case $D = 2$.

A.6 Momentum and the small-step flow

Adding momentum to a scalar-preconditioned method does not change where it goes in the small-step limit: it still follows the flow of Theorem 3.4. The lemma makes this precise over any fixed stretch of time.

Lemma A.6 (Momentum preserves the small-step flow)**.** *Fix $\beta \in [0,1)$ and a gauge-invariant scalar preconditioner a, a function of the parameters θ. Take the first-moment recursion started from $m_0 = 0$, with bias correction and update*

$$m_{t+1} = \beta m_t + (1-\beta) g_t, \qquad \hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta^{t+1}}, \qquad \theta_{t+1} = \theta_t - \eta\, \frac{\hat{m}_{t+1}}{a_t},$$

where $g_t = \nabla L(\theta_t)$ and $a_t = a(\theta_t)$. We compare the iterates with the flow in physical time $s = t\eta$, joining consecutive iterates by straight lines. So $[0,T]$ below is an interval of physical time, and the iterate count grows as $\eta \to 0$. Assume the following.

- *(A1)* *∇L is locally Lipschitz.*
- *(A2)* *a is locally Lipschitz and bounded away from 0 and ∞ on the region visited.*
- *(A3)* *The iterates stay in a common compact set on $[0,T]$ for all small η.*

Then as $\eta \to 0$ the piecewise-linear path through the iterates converges uniformly on $[0,T]$ to the flow of Theorem 3.4, whose pathwise conclusions therefore extend to that flow in this limit.

Scope. Three limits matter.

- *Time horizon.* The lemma covers a finite time horizon $[0, T]$. It does not take the small-step and long-time limits together. So the agreement at the end of training reported in §6 is an empirical observation.
- *Regularity of a.* It is not enough for a to be bounded. The proof needs $-\nabla L(\theta)/a$ to be a Lipschitz vector field, which can fail if a is not continuous in θ. For example, a gauge-invariant clock that is bounded but discontinuous in the state breaks the Euler estimate. This is why assumption (A2) requires $a(\theta)$ to be locally Lipschitz.
- *Scalar-Adam.* This lemma does *not* cover scalar-Adam's own second-moment recursion. Its denominator depends on discrete state, and at fixed β_2 its memory window depends on η. We do not prove the limit for that coupled recursion here, and §6 checks it numerically.

A.7 Balancedness

Balancedness is the quantity $U^\top U - V^\top V$. Gradient flow keeps it constant, and this has been studied as a mechanism behind low-rank recovery in factored optimization.

Proposition A.7 (Balancedness: a special property of GD-type flows, not a general invariance)**.** *Let* $B_t = U_t^\top U_t - V_t^\top V_t$.

1. *For standard gradient flow, and in particular in the setting of Theorem 3.4,* B_t *is conserved during training.*
2. *For diagonally preconditioned flows of the form* $\dot{U} = -D_U \odot \nabla_U L$ *(and analogously for* V*), where* D_U *and* D_V *hold one scale per entry,* B_t *is conserved when* D_U *and* D_V *are one and the same positive scalar. Otherwise it can drift, at a rate bounded by how far each departs from that common scalar. Agreeing up to a factor is not enough:* $D_U \equiv 1$*,* $D_V \equiv 2$ *already breaks the cancellation.*

Conservation of B_t *is therefore a property of flows with common-scalar preconditioning, not of equivariant optimization procedures in general.*

Balancedness is not necessary for good low-rank recovery. Neither Muon nor Shampoo conserves B_t, and the $\|B_T\|_F$ column of Table 2 puts both far above gradient descent's. Yet Muon still recovers near exactly, and Shampoo still beats every coordinate-wise method. The initialization scale is 10^{-3}, so the starting imbalance is negligible and those final values are drift built up during training.

A.8 Discretization and the step size

The following remark addresses what happens when the flow is discretized into actual optimizer steps.

Remark A.8 (Comments on discretization and step-size choice). Theorem 3.4 is about continuous time, while real optimizers take discrete steps. Under smoothness and stability assumptions, the explicit Euler discretization of the memoryless common-scalar flow tracks the flow's trajectory, but it is only guaranteed to do so up to an $O(\eta)$ error over a fixed time interval.

The learning-rate trends of Appendix D.2 are empirical and specific to this task. A smaller η does not always help: GD and scalar-Adam recover better at the smaller rates tested, while Adam's recovery is almost insensitive to the choice of η (Table 5).

A.9 The phase boundary

An analytically tractable model. The decoupled-mode model reduces the factored dynamics to independent scalar components, each following its own one-dimensional ODE. This is the idealization commonly used in studies of greedy dynamics. Because every mode evolves independently, both sides of the schedule trade-off have closed-form solutions. The model also makes clear why the initialization scale influences one schedule but not the other. The two schedules are GD's, $h(\sigma) = \eta\sigma$, which we call *greedy* because it grows large modes first, and the polar map's, $h \equiv \eta$, which grows every mode at the same rate.

Proposition A.9 (Greedy vs. equal-rate schedules: a solvable two-timescale scenario)**.** *Assume the following setting, the idealization adopted in Arora et al. (2019) and Li et al. (2021).*

- *The factored dynamics decompose into independent, balanced scalar components $w = uv$, with $u = v = \sqrt{w}$, and the target is aligned (it shares the modes' singular vectors).*
- *The target modes satisfy $s_1 > s_2 > 0$, and all are initialized at $w_i(0) = w_0$ with $0 < w_0 \le s_2/2$.*
- *T_1 is the first time the leading mode reaches $(1-\delta)s_1$, with δ fixed such that $0 < \delta < 1 - s_2/s_1$.*

Then the two schedules separate as follows.

(i) Greedy schedule, $h(\sigma) = \eta\sigma$. *Each mode follows logistic growth, $\dot{w}_i = 2\eta\, w_i(s_i - w_i)$, and at time T_1 the smaller mode remains inactive:*

$$w_2(T_1) \;\le\; C_{\delta,\rho}\, s_1 \Big(\tfrac{w_0}{s_1}\Big)^{1-\rho} \xrightarrow[w_0\to 0]{} 0, \qquad \rho := s_2/s_1, \quad C_{\delta,\rho} = 2\big(\tfrac{1-\delta}{\delta}\big)^{\rho}.$$

Thus, reducing initialization enhances the timescale gap between leading and trailing modes indefinitely.

(ii) Equal-rate schedule, $h \equiv \eta$. *All modes grow at the same rate, since $\sqrt{w_i}$ increases linearly with time, and saturate at their respective s_i. By T_1, the smaller mode is already fully fitted, $w_2(T_1) = s_2$, regardless of w_0. The flat schedule does not let the initialization scale differentiate between the modes.*

Both strict lower bounds in the setting are needed. $w_0 = 0$ is a fixed point of the dynamics. $\delta \le 0$ would make the threshold $(1-\delta)s_1 \ge s_1$ unreachable. And the proof divides by these terms.

What the model shows, and what it does not. Proposition A.9 (proved in Appendix B) makes the behavior of this simplified model precise, in a transient regime consistent with the phase behavior we observe. If the target matrix has exactly low rank, fitting all the planted modes at the same rate recovers the full signal in this model, consistent with the equal-rate principle of Kang et al. (2026) for Muon. The model shows a timescale contrast. By T_1 the flat schedule has already fitted the trailing mode in full, while the greedy schedule holds it down at $w_0^{1-s_2/s_1}$.

The model also has a clear limit. Unlike the sensing problem, it observes every direction, so it has no sensing nullspace and proves nothing about unconstrained directions. Applying the contrast to a target whose tail the measurements pin down less well, sensed at $m = 2\,\mathrm{dof}$ measurements counted at the head, is an interpretation of Figure 3 (left), not a consequence of the proposition. The observed transition may therefore depend on the target's spectral properties, the measurement design, the initialization scale, the training length, the schedule, and the optimization algorithm.

What is open here. The first open problem is a full-dynamics theorem for the boundary of Figure 3 (left), since Proposition A.9 is a transient argument about two modes. The second is the diffusion term that mini-batch noise adds. That term is outside every result in this paper (cf. Ziyin et al., 2024, under SGD). Appendix E.9 discusses reparameterizing a model mid-training, which the gauge makes possible.

B Proofs

Each subsection proves one numbered result, in the order of Appendix A. The first three subsections are groundwork: the formal statement of Theorem 3.4, and two helper lemmas. Lemma B.2 is the one fact about the loss that every equivariance proof below uses.

B.1 Transfer theorem, formal statement

Theorem 3.4 (transfer theorem), stated in full. Assume the following.

- The clock $a(t) > 0$ is measurable, $1/a$ is locally integrable, and a is determined solely by gauge-invariant statistics of the trajectory up to time t.

- The trajectory $\theta = (U, V)$ is locally absolutely continuous (differentiable almost everywhere and equal to the integral of its derivative) and satisfies $\dot{\theta} = -\nabla L(\theta)/a(t)$.

Define the time change and the reparameterized trajectory

$$\zeta(t) = \int_0^t \frac{du}{a(u)}, \qquad \widetilde{\theta}(\zeta) \coloneqq \theta(t(\zeta)),$$

where $t(\zeta)$ is the inverse of $\zeta(t)$. In words, ζ is a new clock that runs fast when a is small and slow when a is large. Then the following hold.

1. *Same path.* $\widetilde{\theta}'(\zeta) = -\nabla L(\widetilde{\theta}(\zeta))$ for almost every ζ, so both trajectories follow the same path in parameter space, the whole of it if $\int_0^\infty du/a(u) = \infty$ and only an initial segment of it otherwise.
2. *Same limit.* If in addition $\int_0^\infty du/a(u) = \infty$, the transformed time reaches infinity. So if the gradient flow converges, the preconditioned version converges to the same limit.

B.2 Maximal isometric gauge

This lemma is why the gauge we study is $\mathrm{O}(k)$: among the maps that preserve the product $UV^\top$, it is the largest group that also preserves parameter norms.

Lemma B.1 (Orthogonal gauge as maximal isometric symmetry)**.** *For any $A \in \mathrm{GL}(k)$, the transformation $(U, V) \mapsto (UA, VA^{-\top})$ preserves the product $UV^\top$. This transformation preserves the Euclidean product metric $\|U\|_F^2 + \|V\|_F^2$ for all (U, V) if and only if $A \in \mathrm{O}(k)$.*

Proof. The product is preserved. For $A \in \mathrm{GL}(k)$,

$$(UA)(VA^{-\top})^\top = UAA^{-1}V^\top = UV^\top.$$

Isometry implies orthogonal. Suppose the action is an isometry for every (U, V). Take $V = 0$ and let U have a single nonzero row $u \in \mathbb{R}^{1\times k}$, which is arbitrary. Then $\|uA\|_2 = \|u\|_2$ for every u, hence $AA^\top = I$. Since A is square, this is equivalent to $A \in \mathrm{O}(k)$.

Orthogonal implies isometry. If $A \in \mathrm{O}(k)$, then $A^{-\top} = A$, and both right multiplications preserve Frobenius norms. □

B.3 Gradient covariance

This lemma says that rotating both factors rotates both gradients in the same way. It holds for any loss that sees the factors only through their product.

Lemma B.2 (Gradient covariance)**.** *Let $L(U, V) = f(UV^\top)$ and let $Q \in \mathrm{O}(k)$. Then $\nabla_U L(UQ, VQ) = \big(\nabla_U L(U, V)\big)Q$ and $\nabla_V L(UQ, VQ) = \big(\nabla_V L(U, V)\big)Q$.*

Proof. Let $W = UV^\top$ and $G = \nabla f(W) \in \mathbb{R}^{n\times n}$. By the chain rule,

$$\nabla_U L = GV, \qquad \nabla_V L = G^\top U.$$

At the transformed point (UQ, VQ) the represented matrix is unchanged, $W' = UQ(VQ)^\top = W$, so $\nabla f(W') = G$. Therefore

$$\begin{aligned}\nabla_U L(UQ, VQ) &= G(VQ) = (GV)Q = \big(\nabla_U L(U, V)\big)Q,\\ \nabla_V L(UQ, VQ) &= G^\top(UQ) = (G^\top U)Q = \big(\nabla_V L(U, V)\big)Q.\end{aligned}$$

□

B.4 PROPOSITION A.1 (EQUIVARIANT CLASS)

Proof. Write primed quantities for the run started at (U_0Q, V_0Q), with its state initialized by the map σ_Q constructed below. We show by induction on t that

$$U'_t = U_tQ, \qquad V'_t = V_tQ,$$

with the state transforming as stated. The induction step is the same in every case. If the two runs are gauge-related at step t, Lemma B.2 gives $G'_t = G_tQ$ for both factor gradients. Each case below then shows that the update is also right-multiplied by Q, so the runs are still gauge-related at step $t+1$.

Case 1: GD and momentum. For GD,

$$U'_{t+1} = U'_t - \eta G'_t = (U_t - \eta G_t)\,Q.$$

Momentum buffers are linear in the gradients. With $\sigma_Q : M \mapsto MQ$, induction gives

$$M'_t = \beta M'_{t-1} + G'_t = M_tQ,$$

and the update again right-multiplies by Q. For Nesterov, the gradient is evaluated at the lookahead point $U_t + \beta(U_t - U_{t-1})$. In the primed run this is the same point right-multiplied by Q, so Lemma B.2 applies there verbatim. The lookahead gradient is therefore covariant too, and the induction goes through unchanged.

Case 2: scalar-preconditioned Adam. The first moment is an exponential moving average (EMA), which is linear, so $M'_t = M_tQ$. The second-moment state is the single scalar

$$v_t = \beta_2\, v_{t-1} + (1-\beta_2)\,\mathrm{mean}\big(G_t^{\odot 2}\big), \qquad v_0 = 0,$$

where the mean runs over all entries of both factors. The mean of $G^{\odot 2}$ is $\|G\|_F^2$ divided by the entry count, so v_t sees the gradients only through their Frobenius norms. Since $\|G_tQ\|_F = \|G_t\|_F$,

$$v'_t = v_t, \qquad \hat{v}'_t = \frac{v'_t}{1-\beta_2^t} = \hat{v}_t.$$

The state map is therefore

$$\sigma_Q : (M_U, M_V, v) \mapsto (M_UQ, M_VQ, v),$$

with $\sigma_Q(0) = 0$ as Definition 2.1 requires. The update $-\eta\,\hat{M}_t/(\sqrt{\hat{v}_t}+\varepsilon)$ is a scalar multiple of a covariant matrix, hence covariant.

The mean is linear, so it commutes with the EMA. Hence v_t equals $\mathrm{mean}(v_t)$ for the entrywise EMA v_t of $g^{\odot 2}$, and storing the scalar changes no number. Storing the scalar also gives an explicit state map. The entrywise v_t has none, because squaring and right-rotation do not commute.

Case 3: Muon. With momentum $M'_t = M_tQ$ as in Case 1, it suffices to show $\mathrm{msign}(MQ) = \mathrm{msign}(M)\,Q$. Let $M = A\Sigma B^\top$ be an SVD. Then $MQ = A\Sigma(Q^\top B)^\top$ is an SVD of MQ, because $Q^\top B$ still has orthonormal columns. So

$$\mathrm{msign}(MQ) = A\,(Q^\top B)^\top = AB^\top Q = \mathrm{msign}(M)\,Q.$$

The same holds for the Newton–Schulz iteration

$$X_{j+1} = \gamma_1 X_j + \gamma_2\,(X_jX_j^\top)X_j + \gamma_3\,(X_jX_j^\top)^2X_j, \qquad X_0 = \frac{M}{\|M\|_F + \varepsilon_{\mathrm{NS}}}.$$

If $X_j \mapsto X_jQ$, then $X_jX_j^\top$ is invariant, so every term right-multiplies by Q. This holds at every iteration and at any truncation. Equivariance is therefore exact in real arithmetic for the finite practical algorithm, not only for the idealized msign. Rounding error is a separate matter, which the noise twins of §5 measure.

Case 4: Shampoo. The left accumulator is invariant and the right one is conjugated:

$$L'_t = \sum_{u\le t} G_uQQ^\top G_u^\top = L_t, \qquad R'_t = \sum_{u\le t} Q^\top G_u^\top G_uQ = Q^\top R_tQ.$$

For any spectral function K we have $K(Q^\top RQ) = Q^\top K(R)\,Q$, because $Q^\top RQ$ has the same eigenvalues as R, with eigenvectors rotated by $Q^\top$. Take $K(R) = (R+\lambda I)^{-1/4}$. The damping is covered, since $Q^\top RQ + \lambda I = Q^\top(R+\lambda I)Q$. Hence

$$\begin{aligned}(L'_t+\lambda I)^{-1/4} G'_t\,(R'_t+\lambda I)^{-1/4} &= (L_t+\lambda I)^{-1/4} G_tQQ^\top (R_t+\lambda I)^{-1/4}Q \\ &= \big[(L_t+\lambda I)^{-1/4} G_t\,(R_t+\lambda I)^{-1/4}\big]\,Q. \qquad \square\end{aligned}$$

B.5 Proposition 3.1 (coordinate-wise rules break the gauge)

Proof. Let $\Phi(G)_{ij} = \phi(G_{ij})$ be a memoryless entrywise map. Equivariance for all $Q \in \mathrm{O}(k)$ requires $\Phi(GQ) = \Phi(G)\,Q$ for every G. Steps 1–4 show that this forces ϕ to be linear, and Step 5 draws the consequence. The sign map and Adafactor are then checked directly.

Step 1: two scalar identities. Take $k \geq 2$ and let Q_θ rotate the first two coordinates by θ, acting on a row from the right: $(x,y)\,Q_\theta = (x\cos\theta - y\sin\theta,\ x\sin\theta + y\cos\theta)$. Apply the requirement to a matrix G whose only nonzero row is $(x,y,0,\dots)$. Entry by entry, it reads

$$\begin{aligned} \phi(x\cos\theta - y\sin\theta) &= \phi(x)\cos\theta - \phi(y)\sin\theta, \\ \phi(x\sin\theta + y\cos\theta) &= \phi(x)\sin\theta + \phi(y)\cos\theta, \end{aligned} \tag{B.1}$$

for all x, y, θ.

Step 2: $\phi(0) = 0$. Setting $x = y = 0$ in the first identity of (B.1) gives

$$\phi(0) = \phi(0)\,(\cos\theta - \sin\theta) \qquad \text{for every } \theta.$$

At $\theta = \pi/2$ this reads $\phi(0) = -\phi(0)$, so $\phi(0) = 0$.

Step 3: homogeneity on $[-1,1]$. Setting $y = 0$ in both identities of (B.1) and using $\phi(0) = 0$ gives

$$\phi(x\cos\theta) = \phi(x)\cos\theta, \qquad \phi(x\sin\theta) = \phi(x)\sin\theta.$$

As θ ranges over $[0, 2\pi)$, its cosine and sine sweep all of $[-1,1]$. Hence

$$\phi(cx) = c\,\phi(x) \qquad \text{for every } c \in [-1,1] \text{ and every } x \in \mathbb{R}. \tag{B.2}$$

Step 4: linearity. Take any x and any u with $0 < |u| \leq |x|$. Choosing $c = u/x \in [-1,1]$ in (B.2) gives

$$\phi(u) = \frac{u}{x}\,\phi(x), \qquad \text{that is,} \qquad \frac{\phi(u)}{u} = \frac{\phi(x)}{x}.$$

Any two nonzero reals are both dominated in absolute value by a common third real. So the ratio $\phi(x)/x$ takes a single value c on $\mathbb{R} \setminus \{0\}$, and with $\phi(0) = 0$,

$$\phi(x) = c\,x \qquad \text{for all } x \in \mathbb{R}.$$

This is linearity, with no appeal to additivity.

Step 5: consequence. Any nonlinear fixed entrywise map therefore fails equivariance already for a 2×2 rotation. The standard zero-state first steps of Adam and RMSProp are nonlinear maps of this form (Adam's first step is $-\eta D_\varepsilon(G)$, Proposition A.2), and so are signSGD and Lion's outer-sign update.

The sign map, explicitly. Use the convention $\mathrm{sign}(0) = 0$ and take $\theta = \pi/4$ and the single row $G = (1,1)$, so $GQ_{\pi/4} = (0, \sqrt{2})$. The two orders of operation give

$$\begin{aligned} &\text{sign after rotating:} && \mathrm{sign}(GQ_{\pi/4}) = (0,\,1), \\ &\text{rotating after sign:} && \mathrm{sign}(G)\,Q_{\pi/4} = (1,1)\,Q_{\pi/4} = (0,\,\sqrt{2}). \end{aligned}$$

They disagree, so sign is not gauge-equivariant.

Adafactor. Its second-moment surrogate is rank-one factored rather than entrywise, so Steps 1–4 do not apply and we check it directly. Adafactor keeps the accumulator $v = \mathrm{EMA}(g^2)$ and replaces the per-coordinate second moment by the rank-one reconstruction from its row and column sums:

$$r_i = \sum_j v_{ij}, \qquad c_j = \sum_i v_{ij}, \qquad \hat{v}_{ij} = \frac{r_i\,c_j}{\sum_l r_l}, \qquad \text{update } \frac{g_{ij}}{\sqrt{\hat{v}_{ij}}}.$$

Take a single step, so that $v = g^{\odot 2}$ after bias correction. Adafactor's constant positive factors multiply both sides of the test equally, so we drop them. Under $G \mapsto GQ$ the row sums stay the same, but the column sums become the diagonal of $Q^\top G^\top G Q$, which depends on the basis:

$$r_i = (GG^\top)_{ii}, \qquad c_j = (Q^\top G^\top G\,Q)_{jj}.$$

One rotation already shows the failure. Take $G = \mathrm{diag}(1, \delta)$ with $0 < \delta \neq 1$ and the $\theta = \pi/4$ rotation $Q = 2^{-1/2}\left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right)$. It leaves $r = (1, \delta^2)$ fixed and sends

$$c = (1, \delta^2) \;\mapsto\; \tfrac{1}{2}(1+\delta^2)\,(1,1).$$

The resulting zero-state updates are

$$\Phi(G) = \sqrt{1+\delta^2}\,\mathrm{diag}(1, \delta^{-1}), \qquad \Phi(GQ) = \left(\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}\right),$$

whereas

$$\Phi(G)\,Q = \frac{\sqrt{1+\delta^2}}{\sqrt{2}}\left(\begin{smallmatrix} 1 & 1 \\ -\delta^{-1} & \delta^{-1} \end{smallmatrix}\right) \neq \Phi(GQ).$$

The two sides differ at order 1 for fixed δ. A small scalar stabilizer added to $\hat{v}$ perturbs both sides continuously, so a sufficiently small stabilizer does not reconcile them. Adafactor therefore breaks equivariance through a factored, coordinate-dependent second moment. □

B.6 Proposition A.2 (exact first-step Adam defect)

Proof. For either factor and its gradient G, zero initial state and bias correction give

$$m_1 = (1-\beta_1)G, \qquad v_1 = (1-\beta_2)G^{\odot 2}, \qquad \hat{m}_1 = G, \qquad \hat{v}_1 = G^{\odot 2}.$$

Since $\sqrt{G^{\odot 2}} = |G|$, the first Adam update is exactly $-\eta D_\varepsilon(G)$. In particular, it does not depend on β_1 or β_2 under the stated convention. By Lemma B.2, the gradients in the rotated run are $\widetilde{G}_U = G_U Q$ and $\widetilde{G}_V = G_V Q$. Multiply the rotated run's first iterate by $Q^\top$ on the right. This does not change its product $\widetilde{W}_1$, and it puts both runs in the same coordinates:

$$\widetilde{U}_1 Q^\top = U_0 - \eta D_\varepsilon(G_U Q) Q^\top = U_0 - \eta\left[D_U + E_Q(G_U)\right],$$
$$\widetilde{V}_1 Q^\top = V_0 - \eta D_\varepsilon(G_V Q) Q^\top = V_0 - \eta\left[D_V + E_Q(G_V)\right].$$

The gauge-aligned and reference products are

$$\widetilde{W}_1 = (\widetilde{U}_1 Q^\top)(\widetilde{V}_1 Q^\top)^\top,$$
$$W_1 = (U_0 - \eta D_U)(V_0 - \eta D_V)^\top.$$

Subtracting gives

$$\widetilde{W}_1 - W_1 = \big(U_0 - \eta[D_U + E_Q(G_U)]\big)\big(V_0 - \eta[D_V + E_Q(G_V)]\big)^\top - (U_0 - \eta D_U)(V_0 - \eta D_V)^\top.$$

Expanding both products, the terms $U_0 V_0^\top$, $-\eta U_0 D_V^\top$, $-\eta D_U V_0^\top$ and $\eta^2 D_U D_V^\top$ cancel, and what remains is the identity (A.1). The alternative quadratic expression follows from $D_\varepsilon(GQ)Q^\top = D_\varepsilon(G) + E_Q(G)$ for each factor.

For the k=2 witness of Appendix E, $G_U = G_V = (1,0)$ while $G_U Q = G_V Q = (2^{-1/2}, 2^{-1/2})$. Direct substitution gives the two products displayed there. □

B.7 Theorem 3.2 (structure)

Proof. The theorem is an "if and only if", so there are two directions to prove.

If (⇐): every rule of the form $H(GG^\top)G$ is equivariant. Since $(GQ)(GQ)^\top = GG^\top$,

$$\Phi(GQ) = H(GG^\top)\,GQ = \Phi(G)\,Q.$$

Only if (⇒): every equivariant memoryless rule has this form. Assume $\Phi(GQ) = \Phi(G)\,Q$ for every $Q \in \mathrm{O}(k)$. Let G^+ be the Moore–Penrose pseudoinverse and $P = G^+ G$ the orthogonal projector onto $\mathrm{row}(G)$, the span of the rows of G. We build H from the candidate

$$X(G) := \Phi(G)\,G^+ \in \mathbb{R}^{n \times n}$$

by showing that it reproduces Φ (Step 1), is gauge-invariant (Step 2), and depends on G only through $GG^\top$ (Step 3). The argument works at every rank.

Step 1: $\Phi(G)$ ignores directions outside $\mathrm{row}(G)$. The matrix $R := 2P - I$ is a reflection: it fixes $\mathrm{row}(G)$ and flips its orthogonal complement. So R is orthogonal, and $GR = G$ because $GP = G$. Equivariance with $Q = R$ gives

$$\Phi(G) = \Phi(GR) = \Phi(G)R = 2\Phi(G)P - \Phi(G).$$

Hence $\Phi(G) = \Phi(G)P = \Phi(G)G^+G$, that is,

$$X(G)\,G = \Phi(G). \tag{B.3}$$

Step 2: X is gauge-invariant. For orthogonal Q, at any rank, $(GQ)^+ = Q^\top G^+$ (a standard property of the pseudoinverse). Therefore

$$X(GQ) = \Phi(GQ)\,(GQ)^+ = \Phi(G)\,QQ^\top G^+ = X(G).$$

Step 3: equal Grams lie on one gauge orbit. Let $G_1, G_2 \in \mathbb{R}^{n\times k}$ have $G_1G_1^\top = G_2G_2^\top = E\Lambda E^\top$, where Λ is the $r \times r$ diagonal matrix of the nonzero eigenvalues, all strictly positive ($\Lambda \succ 0$), and E holds the matching eigenvectors. (We write E and F_i here to keep U and V for the factors.) Both G_i then have the same left singular vectors and singular values, and differ only in their right singular vectors:

$$G_i = E\Lambda^{1/2}F_i^\top, \qquad F_i \in \mathbb{R}^{k\times r} \text{ with orthonormal columns.}$$

Complete each F_i to $\widetilde{F}_i \in \mathrm{O}(k)$ and set $Q = \widetilde{F}_1\widetilde{F}_2^\top$. Then $F_1^\top\widetilde{F}_1 = [I_r \quad 0]$, so

$$G_1Q = E\Lambda^{1/2}F_1^\top\widetilde{F}_1\widetilde{F}_2^\top = E\Lambda^{1/2}[I_r \quad 0]\,\widetilde{F}_2^\top = E\Lambda^{1/2}F_2^\top = G_2.$$

With Step 2, $X(G_2) = X(G_1Q) = X(G_1)$.

Step 4: conclusion. By Steps 2 and 3, $H(GG^\top) := X(G) = \Phi(G)G^+$ is well defined, and (B.3) gives

$$\Phi(G) = H(GG^\top)\,G \qquad \text{at every rank.}$$

At $G = 0$ equivariance forces $\Phi(0) = 0$. The theorem pins H down only on $\mathrm{range}(G)$ and only at Gram matrices that actually occur. Away from those, H is unconstrained and any extension serves equally well; the construction picks the natural one, $\Phi(G)G^+$.

Special cases. Full column rank is the case $P = I$, with $G^+ = (G^\top G)^{-1}G^\top$, where Step 1 is automatic. Step 3 then has an explicit gauge, $Q := G_1^+G_2$:

$$Q^\top Q = G_2^\top(G_1G_1^\top)^+G_2 = G_2^\top(G_2G_2^\top)^+G_2 = I_k,$$
$$G_1Q = G_1G_1^+G_2 = P_{\mathrm{range}(G_1)}\,G_2 = G_2,$$

where the last step holds because $\mathrm{range}(G_i)$ and $\mathrm{range}(G_iG_i^\top)$ coincide. In the square invertible case, the polar decomposition $G = (GG^\top)^{1/2}O$ with $O \in \mathrm{O}(n)$ gives H in closed form:

$$\Phi(G) = \Phi\big((GG^\top)^{1/2}\big)\,O = H(GG^\top)\,G, \qquad H(S) = \Phi(S^{1/2})\,S^{-1/2}.$$

Uniqueness follows by evaluating at $G = S^{1/2}$. □

B.8 Proposition A.3 (spectral transfer)

Proof. **Step 1: the formula.** Take the compact SVD $G = \sum_i \sigma_i u_i v_i^\top$. Then $GG^\top = \sum_i \sigma_i^2 u_iu_i^\top$, and functional calculus gives

$$\psi(GG^\top) = \sum_i \psi(\sigma_i^2)\,u_iu_i^\top + \psi(0)\,P_0, \qquad P_0 = I - \sum_i u_iu_i^\top.$$

Here P_0 projects onto $\ker(GG^\top)$. It is nontrivial whenever G is rank-deficient, in particular when $k < n$. Since $P_0G = 0$, that term drops out:

$$\psi(GG^\top)\,G = \sum_i \psi(\sigma_i^2)\,\sigma_i\,u_iv_i^\top = \sum_i h(\sigma_i)\,u_iv_i^\top.$$

Step 2: the instances. These follow by inspection. For support-restricted one-step undamped Shampoo,

$$(GG^\top)^{-1/4}_{\text{supp}}\, G\,(G^\top G)^{-1/4} = \sum_i \sigma_i^{-1/2}\sigma_i\sigma_i^{-1/2} u_i v_i^\top = \sum_i u_i v_i^\top = \operatorname{msign}(G).$$

Step 3: the Lipschitz lower bound. This proves the continuity claim (ii) of Appendix A. Fix $0 < \delta < 1$, a small singular value distinct from Adam's stabilizer ε, and evaluate at

$$G_\pm = \operatorname{diag}(1, \pm\delta),$$

embedded in the top-left 2×2 block. The second singular triple of $G_\pm$ is $(\sigma, u, v) = (\delta, e_2, \pm e_2)$, so

$$\Phi(G_+) - \Phi(G_-) = 2h(\delta)\, e_2 e_2^\top, \qquad G_+ - G_- = 2\delta\, e_2 e_2^\top.$$

The Lipschitz constant of Φ is therefore at least the quotient for this pair,

$$\frac{\|\Phi(G_+) - \Phi(G_-)\|}{\|G_+ - G_-\|} = \frac{|h(\delta)|}{\delta}.$$

This lower bound diverges as $\delta \to 0$ whenever $h(0^+) > 0$, and equals η for $h(\sigma) = \eta\sigma$. □

B.9 Corollary A.5 (deeper factorizations)

Proof. Write $L(U_1, \ldots, U_D) = f(U_1 \cdots U_D)$ and fix an interface i. Call the product of the factors before the interface the prefix P, and the product after it the suffix S:

$$P = U_1 \cdots U_{i-1}, \qquad S = U_{i+2} \cdots U_D, \qquad G = \nabla f(W).$$

The chain rule gives

$$\nabla_{U_i} L = P^\top G\,(U_{i+1} S)^\top, \qquad \nabla_{U_{i+1}} L = (PU_i)^\top G S^\top.$$

Under $(U_i Q,\, Q^\top U_{i+1})$ the represented W is unchanged, hence so is G, and

$$\nabla_{U_i} L \mapsto P^\top G\,(Q^\top U_{i+1} S)^\top = (\nabla_{U_i} L)\, Q,$$
$$\nabla_{U_{i+1}} L \mapsto (PU_i Q)^\top G S^\top = Q^\top (\nabla_{U_{i+1}} L).$$

So each factor's gradient picks up the rotation on the same side as the factor. This is the only property the proofs of Propositions A.1 and 3.1 use, transposed for the left action. Items (a)–(c) follow by rerunning those proofs on each side of every interface, and (d) is Theorem 3.4 applied to all D factors at once. □

B.10 Theorem 3.4 (transfer theorem)

Proof. Let $\theta(t)$ be a locally absolutely continuous solution of $\dot\theta = -\nabla L(\theta)/a(t)$, where $a(t) > 0$ is determined by gauge-invariant trajectory statistics.

Step 1: the new clock. Define

$$\zeta(t) = \int_0^t \frac{du}{a(u)}, \qquad \zeta_{\max} = \int_0^\infty \frac{du}{a(u)}.$$

Then ζ is a strictly increasing bijection onto $[0, \zeta_{\max})$: it is continuous because $1/a$ is locally integrable, and strictly increasing because $1/a > 0$. Let $t(\zeta)$ be its inverse and $\tilde\theta(\zeta) = \theta(t(\zeta))$.

Step 2: the chain rule. For almost every ζ,

$$\frac{d\tilde\theta}{d\zeta} = \dot\theta\, \frac{dt}{d\zeta} = \Big(-\frac{\nabla L(\theta)}{a}\Big) \cdot a = -\nabla L(\tilde\theta).$$

So $\tilde\theta$ is a Carathéodory gradient-flow solution (one that satisfies the ODE almost everywhere) with the same initial condition.

Step 3: the same path. The two trajectories traverse the same set of points under a monotone reparameterization. That set is the whole gradient-flow path when $\zeta_{\max} = \infty$, and otherwise its prefix $\tilde{\theta}([0, \zeta_{\max}))$. So any property of the gradient-flow *path* on the traversed portion holds for the scalar-preconditioned flow.

Step 4: the same limit. Suppose $\zeta_{\max} = \infty$. This holds in particular whenever a is bounded above, since then $\int^{\infty} du/a(u)$ diverges. The reparameterization then also reaches the gradient-flow limit, so the limit-point characterizations of Gunasekar et al. (2017); Arora et al. (2019); Li et al. (2021) transfer as well, under their hypotheses. If $\zeta_{\max} < \infty$, only the finite-time path is shared.

Step 5: uniform over the gauge orbit. The clock a is computed from gauge-invariant statistics, so the reparameterization is itself gauge-independent. The statement therefore holds uniformly over the gauge orbit. □

B.11 Lemma A.6 (momentum preserves the small-step flow)

Proof. Idea. Momentum replaces the current gradient by a weighted average of recent gradients. When the step size is small, the recent iterates all sit close to the current one, so this average is within $O(\eta)$ of the current gradient. The iteration is then Euler's method for $\dot{\theta} = -\nabla L(\theta)/a(\theta)$ with a small error at each step, and Euler's method converges to the flow as $\eta \to 0$. Steps 1–4 make this precise, and Step 5 checks that everything stays inside the equivariant class. The three assumptions of Lemma A.6 enter as follows: (A3) keeps everything in one compact set, (A1) makes ∇L Lipschitz there, and (A2) makes a Lipschitz there and keeps it away from zero.

Step 1: a convex combination. Unrolling $m_{t+1} = \beta m_t + (1-\beta) g_t$ from $m_0 = 0$ gives

$$m_{t+1} = (1-\beta) \sum_{j=0}^{t} \beta^{t-j} g_j, \qquad (1-\beta) \sum_{j=0}^{t} \beta^{t-j} = 1 - \beta^{t+1}.$$

So the bias-corrected quantity

$$\hat{m}_{t+1} = \frac{m_{t+1}}{1-\beta^{t+1}} = \sum_{j=0}^{t} w_j^{(t)} g_j, \qquad w_j^{(t)} = \frac{(1-\beta)\beta^{t-j}}{1-\beta^{t+1}}, \qquad \sum_{j=0}^{t} w_j^{(t)} = 1,$$

is an average of past gradients. The weights $w_j^{(t)}$ are positive and sum to 1, so this is a convex combination, not a scaled sum. That is why no factor of $(1-\beta)^{-1}$ appears below. The unnormalized convention at the end of the proof is the case where one does.

Step 2: the lag is $O(\eta)$. Fix a compact time interval $[0, T]$. By assumption (A3), there is a compact set K that contains the flow and every discrete trajectory on $[0, T]$, for all small η. (Without (A3), the same conclusion holds for the iteration stopped when it first leaves a neighbourhood of the flow.) On K, let Λ be a Lipschitz constant for ∇L, which exists by (A1), and let $\bar{G} = \sup_K \|\nabla L\|$ be the largest gradient norm. By (A2), $\inf a > 0$ on K. Since $\hat{m}$ is an average of gradients, $\|\hat{m}\| \le \bar{G}$, so each step moves the iterate by at most $\eta \bar{G} / \inf a$. After i steps,

$$\|\theta_t - \theta_{t-i}\| \le \frac{i \eta \bar{G}}{\inf a}.$$

Hence

$$\begin{aligned} \left\| \hat{m}_{t+1} - \nabla L(\theta_t) \right\| &= \left\| \sum_{j \le t} w_j^{(t)} \left(\nabla L(\theta_j) - \nabla L(\theta_t) \right) \right\| \\ &\le \Lambda \frac{\eta \bar{G}}{\inf a} \sum_{j \le t} w_j^{(t)} \, (t-j) \\ &\le \Lambda \frac{\eta \bar{G}}{\inf a} \cdot \frac{\beta}{1-\beta}, \end{aligned} \tag{B.4}$$

Line by line: the first line uses that the weights sum to 1, so that $\nabla L(\theta_t) = \sum_j w_j^{(t)} \nabla L(\theta_t)$. The second applies the triangle inequality and the Lipschitz bound to each term, with θ_j lying $t-j$ steps before θ_t. The third bounds the weighted average lag $\sum_j w_j^{(t)} (t-j)$ by the average lag of the

untruncated geometric weights, $\sum_{i\geq 0} i(1-\beta)\beta^i = \beta/(1-\beta)$. So the momentum step differs from the plain preconditioned gradient step by $O(\eta)$, uniformly on $[0,T]$. The constant is proportional to the memory length $\beta/(1-\beta)$, which is held fixed.

Step 3: a perturbed Euler scheme for a Lipschitz field. By (B.4), each step equals the Euler step $\theta_t - \eta\nabla L(\theta_t)/a_t$ plus an error of size $\eta \cdot O(\eta) = O(\eta^2) = o(\eta)$. So the iteration is a perturbed Euler discretization of $\dot\theta = -\nabla L(\theta)/a(\theta)$. The right-hand side of this ODE is Lipschitz on K. Here ∇L is Lipschitz by (A1), and a is Lipschitz and bounded below by (A2). A quotient of Lipschitz functions whose denominator is bounded away from zero is Lipschitz on a compact set. This is the step that boundedness of a alone would not carry.

Step 4: convergence. Euler's method with per-step errors of size $o(\eta)$ on a Lipschitz field converges uniformly on $[0,T]$ as $\eta \to 0$, at first order in η. This is the standard convergence estimate for one-step methods (a discrete Grönwall argument).

Step 5: inside the equivariant class. Gauge-invariance of a is inherited from Theorem 3.4. The averaging is linear in the gradients, hence equivariant by Proposition A.1(1). So the limit is taken within the equivariant class, and the conclusion holds uniformly over the gauge orbit. Setting $\beta = 0$ makes the lag bound vanish and returns the hypotheses of Theorem 3.4.

The unnormalized convention. Step 1 is where the two momentum conventions part company. Under $m \mapsto \beta m + g$ (the buffer used in Case 1 of the proof of Proposition A.1) the weights sum to $(1-\beta^{t+1})/(1-\beta)$ instead of 1. The same computation then returns

$$\dot\theta = -\frac{\nabla L(\theta)}{(1-\beta)\,a(t)},$$

which is the same path under the reparameterization of Theorem 3.4. □

B.12 Proposition A.7 (balancedness)

Proof. Let $G = \nabla f(W)$.

Gradient flow. By the chain rule of Lemma B.2, $\dot U = -GV$ and $\dot V = -G^\top U$, so

$$\tfrac{d}{dt}(U^\top U) = -V^\top G^\top U - U^\top GV, \qquad \tfrac{d}{dt}(V^\top V) = -U^\top GV - V^\top G^\top U.$$

The two derivatives are equal, so $\dot B = 0$. A time reparameterization by $1/a(t)$ multiplies both by the same scalar, so conservation persists for the whole class of Theorem 3.4.

Diagonal preconditioning. Now $\dot U = -D_U \odot (GV)$ and $\dot V = -D_V \odot (G^\top U)$, so

$$\dot B = -\big[(D_U \odot GV)^\top U + U^\top (D_U \odot GV)\big] + \big[(D_V \odot G^\top U)^\top V + V^\top (D_V \odot G^\top U)\big].$$

For example, $D_U \equiv 1$ and $D_V \equiv 2$ give $\dot B = U^\top GV + V^\top G^\top U$, which is nonzero in general, so agreeing up to a factor is not enough. In general, write each preconditioner as a *common* scalar c plus a remainder,

$$D_U = c\,\mathbf{1}\mathbf{1}^\top + E_U, \qquad D_V = c\,\mathbf{1}\mathbf{1}^\top + E_V,$$

where $\mathbf{1}\mathbf{1}^\top$ is the all-ones matrix of the factor's shape. The common-scalar terms cancel, while the remainder obeys

$$\big\|\dot B\big\|_F \leq 2\,\|E_U\|_\infty \|GV\|_F \|U\|_F + 2\,\|E_V\|_\infty \Big\|G^\top U\Big\|_F \|V\|_F,$$

by $\|E \odot Z\|_F \leq \|E\|_\infty \|Z\|_F$, with $\|\cdot\|_\infty$ the entrywise maximum, and Frobenius submultiplicativity. So at bounded factor and gradient norms, the drift vanishes as both factor preconditioners approach the same scalar. □

B.13 Proposition A.9 (greedy vs. equal-rate phase boundary)

Proof. **Reduction to one variable per mode.** Each decoupled mode is a balanced two-factor scalar factorization $w = uv$ with $u = v = \sqrt{w}$ and per-mode loss $\frac{1}{2}(uv - s)^2$, so

$$\partial_u L = v\,(w - s), \qquad \partial_v L = u\,(w - s).$$

The balanced set $u = v$ is invariant under both flows below, since $\dot{u} = \dot{v}$ there. So the reduction to the single variable w is consistent. If one uses a single variable u with $w = u^2$ instead of two factors (the tied model), (i) gives the same ODE with time sped up by a factor of 2, which is a uniform time-rescaling.

(i) Greedy schedule, $h(\sigma) = \eta\sigma$. The gradient flow is $\dot{u} = -\eta\, v(w-s)$, $\dot{v} = -\eta\, u(w-s)$. Using $u^2 = v^2 = w$ at balance,

$$\dot{w} = \dot{u}\, v + u\, \dot{v} = -\eta\,(u^2 + v^2)(w-s) = 2\eta\, w\,(s-w).$$

This is logistic growth, with solution

$$w(t) = s\left(1 + \left(\tfrac{s}{w_0} - 1\right) e^{-2\eta s t}\right)^{-1}.$$

Solving this formula for the time at which the head reaches $w_1(T_1) = (1-\delta)s_1$ gives

$$e^{2\eta s_1 T_1} = \frac{1-\delta}{\delta} \cdot \frac{s_1 - w_0}{w_0} \le \frac{1-\delta}{\delta} \cdot \frac{s_1}{w_0}. \tag{B.5}$$

For the tail, dropping the 1 in the denominator of the logistic solution gives an upper bound:

$$w(t) \le \frac{s\, w_0}{s - w_0}\, e^{2\eta s t} \le 2 w_0\, e^{2\eta s t} \qquad \text{for } w_0 \le s/2.$$

Apply this to mode 2 at $t = T_1$. Since $\rho = s_2/s_1$, we have $e^{2\eta s_2 T_1} = \left(e^{2\eta s_1 T_1}\right)^{\rho}$, and inserting (B.5) gives

$$w_2(T_1) \;\le\; 2w_0\left(e^{2\eta s_1 T_1}\right)^{\rho} \;\le\; 2\left(\tfrac{1-\delta}{\delta}\right)^{\rho} s_1^{\rho}\, w_0^{1-\rho} \;=\; C_{\delta,\rho}\, s_1 \left(\tfrac{w_0}{s_1}\right)^{1-\rho}.$$

(ii) Equal-rate schedule, $h \equiv \eta$. This flow replaces the update of every factor with η times the matrix sign of its gradient. In the scalar case, for $0 < w < s$,

$$\dot{u} = -\eta\, \mathrm{sign}\big(v(w-s)\big) = \eta, \qquad \dot{v} = \eta,$$

because $v > 0$ and $w - s < 0$, so the sign is -1. So both factors grow linearly, and

$$\sqrt{w(t)} = \sqrt{w_0} + \eta t \qquad \text{until } w \text{ reaches the threshold } s.$$

There the gradient becomes zero. Since $\mathrm{sign}(0) = 0$, which also follows from $\mathrm{msign}(0) = 0$ by Proposition A.1(3), the mode stops. So the capped dynamics are the stopped flow: once a mode reaches s_i, it stays there. Mode i fits at t_i, and the head reaches $(1-\delta)s_1$ at T_1, where

$$t_i = \frac{\sqrt{s_i} - \sqrt{w_0}}{\eta}, \qquad T_1 = \frac{\sqrt{(1-\delta)s_1} - \sqrt{w_0}}{\eta}.$$

For $\delta < 1 - s_2/s_1$ we have $(1-\delta)s_1 > s_2$, hence $t_2 < T_1$ and $w_2(T_1) = s_2$ exactly, with no dependence on w_0. □

C Experimental details

The two reported metrics. Recovery measures how far the fitted matrix W is from the ground truth X^*, as a fraction of the ground truth's size. Effective rank measures how spread out the singular values of W are: a matrix whose energy is concentrated in a few singular values has low effective rank, while one whose singular values are all comparable has high effective rank. Formally,

$$\mathrm{rec}(W) = \|W - X^*\|_F \,/\, \|X^*\|_F\,, \qquad \mathrm{erank}(W) = \exp\Big(-\textstyle\sum_i \pi_i \log \pi_i\Big),$$

with $\pi_i = \sigma_i / \sum_j \sigma_j$ and $0 \log 0 := 0$ (Roy & Vetterli, 2007). Throughout, *recovery* means the error $\mathrm{rec}(W)$, so smaller is better.

The sensing runs use full overparameterization $k{=}n$, initialization scale 10^{-3}, no weight decay, and stop at interpolation, meaning a training loss below 10^{-7}.

Table 1: Selected rates minimize recovery error among runs that reach interpolation. These grids produce Table 2 on 40×40 rank-3 matrix sensing. Selection and evaluation use the same three seeds (§4).

method	grid	selected
GD	$\{10^{-2}, 3\times10^{-2}, 10^{-1}, 3\times10^{-1}\}$	10^{-2}
scalar-Adam	$\{10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}\}$	10^{-3}
Muon	$\{3\times10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}\}$	10^{-1}
Shampoo	$\{3\times10^{-2}, 10^{-1}, 3\times10^{-1}, 1\}$	3×10^{-2}
Adam	$\{10^{-3}, 10^{-2}, 3\times10^{-2}\}$	10^{-2}
RMSProp	$\{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$	10^{-4}
signum	$\{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$	3×10^{-3}
Lion	$\{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}$	3×10^{-4}
Adafactor	$\{3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}$	3×10^{-4}

Sensing task (§4, §6).

- *Target and measurements.* $X^* = U^*V^{*\top}$, $U^*, V^* \in \mathbb{R}^{40\times3}$ i.i.d. Gaussian scaled by $1/\sqrt{3}$; $m = 462 = 2\,\mathrm{dof}$, $\mathrm{dof} = 3(80-3) = 231$; A_i i.i.d. standard Gaussian; $\mathrm{loss} = \mathrm{mean}_i(\langle A_i, W\rangle - y_i)^2$.
- *Factors.* $U, V \in \mathbb{R}^{40\times40}$ initialized $\mathcal{N}(0, 10^{-6})$ entrywise (10^{-3} scale).
- *Budget.* 2×10^4 steps (extended where noted); interpolation bar 10^{-7} (checked every 200 steps); 3 paired seeds {42, 123, 456}.

Optimizer hyperparameters:

- *Adam, scalar-Adam.* $\beta = (0.9, 0.999)$, $\varepsilon = 10^{-8}$.
- *RMSProp.* Classical form, second-moment decay 0.99 and no first moment, $\varepsilon = 10^{-8}$.
- *Muon.* Momentum 0.9, Newton–Schulz 5 iterations, coefficients $(\gamma_1, \gamma_2, \gamma_3) = (3.4445, -4.7750, 2.0315)$ (Jordan et al., 2024).
- *Shampoo.* Accumulated (non-EMA) L, R, damping $\lambda = 1$, inverse-root refresh every 20 steps.
- *Adafactor.* Row/column EMA factored second moment.
- *Lion.* $\beta_1 = 0.9$ inner, 0.99 EMA.
- *signum.* Sign of an EMA momentum with $\beta_1 = 0.9$ (Adam's β_1) and no second moment, no clipping.

Learning-rate grids appear in full in Table 1, which also shows the rate each method's best-recovery rule selects. Recovery-versus-rate curves for five reference methods are in Appendix D.

The dial's fixed-rate arm (§4) re-runs the full p-sweep at each single rate of the shared grid $\{10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}\}$, same update rule, seeds, and interpolation bar (`experiments/precond_dial_fixed_lr.py`).

Both dial arms use a 3×10^4-step budget (`experiments/precond_dial_scalar_check.py`), longer than the zoo's 2×10^4.

Three code paths implement the dial, and they differ in how they place ε. None of the differences reaches the digits we report at $\varepsilon = 10^{-8}$, but we record them here for reproducibility. The CPU implementation follows Algorithm 1 line by line. The GPU port behind the n=128 dial rows (in the released code) omits the outer ε for $0 < p < 1$ and divides by $\bar{s} + \varepsilon$ rather than $\bar{s} + 2\varepsilon$ at p=0. The FlowAdam implementation, used for the FlowAdam-p rows, folds the bias correction into the step size at p=1 and adds ε to $\sqrt{v}$ before that factor. This is TensorFlow's Adam convention rather than Algorithm 1's, giving an effective $\varepsilon/\sqrt{1-\beta_2^t}$.

The FlowAdam implementation also differs in its scalar convention. Its shared scalar for $p < 1$ *defaults* to the geometric mean rather than Algorithm 1's RMS. Every FlowAdam-p row reported here passes `precond_scalar='rms'` explicitly and sits on the RMS dial (`experiments/flowadam_upgrade_rms.py`, which also re-runs the geometric-mean rows as a bridge to the legacy numbers). Appendix E.22 therefore reads their gain against the RMS dial's 0.201, the same convention on both sides. The geometric-mean endpoint is 0.229, and the geometric-mean variant itself is in Appendix E.

Attention task (§5).

- *Task.* Modular addition modulo 47 on input sequences of the form $(a, b, =)$, with 40% of the data used for training (fixed split).
- *Model.* A 2-layer transformer with 4 attention heads, model dimension $d_{\text{model}} = 64$, using RMSNorm (Zhang & Sennrich, 2019) in pre-normalization configuration and SiLU-activated (Elfwing et al., 2018) MLPs of width 256. The model omits biases and does not apply query–key normalization (qk-norm), so the gauge holds exactly.
- *Training.* Full-batch optimization over 1500 steps, executed deterministically on CPU (`torch.use_deterministic_algorithms`).
- *Gauge.* An orthogonal transformation A_h per head, derived via QR decomposition of a Gaussian matrix, applied to the row-blocks of W_Q and W_K.
- *Learning rates.* Adam uses 10^{-3}; SGD with heavy-ball momentum (0.9, unnormalized, as for Muon's momentum here) uses 0.5; scalar-Adam is set to 3×10^{-3}, while Muon uses 0.02 on matrix parameters, with the standard hybrid applying Adam at 10^{-3} to the embeddings and the output head. These are each family's usual default at this scale and were not selected by a grid search. Nothing here compares one method's rate against another's. Each twin pair trains at a single rate, so the split inside a pair cannot be an artifact of the rate.

Drift measures how much two model copies disagree in function space. It is defined as the relative ℓ_2 logit distance on the validation set, $\|Z - Z'\|_F \,/\, \frac{1}{2}(\|Z\|_F + \|Z'\|_F)$, where Z and Z' are the validation logit matrices of the two copies. The symmetrized denominator keeps the ratio bounded and comparable across scales.

QK drift is computed the same way, but on $W_Q^\top W_K$ for each attention head rather than on the output logits. Per-head ratios are averaged with equal weight, counting every head in every layer once. The 56% quoted in §5 is that mean. It is a relative distance between invariants, not a drop in any performance measure.

Where the numbers come from. Tables 2, 12, 13, 14 and 19 are typeset from run records, so that all displayed quantities have been regenerated from them independently of the text. In particular, the zoo reads `optimizer_zoo_bias.jsonl`; the ladder, its dispersion and its holdout cut read `zoo_n{64,96,128,192,256}.jsonl`; and Table 19's matched-loss comparison reads `{pavia,indianpines}_obscenter.jsonl` for gradient descent and `scalaradam_lr_{pavia,indianpines}_obscenter.jsonl` for the learning-rate grid from 3×10^{-3} down to 10^{-6}, the sweep of §6. All data is released with the code, and our scripts can be used to typeset the tables and reproduce all displayed quantities, applying the selection rules of Appendix D: learning-rate selection, the treatment of non-interpolating runs, and the seeds held out of selection. All numbers in Tables 9 and 19, for both scenes and both densities, were produced by a single experimental harness, running in `float64` precision on GPU and using the train-only selection rule (Appendix D.5). The two training-loss levels quoted for these tables, 3×10^{-5} and $\leq 10^{-5}$, are the same runs read at two depths of fit.

Dataset details. The hyperspectral image of the Indian Pines area of north-western Indiana was taken by the AVIRIS sensor and made available as an archive with 220 spectral bands (Baumgardner et al., 2015). We use the widely-adopted corrected version with 200 bands that excludes the water-absorption bands (Graña et al., 2021), and take 2000 pixels from it. The hyperspectral image of Pavia University was taken by the ROSIS sensor and is distributed by the same source; we take 2000 pixels in its 103 spectral bands. For matrix completion, both datasets are reshaped so that each pixel forms a row. To construct our task, we draw 2000 rows uniformly at random from the whole spatial grid with a fixed seed; we emphasize that these are not taken from a connected patch, which would leave a different singular spectrum. To construct a completion task from the resulting matrix $X \in \mathbb{R}^{N \times B}$, with $N = 2000$ rows and B bands, we observe a random subset $\Omega \subseteq [N] \times [B]$ of entries and aim to learn the whole matrix X from its observed part X_Ω. For evaluation, we report the root-mean-square error of the completion on the held-out entries Ω^c. We report the mean and standard deviation over four seeds, each of which draws its own observation mask and initialization.

Note that mean-shifting the dataset is important here. While this is common practice in hyperspectral imaging, we highlight that we perform the shift inside the loop over random seeds and densities using

training-mask entries only, so that no held-out entry enters the preprocessing: for each band $j \in [B]$, we estimate the mean from the rows $i \in [N]$ such that $(i, j) \in \Omega$, and shift accordingly. Bands with no observed entries remain unshifted (that is, are assumed to have zero mean). The reference intrinsic rank and the spectral tails are read off the complete matrix and are diagnostics, not quantities any method sees.

Details on the hyperspectral matrix-completion task. We solve the entrywise matrix-completion loss on the hyperspectral data at factorization rank 48, using the train-only selection rule of Appendix D.5 and initialization scale 10^{-2}. We do not apply weight decay in these experiments. Table 9 displays two data densities corresponding to m/dof_{24} ratios of ≈ 1.15 and ≈ 1.9, which are densities of ≈ 0.15 and ≈ 0.25 on Indian Pines and ≈ 0.28 and ≈ 0.46 on Pavia University. Here, the degrees of freedom dof_{24} are computed at rank 24, at which point we find that the empirical singular-value spectra have settled. At that rank the singular-value tail carries about 0.10 of the Frobenius norm of the Indian Pines data. This is an amplitude, like τ, not an energy fraction. Checkpoints are compared at matched training-set loss, in the range 3×10^{-3} to 10^{-5}. The experiments were repeated with four random seeds, two for the learning-rate selection and two independent runs, with results aggregated across all four. Since the selection procedure only uses the training set, the learning rate itself carries no test-set information. That is a statement about rate selection alone, not about the preprocessing, the rank cap or the choice of scenes.

The model is over-parameterized to rank 48, which exceeds the reference intrinsic rank of 24, but the cap is not inactive. Effective rank grows faster for Adam at larger model ranks: moving from rank-48 to rank-96 models roughly doubles Adam's effective rank but has a much smaller effect on gradient descent's effective rank, amplifying Adam's disadvantage at rank 96 (Appendix D.7). So the cap leaves room for the implicit bias to do the selecting, and the rank-48 numbers are the conservative side of that comparison.

FlowAdam variants. For the FlowAdam runs we try the clipping modes `percoord` and `globalnorm` as well as different values of the preconditioner exponent `precond_power`, following Singh & Sheel (2026). Under `percoord` each entry is clipped to $[-1, 1]$, and under `globalnorm` the whole velocity is rescaled to norm at most c, the threshold of Appendix E.22. The budget of FlowAdam's runs is 3×10^4 steps in `experiments/flowadam_upgrade.py`, while it is 2×10^4 in the zoo, and the early-stopping audit of Appendix D.9 goes up to 1.2×10^5. We use the same three paired seeds and the same 10^{-7} interpolation threshold as the zoo. Two configurations stop short of that threshold within the budget. FlowAdam-p=0 reaches a training loss of 7×10^{-6}, and continued to a true 10^{-7} it reads the same 0.1691 (Appendix D.9). FlowAdam with global-norm clipping reaches a training loss of 6×10^{-7} with p=1, and its recovery error of 0.219 is a deep fit rather than an interpolating reading (Appendix E.22). The 3–6% comparison of Appendix E.22.1 ran in the dense, well-conditioned regime, tuned versus tuned, with the same-placement L2 control (an identical L2 term in the loss for the Adam baseline) across five paired seeds.

Discrepancies between tables. For completeness, we give additional details that may help explain minor discrepancies among values repeated across tables. They come from different learning-rate grids, different numbers of seeds and different step limits, and none of these reflects a difference in outcome. The zoo uses three paired seeds, and every experiment uses at least three except the single-seed noise-twin and merge rows of Appendix E.8 and the two-pair equivariant attention twins of Appendix D.4; Table 9 uses four seeds. Two observations stand out. First, the zoo's selection grids stop above the smallest interpolating learning rate, while the diagnostic grids of Appendix D.3 go below it, so a flow-limit reading can fall under the corresponding Table 2 row (Remark A.8), as Table 5 shows for GD and scalar-Adam. Second, the dial evaluates p on a single shared grid that differs from each method's own grid, so its endpoint values reflect the rates that grid selects: at p=1 it reads 0.570 (at 3×10^{-3}, a rate the zoo's Adam grid lacks) against the zoo's 0.5734, and at p=0 with n=128 it reads 0.40 against the ladder's 0.345.

In summary, we find these discrepancies to be minor and qualitatively consistent: finer evaluation grids lead to slightly better performance, that is, slightly lower errors, but monotonicity in p is preserved across tables. The remaining discrepancies, those below 0.01, are due to the granularity of the evaluation grids, and we do not highlight them individually.

Precision. The matrix-sensing runs in §4 and their controls are CPU-based and use PyTorch's default float32 precision. The §5 attention twins of Table 7 are CPU-based and use PyTorch's default float32 precision. Table 7 reports these float32 runs and Table 8 reports a rerun at higher (float64) precision, on GPU, for the same experiment. The higher-precision GPU-based float64 experiments in our paper are the Appendix D.10 H100 replication ladder, the ten-seed tail sweep of Table 15, Appendix D.4 at-scale twins, the LoRA twins of Table 17, the merges of Appendices E.8 and E.10, the dial at scale (Appendix E.13), and the §6 hyperspectral experiments. Table 16 is CPU-based float64. For the GPU runs we enabled PyTorch deterministic kernels. The GPT-2 fine-tunes of Appendix E.11 run on GPU in float32, with the gauge matrix built and applied in float64.

Hardware. CPU-based experiments were conducted on an Apple M1. Every GPU run listed above used a single NVIDIA H100 per job.

D THE CONTROL BATTERY

All of the key results of the main paper were re-evaluated for a battery of control setups. Overall, the results either held or sharpened the conclusions. Exceptions are noted below and discussed more thoroughly in each subsection.

Results that held. Class ordering held under schedule symmetrization (Appendix D.1), the learning-rate (LR) axis and LR choice held (Appendices D.2–D.3), the trend with respect to initialization scale held (Table 6), the attention twins were a robust effect for various seeds, model depths, and noise levels (Appendix D.4), and the interpolation-bar audits held (Appendix D.8).

Results that sharpened conclusions. These are the achievability test of Appendix D.2 (which methods reach error below 0.3 at some rate) and the sensitivity of RMSProp to its schedule (Appendix D.8).

Claims that did not hold. Four claims that didn't hold up were modified accordingly in the text. These are:

1. Long-anneal signum staying in the coordinate-wise cluster (Appendix D.11).
2. Fixed-damping Shampoo staying below Adam beyond n=40 (Appendix D.10). It exceeds Adam at four of five sizes.
3. The three-seed crossover, $\tau^* \approx 0.35$ (Appendix D.1). Ten random seeds shift its location to $\tau^* \approx 0.2$.
4. GD outperforming Adam on held-out error (Appendix D.5). On an expanded grid of real-data experiments, with each method at its own best rate, Adam resolvably outperforms GD in 1 of the 4 cells on the two main scenes.

D.1 SCHEDULE SYMMETRIZATION

Algorithms with a constant update norm (e.g., Muon, signum, Lion) require LR decay in order to get to interpolation. Therefore, it is important to check that Muon's performance gains are not explained by using LR decay while other methods don't use it. We ran two arms of experiments to investigate:

- In the first, we run all methods with both constant and cosine schedules, stopping at interpolation. This arm of the study allows us to investigate whether Muon does better than Adam and Shampoo with the same decayed LR schedule.
- In the second, we run all methods for the entirety of 2×10^4 steps using cosine decay (the full-horizon cosine schedule). This arm of the study allows us to investigate whether other methods can match Muon if given equally small steps towards the end of training.

Results are in Table 3. In the first arm of the study, Muon does not manage to reach the threshold of interpolation when run with constant LR. Therefore, this arm of the study does not apply to Muon,

Table 2: The two optimizer groups separate on recovery error. Matrix sensing uses a 40×40 rank-3 target, $m = 2\times$dof measurements, and no weight decay. All runs interpolate across three paired seeds. Individual runs also separate: worst equivariant 0.3134, best coordinate-wise 0.3366. Recovery is $\|W - X^*\|_F / \|X^*\|_F$ (mean $\pm$ standard deviation). Erank is effective rank. Bal is $\left\|U^\top U - V^\top V\right\|_F$. Other entries are means. "cosine" denotes learning-rate decay. Anchors name the work that first reported each bias: GD (Gunasekar et al., 2017), Adam (Wilson et al., 2017). The nuclear-norm reference is feasible, not certified optimal (Appendix E.12).

	method	recovery ↓	erank	bal	train loss	notes
	nuclear-norm ref.	*0.0335*	*3.30*	—	—	convex reference, not an optimizer
equivariant	Muon (cosine)	**0.0000** $\pm$ 0.0000	2.95	1.65	5.7×10^{-8}	near-exact, 6.8×10^{-6}
	GD	0.1312 $\pm$ 0.0374	4.51	0.06	6.5×10^{-8}	Gunasekar anchor
	scalar-Adam (p=0)	0.2010 $\pm$ 0.0381	5.43	0.14	3.5×10^{-8}	equivariant; flow proxy
	Shampoo	0.2856 $\pm$ 0.0393	6.95	1.47	4.5×10^{-8}	
coord.-wise	Lion (cosine)	0.4248 $\pm$ 0.0849	10.58	7.35	8.0×10^{-8}	
	signum (cosine)	0.4454 $\pm$ 0.0695	7.83	1.79	6.5×10^{-8}	low-rank-but-wrong
	RMSProp (cosine)	0.5266 $\pm$ 0.0828	12.80	4.74	6.0×10^{-8}	needs decay, App. D.8
	Adafactor	0.5430 $\pm$ 0.0831	10.72	3.40	5.6×10^{-8}	factored diag. breaks it
	Adam	0.5734 $\pm$ 0.0554	14.37	5.37	1.2×10^{-11}	Wilson anchor

Table 3: Changing the schedule leaves recovery nearly unchanged for the interpolating runs. Entries are mean recovery errors over three sensing seeds, with the best recovery rate selected separately for each cell. Constant and cosine runs stop at interpolation. Full-horizon runs continue for 2×10^4 steps.

method	constant	cosine	cosine, full horizon
GD	0.1312	0.1312	0.1312
Adam	0.5734	0.5734	0.5764
scalar-Adam	0.2010	0.2010	0.2010
Muon	(0.0084†)	0.0000	0.0000
Shampoo	0.2856	0.2856	0.2856

†Muon does not interpolate at a constant learning rate. Its training-loss floor is 6×10^{-2}, so this recovery value is not an interpolating result.

Table 4: The two optimizer groups still separate under shared cosine decay. Entries average three sensing seeds, using each method's best recovery rate on its own grid. Runs stop at training loss below 10^{-7}. Lower recovery is better. Steps counts updates to the stopping threshold. Selection details: §4 and Appendix D.3.

method	class	recovery	erank	selected lr	steps	train loss
Muon	equivariant	0.0000	2.95	0.1	19800	5.7×10^{-8}
GD	equivariant	0.1312	4.51	0.01	3000	8.0×10^{-8}
scalar-Adam	equivariant	0.2010	5.43	0.001	3200	5.4×10^{-8}
Shampoo	equivariant	0.2856	6.95	0.03	2533	5.4×10^{-8}
Lion	coordinate-wise	0.4248	10.58	0.0003	19000	8.0×10^{-8}
signum	coordinate-wise	0.4454	7.83	0.003	19600	6.5×10^{-8}
RMSProp	coordinate-wise	0.5266	12.80	0.0001	18133	6.0×10^{-8}
Adafactor	coordinate-wise	0.5430	10.72	0.0003	2667	4.1×10^{-8}
Adam	coordinate-wise	0.5734	14.37	0.01	400	1.3×10^{-11}

and we can only vary the schedule for GD, Adam, scalar-Adam and Shampoo. The overall ordering of methods is the same in both arms.

We also reran the phase diagram experiment from §6 using cosine LR decay (the three-seed CPU sweep, whose crossover sits at $\tau^* \approx 0.35$ with and without decay). GD, Adam and Shampoo recover within noise of their constant-rate values at all values of τ, and Muon shifts by no more than 0.02. We find that the relative performance of the methods is the same when LR decay is introduced, with Adam performing the worst at all six values of τ. At the highest value of τ we test, all methods match up to seed variance, so their ordering at this value remains unclear.

Ten seeds instead of three. With more random seeds, 10 instead of 3, the crossover moves to a lower tail amplitude: $\tau^* \approx 0.35$ with three seeds but $\tau^* \approx 0.2$, or about 4% tail energy, with ten. That

is a shift of over twice the standard error, so the three-seed estimate was not stable. All conclusions, such as the ranking of methods and its invariance under different schedules, are unchanged.

For each seed, the crossover is determined by interpolating (linearly) the gap in recovery of Muon compared to GD through zero as a function of τ, between the two adjacent τ values on either side of zero. Averaging this quantity over the ten seeds gives mean 0.200 with standard deviation 0.067, hence a standard error of the mean of 0.021. The 95% (two-tailed) t-interval for the mean with 9 degrees of freedom is $[0.15, 0.25]$. The crossover τ^* varies widely across seeds, from 0.108 to 0.288. The interval is on the mean crossover, not a claim that any individual seed crosses inside it.

A separate sweep at n=128. We performed a separate sweep at n=128, reported only qualitatively. The reason for only providing it qualitatively is that Adam's error exceeds Shampoo's at τ=0 on this sweep, even though the opposite happens on the per-size ladder grid used for Table 12, with Shampoo untuned in both. We expect that the difference in procedure, together with the use of a slightly different grid for learning-rate multipliers, explains the difference between these results. On this sweep Adam is not always the optimizer that does worst, contrary to what happens at n=40, although GD $<$ Shampoo $<$ Adam in the low-τ part of the grid. For $\tau \geq 0.25$, Shampoo's error exceeds Adam's by less than 0.01, comparable to the variability between seeds for Adam and Shampoo.

The separate sweep above uses a grid with two learning-rate multipliers, different from the ladder grid of Table 12 described in Appendix D.10. Notably, Muon does not recover on the former: it reads 0.54 instead of 0 at τ=0 for n=128. For this reason, the crossover for Muon is not shown. On the wider ladder grid, Muon's rates at n=128 include one at which Muon does recover, as can be seen in Table 12. This is why we see different behavior in these two experiments.

D.2 The learning-rate axis

Table 5 shows recovery for several step sizes for each method. We do not advocate picking each method's learning rate based on the best recovery, because that would be an outcome-based criterion. We use it in §4 only to show each method at its best; the conclusion rests on the whole curve. Nor do we advocate picking the step size for which each method interpolates the fastest. That ignores the bias and finds each method's most bias-corrupted learning rate, as with Muon, where the recovery at the fastest-interpolating learning rate is 0.91. Neither criterion by itself motivates any conclusion about the methods' implicit biases. Examining recovery across a range of step sizes, however, can test for an asymmetry in what each group can reach at any rate.

We see that every equivariant method gets error below 0.3 at some learning rate in the grid, while none of the coordinate-wise optimizers can at any rate on its grid within this horizon. This highlights that while learning rate and schedule affect recovery within a method's class, equivariance, not the rate, decides which group of recoveries a method lands in.

Adam has nearly constant recovery for five interpolating learning rates spanning two orders of magnitude, indicating that no learning rate on this grid gives Adam the low-rank bias. Gradient descent and scalar-Adam do better with smaller but still interpolating learning rates, as predicted by common-scalar flow models for gradient descent (for scalar-Adam this is empirical; see the scope notes after Lemma A.6). Muon has near-exact recovery at larger learning rates, which still interpolate after decay, and worse recovery at smaller learning rates. This is influenced by the learning-rate schedule used with Muon. It is consistent with equal-rate spectral growth (Kang et al., 2026), as well as with the effect of a fixed number of Newton–Schulz iterations near small singular values.

D.3 Learning-rate curves and selection rules

Table 5 includes some smaller LRs than the selection procedure used, in order to probe the flow limit (the $\to$fl rates). This is why it does not agree with Table 2: the $\to$fl probing LRs are not necessarily the same as the LRs picked by our best-recovery criterion from §4. To ease comparison, however, for each optimizer in our zoo we do include the LR that is chosen by the best-recovery criterion. Details about the LR grids are given in Appendix C. The remaining coordinate-wise optimizers (RMSProp, Lion, Adafactor, and signum) were swept over different LR grids, and are not reported in Table 5. They all belong to the poor-recovery (≥ 0.42) group under both criteria.

Table 5: Adam's recovery stays nearly flat across learning rates, while the other methods are more sensitive. Entries are mean recovery errors over three sensing seeds. "→fl" marks the smallest tested interpolating rate. "n/i" means the run does not interpolate. "div" means it diverges.

method		learning-rate grid				
GD	lr	$0.003^{\to\text{fl}}$	0.01	0.03	0.1	
	recovery	0.113	0.131	0.437	div	
Adam	lr	$0.0003^{\to\text{fl}}$	0.001	0.003	0.01	0.03
	recovery	0.560	0.576	0.570	0.573	0.581
scalar-Adam	lr	$0.0003^{\to\text{fl}}$	0.001	0.003	0.01	
	recovery	0.165	0.201	0.256	0.357	
Muon (cosine)	lr	0.003	0.01	0.03	0.1	
	recovery	0.915	0.748	0.000	0.000	
Shampoo	lr	0.01	0.03	0.1	0.3	
	recovery	(0.881 n/i)	0.286	div	div	

Table 6: Larger initialization worsens GD and scalar-Adam recovery, while Muon remains near-exact. Entries are mean recovery errors over three sensing seeds. "init" is the factor initialization scale.

init	GD	Adam	scalar-Adam	Muon	Shampoo
10^{-3}	0.131	0.573	0.201	0.000	0.286
3×10^{-3}	0.203	0.575	0.260	0.000	0.352
10^{-2}	0.318	0.584	0.358	0.000	0.575

Selection-rule stress test. Across all nine methods, if one chooses the LR that performs best in terms of recovery, using a uniform cosine schedule, the final classification is correct in all cases ($9/9$ accuracy). However, "fastest-interpolating", if interpreted in a bias-blind way, would perform worse ($8/9$): Muon would be marked as failing with recovery 0.91.

"Fastest-interpolating", for a horizon of 2×10^4 iterations, means reaching 10^{-7} in the least number of iterations, not the largest LR that manages to interpolate. For Muon it corresponds to the smallest LR in the grid (0.003), while it succeeds for the two largest ones with near-exact recovery, as can be seen in Table 5.

Initialization scale. Table 6 sweeps the initialization scale for the five reference methods.

D.4 Attention twins across seeds, initialization draws, and noise levels

We use the attention twins setup from §5: two models that start from the same initialization, except that the query and key weights are rotated by a random orthogonal matrix in one of them (a "gauge rotation"). We also create a noise twin that starts from a small additive perturbation of the initialization, in the same basis, to differentiate divergence due to symmetry-breaking from floating-point instabilities.

We report the results of Adam for six pairs of attention-gauge twins. These were the product of two initialization seeds and three independent draws of the random gauge, all on the same deterministic CPU setting as §5. At step 0, the twins are identical up to $\sim 2 \times 10^{-7}$ drift (the floating-point rounding floor of the computation), but by step 1 they have drifted by 3.4 to 5.4×10^{-3}. This is consistent with the 2L/d=64 setup in Table 8, which was run in `float64` precision on GPU. At the end of training, drift is 0.69–0.80 but validation accuracy is comparable ($|\Delta\text{acc}| \leq 0.0023$). While the two models may have similar performance, it's unclear if this means anything strong, such as convergence to a common minimizer manifold. They certainly do not end at the same parameters, and they implement different functions.

For SGD and scalar-Adam, we show results on two different (seed, draw) pairs. The maximum final drifts are 3.0×10^{-4} for SGD and 3.5×10^{-6} for scalar-Adam.

Next, we compare the final gauge drifts to those of the corresponding noise twins. In all cases, noise-twin tests are done in the same basis as the gauge twins and for noise amplitudes in the range

Table 7: Adam's rotated twin separates at the first step, far above its noise control. Entries are relative logit distances on validation inputs for modular addition, computed in CPU `float32`. Gauge twins differ by a rotation. Noise twins receive an additive perturbation. $A{=}I$ is an identical rerun. Step 0 shows the rounding floor. Muon's later drift tracks its noise twin, rather than breaking equivariance (Proposition A.1).

optimizer	twin type	step 0	step 1	step 100	step 1500 (final)
Adam	gauge	1.8×10^{-7}	$\mathbf{3.6 \times 10^{-3}}$	6.5×10^{-1}	7.7×10^{-1}
Adam	$A{=}I$	0	0	0	0
Adam	noise (10^{-7})	2.6×10^{-7}	2.9×10^{-7}	1.7×10^{-5}	1.6×10^{-5}
SGD	gauge	1.8×10^{-7}	2.9×10^{-7}	3.7×10^{-5}	2.0×10^{-5}
scalar-Adam	gauge	1.8×10^{-7}	1.6×10^{-7}	4.1×10^{-6}	4.3×10^{-6}
Muon	gauge	1.8×10^{-7}	2.2×10^{-7}	2.3×10^{-2}	8.5×10^{-1}
Muon	noise (10^{-7})	2.6×10^{-7}	2.9×10^{-7}	4.0×10^{-2}	8.7×10^{-1}

Table 8: Adam's first-step split persists in larger models and on text. Entries are relative logit distances in GPU `float64`. Each configuration uses six Adam initialization–rotation pairs and two per equivariant method. "eq., step 1" is the largest distance across SGD, scalar-Adam, and Muon. "ratio" divides the smallest Adam first-step gauge distance by its noise control's distance, using a 100× stronger perturbation. $A{=}I$ reruns remain identical. Text twins share minibatches.

config	Adam gauge, step 1	eq., step 1	Adam, final	ratio (step 1)
mod-47, 2L, $d{=}64$	$4.0\text{–}5.7 \times 10^{-3}$	5.4×10^{-16}	0.69–0.78	93×
mod-97, 4L, $d{=}128$	$8.2\text{–}11 \times 10^{-3}$	7.3×10^{-16}	0.66–0.75	158×
mod-97, 6L, $d{=}256$	$6.5\text{–}7.2 \times 10^{-3}$	1.1×10^{-15}	0.61–0.64	94×
text (char-LM), 6L, $d{=}256$	$2.2\text{–}2.4 \times 10^{-3}$	8.0×10^{-16}	0.36–0.37	39×

of 10^{-7} to 10^{-5}, resulting in final drifts from 1.6×10^{-5} to 1.0×10^{-3} across this range. We define a conservative "structural ratio", computed as the ratio between the minimum final gauge drift and the final drift of a noise twin 100× stronger than the 10^{-7} one. The observed structural ratio is 662×, and we claim $> 600\times$.

For the case of Muon with seed 42, the mean value of the chaos measure $|\log_{10}(\text{gauge}/\text{noise})|$, averaged over iterations past step 100, is 0.04 decades. Given this low drift ratio, we may view the gauge twin as a kind of perturbed floating-point twin, with the divergence reflecting the chaotic nature of Muon rather than any basis-dependence.

Scaling the twin diagnostic (GPU, `float64`). Results for larger models, and on text, are given in Table 8. The "ratio" column reports the onset ratio at step 1 rather than at the final step: Adam's lowest step-1 gauge split over the value of our 100× stronger noise twin at step 1. For the 2L/$d{=}64$ and 4L/$d{=}128$ problems, the equivariant twins' final values are still negligible, with scalar-Adam values 6.6×10^{-15} and 6.0×10^{-12} and SGD values 2.5×10^{-9} and 1.6×10^{-10}. The ratio of Adam's smallest final gauge-twin value to the stronger noise twin's final value exceeds 500× in the $d{=}64$ case, agreeing with the comparison the CPU harness reports above.

In contrast, we observe high final values for scalar-Adam's and Muon's gauge twins in the 6L/$d{=}256$ case, at 1.8 and 0.58 respectively, though we do not get a measurement for SGD, as SGD fails to learn past chance accuracy. Looking at noise twins in the same setting reveals similar magnitudes of 1.7 for scalar-Adam and 0.83 for Muon. This suggests that the increase in scalar-Adam's gauge-twin values is due to the same instability that drives its drift under a 10^{-7} additive perturbation in the same basis. A similar effect can be observed with scalar-Adam on our char-LM problem, where the gauge and noise twins both have final value 0.44.

However, as before, the behavior of gauge twins at step 1 (the step-1 columns of Table 8) does not change significantly. The deterministic $A{=}I$ twin has value exactly 0 in each setting. Adam's gauge deviation at the first iteration ranges from 10^{-3} to 10^{-2}, and the other methods are at machine precision, about 12–13 orders of magnitude lower than Adam. The resulting onset ratio ranges from 39× to 158×. Only in Adam's case do its gauge and noise twins differ as of step 1. Thus, while equivariant methods' final gauge-twin values may be impacted by phenomena besides gauge deviation, the step-1 onset remains a valid test.

Table 9: GD has lower held-out error at these training-selected rates. Entries are held-out root-mean-square errors (RMSE, $\times 10^{-2}$), shown as mean $\pm$ standard deviation over four seeds. Training loss is matched at $\leq 10^{-5}$ in GPU `float64`. m/dof_{24} counts observations per degree of freedom at rank 24. Parentheses give Muon's effective rank (erank). The ordering can change under best-rate selection (Appendix D.6).

dataset	m/dof_{24}	GD	Adam	Muon (erank)
Indian Pines	1.15	1.514 ± 0.012	2.628 ± 0.045	3.389 ± 0.142 (36)
	1.9	1.262 ± 0.013	1.740 ± 0.038	3.209 ± 0.129 (35)
Pavia Univ.	1.15	0.867 ± 0.027	1.507 ± 0.115	1.371 ± 0.079 (10)
	1.9	0.631 ± 0.010	0.813 ± 0.026	0.788 ± 0.033 (8)

D.5 Hyperspectral learning-rate hygiene

Optimal learning rates in §6 are selected by a train-only rule: lowest training loss within a budget of 3×10^4 steps, ties broken by fewer steps. Two of the four evaluation seeds are also used for rate selection, and the other two are independent of it. At density 0.15, gradient descent's train-only optimal rate is 30. This matches the rate chosen with test-informed hyperparameters, but with zero data leakage. Only the optimal learning rate reaches a train loss of 10^{-5} within budget. The other tested rates stop higher, with a rate of 10 reaching 3×10^{-5}, a rate of 3 reaching 3×10^{-4} and a rate of 1 only 3×10^{-3}.

How closely the two agree, and on what clock. At the smallest learning rate tested in §6, all sixteen pairs of GD and scalar-Adam runs agree to a precision of 10^{-5} on held-out error and 0.02 on effective rank. However, GD requires only $\sim 10^4$ training steps to get there, where scalar-Adam requires 6×10^4. That difference in step counts is the kind of time change Theorem 3.4 describes. Discrepancies are much smaller with smaller step sizes, shrinking at the empirically logged, rather than predicted, log–log slope of about 1 (Table 19). For scalar-Adam this agreement is empirical (see the scope notes after Lemma A.6). We also compare the runs at checkpoints that match in loss rather than at even temporal intervals.

Muon's inversion on Indian Pines, which we do not resolve. Muon, the most aggressive of the equivariant schedules, trails Adam and GD on Indian Pines, yet beats Adam on Pavia, where it undergoes rank collapse. On Indian Pines, Muon's effective rank never collapses within our training budget, remaining in the mid-30s at both densities. The performance deficit of Muon on Indian Pines could be due to Muon's tail-axis weakness, but more likely reflects an inadequate training budget, since a rank that has not collapsed is a run that has not finished its spectral schedule. We have not run the sweep that would settle it, so we do not claim it. The conclusions of §6 use only GD and Adam, so are unaffected.

The GD–Adam gap is not an edge-of-stability artifact. We also rule out an alternative explanation for GD's advantage, the so-called edge-of-stability effect (Cohen et al., 2021), which can give high learning rates an advantage for GD. To do so, we analyze GD's trajectory with the smaller rate lr 3 across four different seeds. GD(lr=3) follows the same trajectory as GD(lr=10), as seen by comparing held-out RMSE as a function of train loss. The two achieve nearly identical held-out RMSEs, differing by at most 10^{-5} at the same train losses; for example, both read 0.03487 at one matched loss and 0.02207 at a later one. GD(lr=3) has a 12% advantage over Adam at the deepest losses it achieves on all four seeds, that is, training loss 3×10^{-4} or lower. GD's advantage is 44% for the deeper losses reached by higher learning rates. Thus, GD's matched-loss trajectory is the same (rate-invariant), at least for these learning rates and this density, and the learning rate only affects how far down this trajectory GD proceeds in a given number of optimization steps.

Limit 1: the reported improvement depends on the selection criterion. Note that Adam achieves its best held-out RMSE with its lowest learning rate of 10^{-3} in this run. At effective rank 17 (vs. GD's 14), Adam's best hyperparameter achieves a held-out RMSE of 0.0174, vs. 0.0151 for GD(lr=30). While the exact size of GD's advantage depends somewhat on hyperparameter selection (see below), GD's qualitative advantage on this grid does not. The extent of GD's advantage depends on whether we select the best-performing learning rate for GD and Adam individually, in which case GD has a 13% advantage, or choose learning rates by the best training loss using only train-set performance, giving a 42% advantage. That second criterion is somewhat GD-biased, as train-only hyperparameter

selection may overly prefer larger learning rates to reach low loss faster in a given optimization budget. Either way, the ordering does not change across four paired random seeds in our GPU-replicated experiment, and on this grid GD outperforms Adam according to held-out performance with sensible hyperparameters for each method.

We report results based on a simple selection protocol that depends only on training-time quantities. The purpose is not to identify the best possible choice of learning rate, but to avoid introducing additional bias through use of the held-out data: the held-out RMSE would be used for selecting the learning rate and then reported as the final performance. The same argument is made in Appendix D.2. Under this protocol, at every rate on Adam's grid (held-out RMSE/effective rank of $0.0174/17$, $0.0201/20$ and $0.0263/28$), the selected GD run achieves both a lower held-out error and a lower effective rank than Adam.

Best-rate selection on an extended grid. We now describe what happens when we extend Adam's grid three orders of magnitude further down and include the best-performing rates for GD and Adam. Our main conclusion remains unchanged, but the relationship between GD and Adam in terms of held-out error becomes mixed. We pick the best learning rate separately for GD and Adam, within the set of learning rates considered for each method, by minimizing held-out RMSE. The resulting differences in held-out error, with positive meaning GD ahead, are -3.1% $[-4.6, -1.7]$ on Indian Pines at m/dof_{24}=1.15, $+0.2\%$ $[-0.8, +1.1]$ on Indian Pines at m/dof_{24}=1.9, $+2.9\%$ $[-1.7, +7.6]$ on Pavia at m/dof_{24}=1.15, and $+7.6\%$ $[+5.4, +9.9]$ on Pavia at m/dof_{24}=1.9. These percentages and their 95% t-intervals are computed by first pairing seeds across methods, computing the percentage differences, and then averaging. There are 3 degrees of freedom. One cell is resolved in each direction, and in the other two we do not have sufficient resolution to detect any difference in error.

The above results suggest that the conclusion of Table 9, that GD achieves lower error, is an artifact of the learning-rate selection procedure (train-only) that we use there. For this reason, we do not claim from Table 9 that moving from Adam to GD reduces error. By contrast, the conclusion that GD achieves a lower effective rank is true for each of the learning rates tested. At the best-performing learning rate for each method, Adam achieves a higher effective rank in all four cells, with differences of $+0.15$, $+0.44$, $+0.60$ and $+0.63$ from the first Indian Pines cell through the last Pavia cell, statistically resolvable in three of them. The effective-rank result is the basis for the evidence presented in §6, since effective rank is a function of the interpolants, and its ordering does not depend on the above issues around learning-rate selection.

A comparison between the two adaptive update rules is given in Table 21. It uses a common learning rate chosen to lie in the small-step limit for each method. At that rate the comparison resolves all four problem cells, and the two rules' rank offsets from GD differ by a factor of roughly 65 to 1300.

Limit 2: scalar-Adam was undersampled on this grid. A caveat applies to scalar-Adam on this grid: its learning-rate grid was undersampled, namely $\{3\times10^{-3}, 10^{-2}, 3\times10^{-2}\}$. These learning rates are $3\times$ higher than the ones used for Adam. There is no sign of saturation for scalar-Adam, which is still improving at the smallest rate tried (0.0210 vs. 0.0362 at the selected learning rate). On an extended grid of learning rates going as small as 10^{-6}, scalar-Adam is indistinguishable from GD on the two reported statistics in all four problem cells, as shown in Table 19. A similar caveat applies to the comparison with Shampoo at different problem sizes (Appendix D.10).

D.6 Four scenes, and what holds on all of them

With only two scenes, tail size and sampling density vary together, so their effects cannot be told apart. We therefore also test Adam vs. gradient descent on two more hyperspectral images (Graña et al., 2021), Salinas and Botswana, in addition to Pavia and Indian Pines from §6. Salinas has a thinner tail than either of the first two scenes, and Botswana's tail sits between theirs (Table 10).

Making the four comparable. Their other properties are different enough that we count degrees of freedom at each scene's own target effective rank, a rule applied identically to all four. These ranks are 7, 9, 15 and 34 for Salinas, Pavia, Botswana and Indian Pines, respectively, instead of the rank-24 degrees of freedom we used in §6. We then choose sampling densities in each dataset that give the same two values in terms of degrees of freedom, m/dof=1.15 and m/dof=1.9. Fit rank is held

Table 10: Spectral-tail size does not order the held-out error gap across scenes. m/dof counts observations per degree of freedom at the target's effective rank r^*. τ is tail amplitude beyond the fit rank 48. Gap reports paired percentage differences in held-out root-mean-square error at each method's best rate. Positive values favor GD. Brackets give 95% t-intervals over four seeds (3 degrees of freedom). The last two columns subtract GD's effective rank from each method's at $\eta = 10^{-6}$.

dataset	τ	r^*	m/dof	gap (%)	scalar-Adam	Adam
Salinas	0.0073	7	1.15	$+2.7$ $[-3.4, +8.9]$	$+0.046$	$+7.641$
			1.9	-11.0 $[-14.7, -7.3]$	$+0.040$	$+3.903$
Pavia Univ.	0.0094	9	1.15	-6.5 $[-15.0, +2.0]$	$+0.023$	$+2.788$
			1.9	-7.0 $[-10.5, -3.5]$	$+0.007$	$+1.978$
Botswana	0.0280	15	1.15	-6.5 $[-10.1, -3.0]$	$+0.047$	-2.250
			1.9	-0.5 $[-1.7, +0.7]$	$+0.022$	-1.163
Indian Pines	0.0515	34	1.15	-0.4 $[-1.4, +0.5]$	$+0.010$	$+0.810$
			1.9	$+0.2$ $[-0.4, +0.8]$	$+0.004$	$+0.827$

constant at 48 on all four scenes. Tail amplitude is measured beyond the fit rank, and the ordering of the four scenes is the same beyond rank 8, 16, 24 and 48, so it does not turn on the reference rank.

Held-out error is not ordered by the spectrum. The best-rate held-out gap shows no obvious relation with the relative tail amplitudes of the four scenes: Salinas, which has the thinnest tail, actually has Adam's largest relative advantage, -11.0% at $m/\text{dof}=1.9$. Nor do target rank or sampling density seem to predict generalization performance in any consistent way. So, we make no claim, even empirically, about the ability of a scene's spectral profile to predict whether Adam's solution will give a lower held-out error, although, as discussed in §6, our sweep there varies τ while keeping all else fixed, something we cannot do on these scenes.

The limit statistics hold on all four. In contrast, all of our limit statistics hold on all of the scenes. Scalar-Adam stays within 0.05 effective ranks of gradient descent on all eight cells, while Adam is 0.8–7.6 effective ranks away, above gradient descent's effective rank on Salinas, Pavia and Indian Pines and below gradient descent's on Botswana. The fact that the rank deviation works both ways suggests that Adam is finding a different interpolant in general, not necessarily a worse interpolant. Unsigned matrix distance also separates Adam from gradient descent on all cells (Table 20).

D.7 Doubling the rank cap

One potential concern about Appendix C is that its rank cap of 48 may be binding. The cap is above the reference's intrinsic rank of 24, but Muon reaches effective rank in the mid-30s on Indian Pines, close enough to 48 that the cap could be doing some of the selecting. Could the gap have been an artifact of the rank cap? To address this question, we double the rank cap to 96 on one cell per scene (the lower density), leaving everything else unchanged. Results are summarized below. We find that (a) Adam's effective rank increases substantially while GD's rises far less, supporting the theory of implicit bias, and (b) the gap actually widens, meaning that our results in §6 were conservative.

Indian Pines. GD's RMSE only changes slightly, from 1.514×10^{-2} to 1.579×10^{-2}, but its effective rank does increase from 13.6 to 20.3. Adam's effective rank increases dramatically, from 27.7 to 58.3. The gap actually increases from $+42.4\%$ to $+56.5\%$, and GD still wins across all four seeds. This shows that the cap was holding back Adam's effective rank at rank 48.

Pavia. We observe a similar effect here. Adam's effective rank increases from 10.7 to 24.9. GD's increases from 6.3 to 8.3, but it still outperforms Adam, with the gap increasing from $+42.4\%$ to $+64.1\%$ across four seeds. GD's RMSE increases from 0.867×10^{-2} to 0.971×10^{-2}.

Muon. Muon does show signs of consuming the extra capacity when allowed to do so. However, as mentioned in Appendix D.5, Muon has an unresolved step-budget issue. On Pavia at rank 96 the train-only rule's tie-break picks lr 0.03, where it fails outright, with held-out RMSE 11.2×10^{-2} at effective rank 93.6, compared to 1.371×10^{-2} at rank 48; at lr 0.1 it reads 0.943×10^{-2} at effective rank 9.8. For reference, Muon had an effective rank in the mid-30s at rank 48 on Indian Pines. There,

Table 11: FlowAdam-p=0 recovery stays stable as training loss falls. Three sensing seeds run with an extended 1.2×10^5-step budget. Columns report recovery error when training loss reaches each threshold. Erank is effective rank at the final threshold.

train loss $\leq$	10^{-4}	10^{-5}	3×10^{-6}	10^{-6}	10^{-7}	erank
seed 42	0.1404	0.1401	0.1401	0.1401	0.1401	4.77
seed 123	0.1558	0.1556	0.1555	0.1555	0.1555	4.84
seed 456	0.2120	0.2117	0.2117	0.2117	0.2117	5.41
mean	0.1694	0.1691	0.1691	0.1691	0.1691	5.01

its RMSE goes from 3.389×10^{-2} at rank 48 to 3.068×10^{-2} at rank 96, and its effective rank from 35.9 to 45.0. Therefore, for the claims in §6, it's still best to focus on the GD-to-Adam gap.

D.8 Interpolation-bar audits

Without requiring methods to achieve interpolation (§2), it's possible to score RMSProp on a run that never fits the data. For example, if trained with constant LR, RMSProp fails to interpolate, achieving a minimum training loss of 2.9×10^{-4}. With a larger compute budget ($3\times$), it still fails to interpolate, achieving a minimum training loss of 2.8×10^{-4} after 6×10^4 training steps. The cause is the ε-bounded denominator of its update rule, a stiff obstacle that doesn't yield over time. Judged at its constant-LR training-loss plateau (it never interpolates), RMSProp's recovery score is 0.5270. With decay, RMSProp does achieve interpolation and falls within the coordinate-wise cluster, achieving a score 0.0004 lower than 0.5270 (Table 4). Thus the choice of LR schedule can influence whether RMSProp's final position is measurable, but doesn't influence where it lands.

Signum. The signum method fares worse. Judged at its constant-LR training-loss plateau (training loss 5×10^{-2}, where it stalls without interpolating), its score is 0.009, falsely implying that sign descent might retain bias. With decay, it does achieve interpolation, and its score of 0.445 corroborates bias loss.

Our protocol (§2) requires every method, adaptive or not, to interpolate, because scoring a method that has not fit the data against eight that have is unfair. Luckily, with decay, all nine achieve the 10^{-7} bar, giving $9/9$ (Table 2). This also holds when every method follows the same cosine schedule (Table 4), so we see that clustering doesn't depend on any method-specific LR-scheduling quirks.

D.9 FlowAdam-p is not an artifact of early stopping

FlowAdam-p is FlowAdam (Singh & Sheel, 2026) with the preconditioner dial of Algorithm 1 added (Appendix E.22). To control for any effect of where training stops on the recovery of FlowAdam-p=0, we reproduce the experiment in Appendix E.22 using FlowAdam-p=0 with the RMS convention, global norm clipping with c=10 and a learning rate of 10^{-3}. We train longer, though, with a budget of 1.2×10^5 steps, and examine recovery at matched training losses between 10^{-4} and the interpolation threshold of 10^{-7}. The results of this experiment are shown in Table 11. Recovery values are stable in this regime, with differences in recovery between the 10^{-4} and 10^{-7} loss values being at most 3×10^{-4} for a given run. Therefore, the bias does occur before termination and is not an artifact of our choice of early stopping.

D.10 The H100 replication ladder

To confirm that the deterministic separations we find at small n are not artifacts of learning-rate selection and generalize at scale, we perform a larger experiment on H100 GPUs. We consider sensing at $n \in \{64, 96, 128, 192, 256\}$, planted rank 3, $m = 2\,\text{dof}$, and use `float64` arithmetic. For each setting, we use 10 random seeds and train for 4×10^4 steps. Following the best-recovery rule tested in Appendix D.3, three seeds are used to select the learning rate (the one that leads to the best average recovery among interpolating rates). We then rerun the experiment at this learning rate for the remaining seven "holdout" random seeds, which are also reported on their own in Table 14.

We define E and C to be, respectively, the worst mean recovery in the first (equivariant) and the best mean recovery in the second (coordinate-wise) of these two sets of optimization algorithms:

Table 12: GD, scalar-Adam, and Muon beat Adam, RMSProp, Lion, and Adafactor at every tested size. Entries are mean recovery errors for rank-3 sensing with a 4×10^4-step budget on H100 `float64`. Means include all ten seeds except RMSProp at n=96, where nine interpolate. sc-Adam means scalar-Adam. Step-size dependence: Remark A.8.

	equivariant				coordinate-wise				
n	GD	sc-Adam	Muon	Shampoo[†]	Adam	RMSProp	Lion	Adafactor	signum[‡]
64	0.143	0.195	**0.000**	0.646	0.574	0.596	0.483	0.612	0.000
96	0.202	0.279	**0.000**	0.595	0.609	0.665[§]	0.548	0.687	0.000
128	0.248	0.345	**0.000**	0.659	0.643	0.617	0.602	0.745	0.000
192	0.346	0.468	**0.000**	0.838	0.696	0.669	0.760	0.817	0.000
256	0.410	0.548	**0.000**	0.918	0.741	0.708	0.895	0.867	0.000

[†]Shampoo uses fixed damping λ=1. Damping was not tuned, so these results do not establish an intrinsic limitation. [‡]Signum recovers near-exactly with longer annealing (Appendix D.11). [§]One RMSProp seed at n=96 fails to interpolate and is excluded from the mean.

Table 13: RMSProp at n=96 has the largest variation across seeds. Entries are sample standard deviations of recovery error at the rates selected in Table 12. All ten seeds contribute except RMSProp at n=96, where nine interpolate. Zeros are rounded, not exact.

	equivariant				coordinate-wise				
n	GD	sc-Adam	Muon	Shampoo	Adam	RMSProp	Lion	Adafactor	signum
64	0.024	0.029	0.000	0.065	0.032	0.033	0.047	0.029	0.000
96	0.044	0.045	0.000	0.037	0.039	**0.297**	0.042	0.034	0.000
128	0.041	0.044	0.000	0.031	0.027	0.029	0.028	0.016	0.000
192	0.029	0.027	0.000	0.022	0.020	0.022	0.020	0.009	0.000
256	0.026	0.027	0.000	0.010	0.020	0.022	0.026	0.011	0.000

- equivariant: GD, scalar-Adam, Muon;
- coordinate-wise: Adam, RMSProp, Lion, Adafactor.

In each case, E is strictly less than C, confirming that the observed separation between these two sets of optimization algorithms holds even among holdout seeds and at scale. Furthermore, E and C computed using all 10 random seeds instead of just the holdout seeds agree with the holdout values to within 0.01 in all cases.

Three separation strengths, which are not the same claim. Table 12 shows the means separating at every size. Here we also report two stronger forms of the separation between the groups, equivariant GD, scalar-Adam and Muon vs. coordinate-wise Adam, RMSProp, Lion and Adafactor. For each random seed, we compare the worst-performing equivariant optimizer to the best-performing coordinate-wise optimizer in terms of recovery. The worst-performing equivariant outperforms the best-performing coordinate-wise for $10/10$ seeds at all sizes except n=96, for which this holds for $8/10$ seeds. If we compare runs instead, then there is separation except at n=96, where the worst recovery for an equivariant optimizer is 0.355 but one coordinate-wise run has recovery of 0.144. Both exceptions are due to instability of RMSProp with its chosen learning rate.

How failed runs are counted. One seed of RMSProp at n=96 fails to converge, having a final training loss of 6.5. Since this checkpoint does not interpolate the dataset, its recovery is not meaningful. Including it would lower RMSProp's mean error from 0.665 to 0.615 and flatter it, so Table 12 reports the average over the other 9, which do interpolate. These range from recoveries of 0.14 to 0.90, and their average is indeed 0.665. All other results in Table 12 interpolate $10/10$, with no clipped or replaced results.

Learning-rate grid scaling. The scaling of the learning-rate grids as the problem size changes is chosen empirically, such that all methods stay within the interpolating regime. The grids for Muon, Shampoo and ScaledGD are scaled by $\sqrt{40/n}$, with ScaledGD not reported but included as a schedule control. The grids for Adam-family methods are scaled by $\sqrt{n/40}$, and the grid for GD by $40/n$. In particular, all grids agree with those used in the main text for n=40. Note that these scalings

Table 14: Dropping the three selection seeds leaves the ladder unchanged. Entries are mean recovery errors over the seven seeds held out of learning-rate selection, at the rates selected in Table 12. All seven interpolate everywhere except RMSProp at n=96, where six do.

	equivariant				coordinate-wise				
n	GD	sc-Adam	Muon	Shampoo[†]	Adam	RMSProp	Lion	Adafactor	signum[‡]
64	0.145	0.196	**0.000**	0.664	0.583	0.600	0.490	0.616	0.000
96	0.213	0.288	**0.000**	0.605	0.613	0.730[§]	0.553	0.689	0.000
128	0.238	0.335	**0.000**	0.655	0.640	0.614	0.594	0.740	0.000
192	0.350	0.472	**0.000**	0.842	0.699	0.672	0.761	0.818	0.000
256	0.419	0.558	**0.000**	0.917	0.748	0.715	0.899	0.872	0.000

[†]Shampoo uses fixed damping λ=1, which was not tuned. [‡]Signum recovers near-exactly with longer annealing (Appendix D.11). [§]The holdout standard deviation here is 0.268, the same RMSProp instability Table 13 records.

are not claimed to follow from any theoretical scaling law, and were chosen so that the resulting learning-rate grids covered the range of empirically best learning rates.

Intuitively, the target's Frobenius norm increases as n increases, while the starting point is always initialized at scale 10^{-3}. With a fixed number of steps, methods must travel a larger Frobenius distance as n grows. For gradient descent, this change in scale is accounted for in the gradient, so learning rates can scale down. In contrast, the magnitudes of the per-coordinate updates for Adam-family methods are independent of the scale of the gradient, so learning rates can scale up. For those methods, the magnitude of a single per-coordinate update is approximately η.

Schedule control. To control for schedule, we repeat the benchmarking procedure with cosine-annealed schedules, against each method's baseline schedule. Methods with constant update norm, such as Muon, signum and Lion, need some decay in the learning rate in order to interpolate, so their baseline already decays (Appendix D.1). We find that, except for the unstable behavior of RMSProp at n=64 and n=96, the reported recoveries agree to three decimal places between the two schedules. For all algorithms except RMSProp, the reported recovery is the average over all seeds. At n=64 and n=96, RMSProp is unstable for some learning rates. Reading both schedules on the mean over all 10 seeds, its recovery error rises from 0.596 to 0.610 at n=64 and from 0.615 to 0.767 at n=96 under the cosine schedule. The reported 0.665 at n=96 is the average of the seeds that interpolate. Regardless of schedule, RMSProp is still clearly part of the cluster of coordinate-wise methods, and the ladder of methods remains unchanged. See Table 12 for details.

Why not include runs at n=384? For Muon it is not possible to choose a learning rate which achieves interpolation at the common threshold of 10^{-7} for any of the rates in the sweep. The loss plateaus at about 10^{-5}, and the best-performing rate is at the edge of our sweep. Two other methods would have rates on the edge of our sweep as well. Including them would give a picture influenced by these issues, and fixing them requires a wider sweep and a rerun that we have not done. We choose to leave them out. This reflects the limits of our sweeps, and is not evidence about n=384.

The ladder read on the holdout seeds alone. Table 14 is identical to Table 12, except that it is measured using the 7 holdout seeds not included in learning-rate selection. Thus, none of its entries are influenced by the particular hyperparameter selection made. The entries are nearly identical, no more than 0.018 away, except in the n=96 RMSProp entry. We do not feature Table 14 because, unlike every other table in this paper, it does not use all seeds. Table 12 is our headline result, but we include Table 14 for reference.

A caveat on the ladder. The ratio of the Adam-family method error to GD error decreases with increasing n. This is not a matter of Adam-family methods doing better in a relative sense. Rather, the absolute errors for GD are increasing, a finite-budget phenomenon, while Adam-family methods are performing as well as they're going to perform. This phenomenon is not an approach to the continuous-time, or flow, limit, since increasing n with a fixed budget is not a small-step limit. Given more, smaller steps, both GD and scalar-Adam should approach their flow limits. Table 12 is designed to show the relative merits of the methods, not necessarily their absolute errors.

Two exceptions. Exceptions are Shampoo with untuned damping and signum, which recovers near-exactly over the ladder's longer horizon. They do not adversely affect the separation of GD, scalar-Adam and Muon from Adam, RMSProp, Lion and Adafactor, but do obstruct a complete separation of all nine methods. Accordingly, we make this stronger claim only for n=40 in §4, and flag Shampoo and signum for further discussion in §7.

The n=40 canary. The canary, the ladder's H100 pipeline rerun at the zoo's size, recovers the numbers of Table 2 for Lion (0.424 here and 0.425 there) and Adafactor (0.543 in both), while Adam stays flat under the extended grid and obtains 0.550 at its finest interpolating rate. In contrast, as the step sizes are lowered, GD and scalar-Adam improve to 0.108 and 0.143, respectively, on their finest grids. However, in the case of GD, training stalls at an average training loss of 1.1×10^{-7}, just over our threshold of 10^{-7} for interpolating runs, so we use GD's 0.108 only as a step-size diagnostic rather than a definite instance of interpolation. Overall, our observations are consistent with the flow-limit behavior described in Remark A.8, and suggest that the other (larger-n) points on the ladder may be conservative.

D.11 Recovery without equivariance: annealed sign descent at long horizons

The zoo's two-cluster separation (§4) is defined under a fixed computational budget. One coordinate-wise method escapes the collapsed cluster when training is extended: signum (signSGD with momentum) using cosine decay over a prolonged annealing schedule.

At the paper's standard budget of 2×10^4 steps, signum's recovery error is 0.445, classified as collapsed. When extended to the ladder's 4×10^4-step horizon, recovery drops to near-zero (below 10^{-5}) for all $n \geq 64$ (across all 10 random seeds). The transition begins at n=40, a borderline scenario depending on the platform and the random seed. On H100 in `float64`, two of the three seeds recover, for a mean error of 0.013. The original CPU-based implementation (`optimizer_zoo_bias.py`, `MAX_STEPS`=4×10^4, lr 3×10^{-3}) gives a mean of 0.054, with the run using seed 42 recovering to 1.5×10^{-5}. At lr 10^{-3}, recovery stays poor at about 0.71 throughout.

The values shift between individual runs and platforms, but the configuration as a whole (large step sizes and an extended annealing schedule) reproduces the effect. It is not random, and it is not due to numerical instability. We observe the phenomenon but lack a mechanistic account, so we report it as an empirical fact.

The example does not disprove the theorem. It shows where the theorem does not apply. Theorem 3.4 concerns flows with memoryless updates and a scalar preconditioner shared across the factors, and does not consider alternatives. A different set of update rules, not respecting the equivariance of the loss under the gauge, could have different dynamics and still exploit the low-rank structure of the target signal. We considered one possibility and ruled it out.

Ruling out the rank-one hypothesis. One candidate explanation is that signum's momentum concentrates into a rank-one structure. If the momentum were an outer product, then since $\mathrm{sign}(\sigma uv^\top) = \mathrm{sign}(u)\mathrm{sign}(v)^\top$ for $\sigma > 0$, the signed update would itself be rank-one and aligned with the planted signal. The data rule this out.

At 4×10^4 steps, the successful runs at lr 3×10^{-3} show momentum singular-value ratios $\sigma_1/\sigma_2 \approx 1.1$–$1.3$ at convergence. The signed updates themselves have even lower values (1.05–1.17). The leading singular component carries 10% to 24% of the squared Frobenius norm (`experiments/signum_c9_audit.py`), against the 100% a genuinely rank-one matrix would carry. The update is not rank-one. How signum achieves low-rank recovery after long annealing remains open.

A more promising direction: ℓ_∞ steepest descent. Sign-based updates are ℓ_∞ steepest-descent steps (Bernstein et al., 2018; Xie & Li, 2024; Xie et al., 2025), which implicitly bias towards max-margin solutions. Baek et al. (2026) prove that signum converges to the ℓ_∞-max-margin classifier at any batch size on separable data, provided the momentum parameter β is sufficiently close to 1. If such solutions were low rank only when the ground truth is, that would explain why this success disappears once a spectral tail appears (as illustrated below).

Table 15: Muon gives way to GD as the target's spectral tail grows. Entries are mean recovery errors over ten sensing seeds in `float64`. τ is tail amplitude, and τ^2 is its fraction of target energy. Bold marks the best error and ties. A tie means the means differ by less than the larger seed standard deviation. Figure 3 (left) shows these spreads.

τ	GD	Adam	Muon	Shampoo	regime
0.00	0.112	0.542	**0.000**	0.334	Muon near-exact
0.05	0.150	0.543	**0.095**	0.343	Muon
0.10	0.214	0.555	**0.191**	0.375	Muon
0.20	**0.351**	0.597	**0.354**	0.449	boundary ($\approx 4\%$ tail); Muon cedes
0.35	**0.549**	0.685	0.580	0.576	GD
0.50	**0.727**	0.776	0.752	**0.723**	tail regime; GD/Shampoo within noise

We flag this as speculation. No theorem connects the cited ℓ_∞-max-margin classification results to recovery under a squared loss on matrix sensing, and we do not supply one. We posit it as an open problem.

This does not look like a continuous limit. This example differs from the common-scalar case in two ways.

First, it emerges from large learning rates and long annealing schedules rather than from a smooth flow limit. At a learning rate of 3×10^{-3}, the average recovery over three random seeds improves from $0.445 \to 0.054 \to 0.000$ as the cosine-annealing schedule is extended from $2 \times 10^4 \to 4 \times 10^4 \to 6 \times 10^4$ steps. Extending the schedule stretches the decay period to match, so the rate follows $\eta \cdot \frac{1}{2}(1+\cos(\pi t/T))$ with T the full horizon in each case. It does not decay over the original 2×10^4 steps and then hold flat. At a learning rate of 10^{-3}, recovery remains poor (0.65–0.72) throughout the schedule. A linear decay schedule has similar threshold behavior ($0.479 \to 0.103 \to 0.000$ at analogous durations), suggesting this is not specific to cosine annealing.

Second, this success breaks down when a spectral tail is added to the planted matrix. In the tail sweep described in §6, signum at lr 3×10^{-3} and 4×10^4 steps shows recovery values of 0.025, 0.163, 0.290, 0.408, 0.629, and 0.787 at $\tau = 0$, 0.05, 0.10, 0.20, 0.35, and 0.50 respectively (averaged over three seeds). The τ=0 value again reflects the seed-sensitive borderline case, and sign descent is chaotic enough that individual digits shift by up to 0.03 across platforms. The monotone breakdown itself does not. Common-scalar methods follow the smoother degradation patterns seen in Table 15.

We view signum as an exceptional case that helps define boundaries. In the configurations we tested, recovery succeeds in a restricted regime: an exactly low-rank target, large steps, and a long anneal. We tested one non-equivariant method on one grid, so this is not evidence about non-equivariant dynamics in general. What it does show is that the common-scalar route is not the only one, while remaining the one that is robust here and open to analysis.

E Additional results and derivations

This appendix holds supporting material for the body: experiment readings, controls, derivations, and the floats that the text refers to. Appendix A collects the numbered results.

E.1 The spectral-tail sweep: design and table

The sweep asks how recovery changes as the target gains a spectral tail. The target is a mixture of a rank-3 signal and a dense residual: plant $X = \sqrt{1-\tau^2}\,X_3 + \tau E_\perp$, where X_3 is the rank-3 matrix and $E_\perp$ is a dense rank-$(n-3)$ component supported on the orthogonal complements of X_3's column and row spaces. Both components are normalized to $\|X_3\|_F = \|E_\perp\|_F = 1$. The support condition gives $\langle X_3, E_\perp \rangle_F = 0$ and hence $\|X\|_F = 1$, so $\tau \in [0,1]$ and τ^2 is exactly the fraction of target energy in the tail.

Sensing uses $m = 2\,\mathrm{dof}$ counted at the rank-3 target, as elsewhere. All methods interpolate. The sweep is a separate 10-seed run on its own two-rate grid: $\{10^{-2}, 3\times10^{-2}\}$ for GD, $\{3\times10^{-3}, 10^{-2}\}$ for Adam, $\{10^{-3}, 3\times10^{-3}\}$ for scalar-Adam, and $\{3\times10^{-2}, 10^{-1}\}$ for Muon and Shampoo.

That grid is held fixed across every τ, so the candidate set cannot widen as the tail grows. Within each (τ, method) cell, the rate giving the best mean recovery among those reaching the 10^{-7} bar inside 2×10^4 steps is selected. This is the same selection rule §4 uses. Because the chosen rate can move with τ, the crossover τ^* is a crossover between per-τ best-rate envelopes rather than between fixed-rate trajectories.

Seeds are $\{42, 123, 456, 789, 1011, 1213, 1415, 1617, 1819, 2021\}$ and τ runs over $\{0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.325, 0.35, 0.375, 0.4, 0.45, 0.5, 0.6\}$. Because this grid is coarser than the zoo's, its τ=0 column reproduces Table 2's *ordering* rather than its exact digits (for example, GD reads 0.112 here against 0.131 there). Table 15 tabulates the results, and Figure 3 (left) plots them. At the largest τ, GD and Shampoo fall within seed noise, so the ordering there is uninformative. Table 15 and Figure 3 (left) show six of the fourteen τ values; at τ=0.6, not shown, Muon (0.846) sits above Adam (0.837), within seed noise.

E.2 A TWO-DIMENSIONAL WITNESS FOR PROPOSITION 3.1

The failure of a coordinate-wise rule shows up in the represented matrix at the first step. A k=2 example makes the point without any appeal to the general proof. With $n=1$, $k=2$, $f(w)=\frac{1}{2}w^2$, $U_0=V_0=(1,0)$, and $Q=2^{-1/2}\left(\begin{smallmatrix}1 & 1\\ -1 & 1\end{smallmatrix}\right)$, the two resulting products after one step are $(1-\eta/(1+\varepsilon))^2$ and $(1-\eta/(2^{-1/2}+\varepsilon))^2$, which differ whenever $0<\eta<2^{-1/2}+\varepsilon$.

E.3 READING THE ZOO FIGURE, AND TWO FURTHER READINGS FROM THE TABLE

Figure 1 and three rows of Table 2 carry more than their recovery numbers, and this subsection reads them.

The markers. Each bar in Figure 1 shows the mean of the three seeds, and we plot the value for each individual seed as a black dot. A thin dark line joins the seeds of each row. A seed above the mean lies past the end of its bar, and the line keeps it attached to that bar. The shaded region shows the range of values from the worst equivariant run (0.313) to the best coordinate-wise run (0.337): no individual runs fall within this range. Muon is plotted using scientific notation since its value is 0.000 when rounded to 3 decimal places. The budget is 2×10^4 steps. Beyond this horizon, signum eventually catches up (Appendix D.11).

Adafactor. Adafactor loses the bias since its factored diagonal still computes the surrogate in a coordinate-wise fashion, suggesting that gauge-dependence, and not just compression, is the issue. Adafactor is not directly covered by Proposition 3.1, since it doesn't compute a fixed entrywise map, so Appendix B gives it its own counterexample, which we include for completeness.

Signum. Signum has the lowest effective rank among the coordinate-wise optimizers and has effective rank just above Shampoo's, which is equivariant, but has final recovery more like the other coordinate-wise optimizers. That is why we mark signum as "low-rank-but-wrong" in Table 2: a low effective rank does not necessarily mean the right solution. Recovery, not effective rank, is therefore the metric we sort on.

RMSProp. RMSProp's insensitivity to the schedule has a cause unrelated to the gauge, its ε-bounded denominator. Because of it, RMSProp cannot reach the interpolation threshold with a fixed step size, and it lands on the same value either way (Appendix D.8).

E.4 THE NORM FORM OF THE CRITERION, STATED IN FULL

Corollary 3.3 (norm form of the criterion), stated in full. Let $\|\cdot\|$ be a norm on $\mathbb{R}^{n\times k}$ and let $\mathscr{D}(G)=\arg\max_{\|\Delta\|\le1}\langle G,\Delta\rangle$ be the corresponding steepest-descent correspondence. *(i) Memoryless.* If $\|\cdot\|$ is right-invariant, meaning $\|MQ\|=\|M\|$ for all $Q\in\mathrm{O}(k)$, then $\mathscr{D}(GQ)=\mathscr{D}(G)\,Q$. A single-valued selection $\Delta(G)\in\mathscr{D}(G)$ is therefore gauge-equivariant when the maximizer is unique, and otherwise under an equivariant selection such as Muon's support-restricted polar factor. A right-invariant norm is exactly one that sees M only through the Gram matrix $MM^\top$. *(ii) Stateful.* If the

norm carries optimizer state, $\|\cdot\|_s$, and that state is gauge-covariant in the sense of Definition 2.1, so that $\|\Delta Q\|_{\sigma_Q(s)} = \|\Delta\|_s$, the same conclusion holds.

Proof. The whole argument is a change of variables on the unit ball.

(i) Right-invariance says the ball $B = \{\Delta : \|\Delta\| \le 1\}$ is unchanged by right multiplication, $BQ = B$ for every $Q \in \mathrm{O}(k)$. So $\Delta \mapsto \Delta Q^\top$ maps B onto itself. Substituting $\Delta = \Delta' Q$ in the maximization and using $\langle GQ, \Delta' Q\rangle = \mathrm{tr}(Q^\top G^\top \Delta' Q) = \mathrm{tr}(G^\top \Delta') = \langle G, \Delta'\rangle$ gives

$$\mathscr{D}(GQ) \;=\; \arg\max_{\Delta\in B} \langle GQ, \Delta\rangle \;=\; \Big(\arg\max_{\Delta'\in B} \langle G, \Delta'\rangle\Big) Q \;=\; \mathscr{D}(G)\, Q.$$

A single-valued selection therefore satisfies $\Delta(GQ) = \Delta(G)Q$ whenever the maximizer is unique, and under any selection rule that commutes with the gauge otherwise. For the Gram statement, $\|\cdot\|$ is right-invariant exactly when it is constant on right-$\mathrm{O}(k)$ orbits, and the argument in the proof of Theorem 3.2 shows that two matrices of the same shape lie on one such orbit precisely when their Grams agree. A right-invariant norm therefore factors through $MM^\top$, and conversely any function of $MM^\top$ is right-invariant. The converse of the corollary fails: $\Phi(G) = -G$ is equivariant, with $H = -I$, but is never a steepest-descent direction, since $\langle G, -G\rangle < 0$.

(ii) The state-dependent hypothesis $\|\Delta Q\|_{\sigma_Q(s)} = \|\Delta\|_s$ says the ball at the transformed state is the transformed ball, $B_{\sigma_Q(s)} = B_s Q$. The same substitution then gives $\mathscr{D}_{\sigma_Q(s)}(GQ) = \mathscr{D}_s(G)\, Q$, which is Definition 2.1 for the pair (update, state). □

E.5 Shampoo and part (ii) of the norm form

The norm that generates Shampoo's step is $N_t(\Delta) = \left\| L_t^{1/8} \Delta R_t^{1/8} \right\|_F$, not the quarter powers the update itself carries. To see why: for $N(\Delta) = \|A\Delta B\|_F$ with A, B symmetric positive definite, the change of variables $Y = A\Delta B$ makes the steepest-descent direction proportional to $A^{-2} G B^{-2}$. The eighth powers are therefore what return Shampoo's $L_t^{-1/4} G R_t^{-1/4}$. Damping replaces L_t, R_t by $L_t + \lambda I, R_t + \lambda I$ throughout.

That norm is *not* right-invariant in Δ when the metric factors are held fixed, since Q does not commute past $R_t^{1/8}$. What rescues it is that the metric itself transforms with the gauge. Under the gauge, L_t is invariant while $R_t \mapsto Q^\top R_t Q$. The chain of equalities then reads: $\left\| L_t^{1/8} (\Delta Q)(Q^\top R_t Q)^{1/8} \right\|_F = \left\| L_t^{1/8} \Delta R_t^{1/8} Q \right\|_F = \left\| L_t^{1/8} \Delta R_t^{1/8} \right\|_F$. Shampoo is therefore an instance of part (ii), not part (i), and only in its idealized form.

The exponent does not enter this argument. What matters is that L_t is invariant and R_t conjugates. Deployed variants that graft an entrywise Adam step size onto the direction fall outside part (ii). A cached full-matrix preconditioner does not, provided both runs share the refresh schedule and the cache is transformed with the gauge.

E.6 Three qualifications on the norm correspondence

Corollary 3.3 matches the criterion to steepest descent under a norm, and three qualifications limit how far that match goes.

1. The condition is right-invariance, not the more familiar unitary-invariance of a symmetric gauge function of the singular values. Unitary-invariance is sufficient but a stronger assumption. For example, the weighted Frobenius norm $\|M\|_D = \|DM\|_F$ with fixed, non-scalar D is right-invariant without being unitarily invariant.
2. The characterizations cited in §3 refer to the unfactored parameter θ and the direction of the asymptotic iterates during separable classification. By contrast, Proposition A.2 is about the exact algebraic identity for one step. The same coordinate geometry appears at the start and at infinity; we do not read an implication one way or the other.

3. The characterization of Muon as msign holds exactly for the map from polar decomposition, and only approximately for Newton–Schulz with a finite number of iterations (see Proposition A.1(3)).

E.7 One instance, one initialization, four latent bases

The twin experiments elsewhere in the paper compare an $\mathrm{O}(k)$ rotation against a 10^{-7} perturbation, so moving outside the symmetry and moving far happen together there. A reader is entitled to ask whether Adam's apparent sensitivity comes from moving *outside the symmetry*, or merely from making a *large* parameter change. The following experiment separates those explanations. On the sensing task of §2, with fixed data and fixed initialization, it runs Adam, gradient descent and scalar-Adam from (U_0, V_0) in four kinds of arm: identity, signed-permutation $Q = PS$, dense orthogonal $Q \in \mathrm{O}(k)$, and noise. Signed-permutation Q has one entry of modulus 1 per row and column; the dense arm uses four independent Haar-random rotations per instance. The perturbation norm is the Frobenius norm of the parameter-space displacement, $\|U_0 Q - U_0\|_F / \|U_0\|_F$, not a function-space error. All cases ran on CPU with `float64`. The resulting relative perturbation is almost identical for signed-permutation and dense orthogonal transformations, being 1.41 and 1.42, but very different from the added-noise case, where the relative size of the added-noise perturbation is 10^{-7}.

Adam is equivariant under signed permutations, but not under general rotations: momentum permutes and flips sign; squaring erases sign flips, so second moments permute unsigned; the denominator thus just permutes; and the update itself permutes and flips sign. Moreover, the ℓ_∞ ball is signed-permutation-invariant but not dense-rotation-invariant, which is all Corollary 3.3 requires. So the criterion predicts which basis changes Adam will be blind to, not just that there are some it will not be blind to, and that prediction is fixed before the runs.

The implementation was checked first. Numerical covariance requires Definition 2.1's correspondence of parameters, not just of their product $UV^\top$. Over the first 40 steps, Adam's maximum relative parameter error, $\left\| U_t^{(Q)} Q^\top - U_t \right\|_F / \|U_t\|_F$, was 3.4×10^{-16} under $Q = PS$, and 0.91 under dense Q's. Gradient descent and scalar-Adam had worst-case relative error $< 10^{-15}$ for both kinds of Q. This protocol is implemented in `experiments/equivariance_balance_probe.py`.

What Table 16 says. Table 16 makes two points. First, Adam chooses different interpolants in the two gauge arms. This can be seen from d_W, the relative distance between the optimized matrices. Although the arms' parameter changes are nearly identical in magnitude (1.41 vs. 1.42), the gauge transformations' effects are wildly different. One preserves W up to floating-point precision, since Adam is equivariant under signed permutations, while the other changes the matrix by 0.332, which is one third its size. The arms differ as soon as Adam takes its first update and remain different throughout optimization. The dense arms' d_W reads 0.94 at step 1, 0.48 at step 10 and 0.33 at step 100, while the signed-permutation arm stays at most 3.3×10^{-16} at each of those steps. Proposition A.2 is the step-1 entry of that row.

What it does not say. The second point is what Table 16 does not say. Despite dramatically different W's, both arms satisfy the interpolation criteria identically. The discrepancy in W via the measurement operator is minuscule, only 4×10^{-6} of the measured signal on average and at most 1.6×10^{-5} for a given run. We see vastly different interpolants, but they're difficult to distinguish from training data alone. The difference in recovery is more pronounced but not dramatic either. On this task Adam's recovery does move from gauge to gauge, but we do not offer that as an explanation of the gap in mean recovery seen in Figure 1. Within a given instance of four dense gauges, recovery varies with a standard deviation of 0.017, while between instances it varies with a standard deviation of 0.047. The best and worst gauges' recovery can be 0.040 apart. On average, the paired difference in recovery from the identity basis is -0.008 ± 0.017 over the twelve instances, which does not suggest a systematic difference in recovery ($t = -1.6$ on 11 degrees of freedom).

We emphasize that averaging over dense gauges within an instance is not like averaging over instances. Although the task itself is rotationally invariant, the optimizer's behavior is not: Adam's update rule depends on a particular basis. Since initialization is isotropic Gaussian, it is invariant in distribution

Table 16: Dense rotations change Adam's fitted matrix, while signed permutations do not. Twelve sensing instances use four dense rotations each. All runs reach the interpolation threshold. Rates are selected on identity runs and then fixed. "in move" is relative parameter displacement at initialization. $\mathrm{d}_W = \left\|W^{(Q)} - W^{(I)}\right\|_F / \left\|W^{(I)}\right\|_F$ measures fitted-matrix distance. $\mathrm{d}_{\mathscr{A}} = \|\mathscr{A}(\Delta W)\|_2 / \left\|\mathscr{A}(W^{(I)})\right\|_2$ measures distance in observed measurements. Dense rows average all 48 runs, with minimum and maximum distances in brackets. Lower recovery is better. Its standard deviation is across twelve instances, using each instance's mean over rotations.

method	arm	in move	d_W	$\mathrm{d}_{\mathscr{A}}$	recovery
Adam	identity	0	0 exactly	0 exactly	0.5444 ± 0.0468
	signed perm.	1.41	5.0×10^{-16}	2.6×10^{-16}	0.5444 ± 0.0468
	dense rotation	1.42	**0.332** [0.276, 0.402]	4.1×10^{-6}	0.5366 ± 0.0540
	noise (10^{-7})	10^{-7}	3.8×10^{-7}	7.0×10^{-12}	0.5444 ± 0.0468
GD	identity	0	0 exactly	0 exactly	0.0960 ± 0.0289
	signed perm.	1.41	5.6×10^{-16}	2.6×10^{-16}	0.0960 ± 0.0289
	dense rotation	1.42	8.3×10^{-16}	2.8×10^{-16}	0.0960 ± 0.0289
	noise (10^{-7})	10^{-7}	3.4×10^{-9}	1.8×10^{-12}	0.0960 ± 0.0289
scalar-Adam	identity	0	0 exactly	0 exactly	0.1770 ± 0.0404
	signed perm.	1.41	4.0×10^{-16}	2.8×10^{-16}	0.1770 ± 0.0404
	dense rotation	1.42	7.1×10^{-16}	3.1×10^{-16}	0.1770 ± 0.0404
	noise (10^{-7})	10^{-7}	6.9×10^{-9}	1.4×10^{-12}	0.1770 ± 0.0404

to rotation; we cannot say there is a basis which performs best on average over initializations. All four dense gauges are predetermined, their seeds are fixed, and all of them are reported.

A few details about Table 16. Adam's dense arms were optimized for an average of 500 steps and the identity arm 483; dense arms converged to a training loss of 6.9×10^{-9}, identity arm 1.1×10^{-8}, so neither arm got there by stopping early. Gradient descent and scalar-Adam are invariant to the gauge up to 10^{-15} and their recovery agrees to 4 decimals for all arms. Their respective "noise" arms, which add a perturbation of 10^{-7}, move d_W by about 10^{-9}, which measures how far this task carries a 10^{-7} perturbation through the run. These results should be treated as a baseline for the scale of variation seen in the other arms.

E.8 Merge experiment details

This subsection gives the protocol behind Table 17 and the end-to-end merges of §5.

Merge metric definition. Table 17 reports the held-out loss of the merged model, averaged in the product space, minus the average of the held-out loss of the two adapted models, in units of nats. We also report the relative distance between the twins' gauge-invariants $\Delta W = BA$, both at the first LoRA update, since LoRA's invariant distance is 0 at step 0 and the first update is the first that could exhibit a difference between twins, and at the end of fine-tuning. Negative values of the merge metric indicate that merging improves performance. The distance is the symmetrized relative Frobenius distance

$$d(\Delta W_1, \Delta W_2) \;=\; \frac{\|\Delta W_1 - \Delta W_2\|_F}{\frac{1}{2}\big(\|\Delta W_1\|_F + \|\Delta W_2\|_F\big)},$$

taken per adapted matrix and averaged over them, with $d := 0$ when both norms vanish. The noise twin is the same optimizer with A perturbed by 10^{-7} or 10^{-5} and no basis change. Noise-twin rows are single-seed, while the gauge and optimizer rows use three draws. All runs are rank 8 in GPU `float64`.

Details on frozen-backbone results in Table 17. We find that gauge-differences in models with frozen backbones merge without cost, at both final invariant distances in the table, on both tasks we test. We also ran an ablation where both twins had different initializations, finding that differences arising from gauge transformations merged just as well as, or better than, differences arising from the usual randomness of neural-net training.

Table 17: LoRA gauge twins. Relative distance between adapter products $\Delta W = BA$ at step 1 and after fine-tuning; merge cost in nats (negative is better). †: Muon's modular-addition adapters underperform the frozen backbone, so these cells are not interpreted as merge results.

	modular addition			character LM		
optimizer	step 1	final	merge	step 1	final	merge
Adam	$\mathbf{7.9 \times 10^{-1}}$	9.7×10^{-1}	-0.15	$\mathbf{7.9 \times 10^{-1}}$	2.8×10^{-1}	$< 10^{-5}$
SGD	7.2×10^{-16}	3.6×10^{-15}	0	7.5×10^{-16}	3.6×10^{-15}	0
scalar-Adam	7.3×10^{-16}	6.3×10^{-15}	0	7.7×10^{-16}	6.6×10^{-15}	0
Muon	1.9×10^{-15}	$8.4 \times 10^{-1\,\dagger}$	$-0.18^{\dagger}$	2.3×10^{-15}	1.0×10^{-10}	0
Adam, noise (10^{-7})	1.4×10^{-6}	7.8×10^{-6}	0	3.2×10^{-6}	2.0×10^{-6}	0
Adam, noise (10^{-5})	1.8×10^{-4}	7.8×10^{-4}	0	5.1×10^{-4}	2.2×10^{-4}	0
Adam, different init	1.4	1.4	-0.09	1.4	1.3	$+3 \times 10^{-3}$

In a larger run not shown in Table 17, we adapted all feed-forward layers with rank 16 adapters for 6000 training steps. The reported merge cost is the held-out loss of the averaged model minus the average of the held-out loss of the two models before merging. We found the merge cost to be -2.6×10^{-4} nats. It's important to emphasize that these results do not generalize to the claim that models with different adapters always merge well, and indeed the adapters were too weak to provide much improvement over the original model: we find that the adapters improve held-out loss on text by only about 0.05 nats. It's possible that stronger adapters might merge worse. Also, this is not evidence of linear-mode-connectivity without inspecting the loss along the path of interpolation (Frankle et al., 2020; Neyshabur et al., 2020; Wortsman et al., 2022).

Details on gauge-twin merging results (§5). For the merge results in §5, the two models are attention-twins trained end-to-end, not just adapters, from an identical random initialization, up to the gauge rotation, on the same stream. We average the final parameters of the two trained models after aligning them. That is, for each head, we find the orthogonal transformation A that applies identically to Q and K in each attention matrix of each layer and minimizes $\left\|W_Q^1 - AW_Q^2\right\|_F^2 + \left\|W_K^1 - AW_K^2\right\|_F^2$, the distance between the two models' factors, apply this map, and only then compute the averaged model's held-out loss.

Adam is not gauge-equivariant because of the denominator, which acts entrywise and does not commute with multiplication by an orthogonal map Q. This explains why Adam-incurred gauge-differences merge worse than scalar-Adam-incurred gauge-differences, which agree to floating-point error and therefore merge at no cost. The fact that Muon-incurred gauge-differences merge without cost, showing only numerical drift that is indistinguishable from its noise-twin's trajectory, shows that numerical drift alone does not produce a merge cost. Similarly, the fact that a tiny 10^{-7} perturbation at initialization incurs no merging cost under Adam addresses potential concerns of chaotic behavior in Adam, and so rules out Adam's fragility to small perturbations as the source of the cost. Alignment is checked separately: SGD's gauge twins cost $+0.68$ to merge without it and exactly 0 with it, so Adam's residual is not a failed alignment.

The twin models agree fairly closely in performance; Adam-twins have held-out losses that agree to 0.013 nats within an arm. The gauge-incurred difference is much smaller in scale than the difference between two identically-trained models trained from different initializations, with $+1.72$ and $+4.42$ nats merge cost on text and modular addition respectively, amounting to 3% and 0.1% of an ordinary difference, respectively. This helps calibrate the surprisingness of Adam's $+0.052$ (text) or $+0.005$ (modular addition, scored on the training objective the twins interpolate) nats merge cost for gauge-incurred differences. The equivariant methods cost between -0.0004 and 0, so nothing else in the setup produces a positive merge cost. The 1.32 nat symmetrized difference in predictions between Adam's final LoRA gauge twins on modular addition compares with only 1.5×10^{-10} for its 10^{-7} noise-twin.

No per-head rotation makes the twins identical. A per-head orthogonal alignment can only undo the gauge, so it can align two runs only if their invariant $W_Q^\top W_K$ agrees. Adam's twins' invariants do not agree, so nothing in that family closes the gap, and other reparameterizations such as head permutation or neuron alignment (Ainsworth et al., 2023) are outside this test. Procrustes also

minimizes distance between factors rather than merged loss, so this does not rule out that some other alignment would remove Adam's residual.

Why step 0 is exactly zero for LoRA but not for the CPU attention twins. For the first update of LoRA with Adam, Proposition A.2 applies verbatim because Adam's entrywise denominator does not commute with Q. Also, standard LoRA initializes the B-matrix to be 0, while initializing the A-matrix to be sampled from a random normal. As a result, the gradient with respect to A is zero, and gradient descent would perform an identical B_1A update, $-\eta\, GA^\top A$ in both twins, since $(Q^\top A)^\top (Q^\top A) = A^\top A$. In the CPU attention twins, by contrast, the rotated weights are nonzero at step zero, so rotating them costs rounding error. Thus, the gauge-twin logit difference is nonzero at step 0: it is 1.8×10^{-7}, compared to the noise-twin logit difference of 2.6×10^{-7} (Table 7). The $A{=}I$ twin is exactly 0, so this is one function evaluated in two bases, not a split.

E.9 Reparameterizing a model mid-training

Another important difference between equivariant and coordinate-wise update rules arises if we change the parameterization of the model, say, halfway through training. We might for example perform model surgery and rotate the factors of a LoRA adapter; re-base the head of a model; fold the scale of one layer into the next; and so on. In all of these cases, the function being computed stays the same.

What happens to θ_t and the optimizer state v_t (for Adam, its second moment) under such transformations?

If the update rule is equivariant, then nothing special needs to happen. By Definition 2.1, v_t will be transformed along with θ_t and training will continue on its way, converging to a gauge-equivalent version of its intended destination. As we saw in Table 17, there is no penalty for making the (simple) choice of an equivariant update rule when we want to be able to merge LoRA adapters.

If the update rule is coordinate-wise, the answer depends on what kind of reparameterization we're performing. Signed permutations are OK: if v_t stores the square of the gradient at the previous iteration, say, and we apply the signed permutation to the parameters, then permuting v_t the same way keeps it correct, because permutation and entry-wise squaring commute. But more general rotations will not be OK: squaring and (dense) right-rotation do not commute. Given a rotated θ_t and only the stored diagonal, there is no way to determine what the diagonal second moment matrix should be. One might want to keep the old v_t to retain the old coordinate system or basis, reset v_t and lose the gradient statistics it holds, or use an equivariant update rule. We do not measure which of the first two costs less.

E.10 The fork-time sweep

The merges above fork the twins at initialization, and this sweep asks whether the cost survives a shared prefix of training.

We pretrain a base model for K steps and make two copies of it, one of them rotated by the gauge; we refer to these as the twins, which continue training for another 1500 steps. Their optimizer states are reset. $K{=}0$ corresponds to splitting immediately after initialization. Both branches see the same sequence of training data. We merge them afterwards.

For modular addition, the Adam-related merge penalty is gone by $K{=}25$, the shortest prefix we tested. On the fork times we sampled the cost falls sharply between two of them, and a sparse grid cannot separate a threshold from a steep smooth decay. For the character LM, the cost is $+0.026$ to $+0.059$ over three random gauge draws at $K{=}25$ and $K{=}50$, compared to $+0.035$ to $+0.048$ at $K{=}0$, a separate run from the $+0.052$ of Appendix E.8. It then drops seventeen-fold to $+0.0022$ to $+0.0031$ at $K{=}100$. It continues to decrease afterwards: at $K{=}300$ we observe a slight improvement, -0.004. The determinism and noise twins merge at exactly 0 at every K on both tasks. For comparison, the ancestor's loss at the character LM's crossing is 2.50 nats and the twins' final loss is 1.77 nats. Over the same stretch, the different-init arm's merge cost increases from $+1.63$ to $+2.32$ nats. We see the same qualitative picture for signSGD, the other coordinate-wise optimizer we sweep: it has a cost of $+0.019$ to $+0.032$ for $K{=}25$ and is still higher than the noise arm at $K{=}100$.

At K=300 on text they still disagree in $W_Q^\top W_K$ by 0.28, against 0.54 at K=0. Indeed, if we merge them without accounting for the rotation, the merged model does worse than the twins by $+0.6$ to $+0.7$ nats.

GPT-2-small. As a larger-scale sanity check, we can measure the analogous phenomenon on the publicly available GPT-2-small checkpoint: we find that rotating the biases and pairs of columns corresponding to Q and K heads in attention, that is, according to the head gauge of §5, shifts its held-out loss by 1.2×10^{-7} nats. We can fine-tune two copies of GPT-2-small, one of which was rotated according to this transformation, using the same training data as detailed in Appendix E.11, and find that the gauge-transformed merge performs identically to the twins, with a cost of -7×10^{-5} nats, whereas the naïve merge is penalized. We don't investigate this setup further, since our goal is not to understand the extent to which model soups can be applied in practice but instead to understand optimizer behavior with regard to symmetries, for which merge cost serves as a convenient measurement.

E.11 Fine-tune twins on released GPT-2 weights

This subsection gives the protocol for the GPT-2-small fine-tune twins of Appendix E.10.

Model and run. GPT-2-small (Radford et al., 2019) has 12 layers, hidden dimension d=768, and 12 heads. We initialize from the public HuggingFace `gpt2` checkpoint (revision `607a30d783dfa663caf39e06633721c8d4cfcd7e`) via `transformers` 5.14 and fine-tune on byte-pair-encoded tiny-shakespeare with context 128, batch 8, and 500 steps in `float32` with eager attention and deterministic kernels. The released checkpoint holds 124M parameters, while Radford et al. (2019) tabulate 117M for GPT-2-small.

Arm definitions. Arms are (1) gauge $\times 3$: apply an independent orthogonal $A \in \mathbb{R}^{d_{\text{head}} \times d_{\text{head}}}$ to every head in every layer, leaving the V block untouched. In the fused `c_attn` projection of shape $(d, 3d)$, head h reads columns $[h\,d_{\text{head}}, (h{+}1)\,d_{\text{head}})$ of the Q and K blocks; the gauge acts as $Q_h \mapsto Q_h A$, $K_h \mapsto K_h A$ with the matching bias slices transformed identically; A is drawn and applied in `float64`. (2) A=I twin (determinism check); (3) noise twins add iid $\mathcal{N}(0, \sigma^2)$ to the same `c_attn` Q/K slices and bias slices, with $\sigma = 10^{-7}$ or 10^{-5} (stress case); (4) different-run: two independent fine-tunes of the same pretrained weights on different data streams, isolating optimization-path disagreement. Alignment uses the per-head Procrustes alignment of §5, with the bias slice stacked as an extra row of the block.

Results. Two rates are run because a null at a standard rate proves less than one at an aggressive rate. At the standard rate 5×10^{-5}, zero-shot held-out loss is 4.577, Adam reaches 3.548, scalar-Adam 3.764; the determinism twin merges at exactly 0 and the 10^{-7}-noise twin at 0 to five decimal places. Both fine-tuned Adam models improve by 1.03 nats over pretrained, and their aligned merge costs -7×10^{-5} nats, so it is free. We additionally test an aggressive learning rate 5×10^{-4}. There scalar-Adam's held-out loss is 6.62, meaning it has diverged, so we do not read its cells at that rate. At the standard rate, merging the two fine-tunes without alignment performs poorly, paying a penalty between $+0.85$ and $+1.19$ nats, while aligned merging at ten times the standard learning rate actively helps, improving on the inputs' mean by 0.046 nats.

E.12 The nuclear-norm reference solve

Muon outperforms the convex benchmark that gradient-flow theory points to. The reference solve of Table 2 solves $\min_{X \in \mathbb{R}^{n \times n}} \|X\|_*$ s.t. $\langle A_i, X \rangle = y_i$ for $i = 1, \ldots, m$ on each zoo instance (`experiments/nuclear_norm_reference.py`). The solver certifies feasibility but not optimality, as no duality gap is computed; its constraint residual is 7×10^{-16}. On two seeds it returns X^*; on the third, at $m = 2\,\text{dof}$, it returns a feasible point with strictly smaller nuclear norm of 38.416 versus X^*'s 38.509, meaning the relaxation is not tight there, and Muon still recovers X^*. That is evidence that its equal-rate terminal bias (Kang et al., 2026) need not minimize the nuclear norm. Gradient descent's recovery error is 0.131, which improves to 0.113 as the step size falls but does not reach Muon within budget; a finer grid on the same ladder reaches 0.108. Loss is sampled every 200 steps, so the faster methods are recorded well below the interpolation threshold (Adam's final loss is 1.2×10^{-11}). On a

separate, larger-rank 60×60 rank-5 problem that is not tabulated, recovery error is 0.000 for Muon versus 0.077 (gradient descent) and 0.465 (Adam) (`experiments/zoo_size_check.py`).

E.13 THE DIAL: GEOMETRIC-MEAN VARIANT, BALANCEDNESS, AND SCALE

Figure 2 (right) plots the dial at one scale, and this subsection gives its numbers in full.

The envelope and fixed-rate arms. Recovery decreases monotonically along the envelope arm of Figure 2 (right), going from 0.570 to 0.459, 0.348, 0.260 and 0.201 as p goes from 1 to 0. This arm was obtained by taking the best recovery over a common grid of learning rates for each p. Sweeping at fixed learning rates that interpolate gives the following ranges of recoveries (monotonically improving from p=1 to p=0): 0.576 to 0.201 for 10^{-3}, 0.570 to 0.256 for 3×10^{-3}, 0.573 to 0.357 for 10^{-2}, and 0.581 to 0.493 for 3×10^{-2}. Taking the best for each p thus gives rise to a monotonically decreasing recovery, agreeing with the plotted envelope to three decimal places for all $p\le 0.75$. How much of the recovery is restored depends on the learning rate; for example, the largest rate 10^{-1} stays in the range 0.55–0.62 and does not approach the flow-limiting regime, and there the ordering is not monotone (0.600 at p=1, 0.617 at p=0). Both arms were obtained with the same code, and the ε conventions of the other code paths differ below the digits we report (Appendix C).

The geometric-mean variant and balancedness. The geometric-mean dial leads to a qualitatively identical picture, with recovery decreasing monotonically as p falls. At p=0 we get a recovery of 0.229 (cf. the endpoint of the RMS dial, within 0.03) and a mean effective rank of 5.8. We do not use it in §4 only since it is not gauge-invariant in the p=0 limit. The balancedness drift of Proposition A.7 also falls with the dial, landing between Adam's and scalar-Adam's values in Table 2.

Larger scales. Qualitatively similar pictures are obtained at different problem sizes on the H100 as well; for scale n=128, recovery decreases monotonically from 0.67 to 0.40 as p decreases from 1 to 0 (averaged over five random seeds). Meanwhile, mean effective rank decreases monotonically as it goes from 36.6 to 12.0. For scale n=40 we obtain a qualitatively similar picture. There, recovery decreases monotonically going from 0.52 to 0.14 as p decreases from 1 to 0, averaged over ten random seeds and a different, finer, grid of learning rates in order to extend the flow-limiting regime. Because these are ten H100 seeds rather than three CPU ones, the endpoint at p=1 does not agree with the 0.56–0.58 that §4 quotes for Adam across Table 5's rates, being below it. Both endpoints are selected at the learning rate 3×10^{-4}, below every rate the zoo grid carries. See Appendix C for a discussion of the learning-rate grid.

E.14 THE DIAL IN PSEUDOCODE

Why Line 8 is the equivariant step. By pooling $\hat{v}$ over all entries of the two factors, we make $\bar{s}$ depend only on the history of $\|G_U\|_F^2+\|G_V\|_F^2$, which is invariant under gauge transformations by Lemma B.2. The second-moment term v_θ is kept elementwise for all p, including p=0. Because taking the entry mean commutes with the exponential moving average, the pooled $\bar{s}^2$ equals the bias-corrected scalar v_t of Proposition A.1(2) exactly. This makes the parameter trajectory the equivariant one.

One point of bookkeeping, since it is easy to misread. At p=0 the stored v_θ is elementwise and admits no σ_Q of its own, because squaring does not commute with a right rotation. However, the non-pooled components never reach the update. The step at p=0 depends on v_θ only through the pooled scalar $\bar{s}$, and otherwise only on $\hat{m}_\theta$. Both of these do transform. That is exactly the case Definition 2.1 allows, since it quantifies over the state the update reads.

An implementation that stored the scalar alone would be equivariant on its full state and would produce the identical trajectory. We keep v_θ so that one code path serves every p.

The ε convention. For $p<1$ an extra ε sits outside the product as well, so at p=0 the denominator is $\bar{s}+2\varepsilon$ where Proposition A.1(2) has $\bar{s}+\varepsilon$. With $\varepsilon=10^{-8}$ this $O(\varepsilon)$ offset has no effect on any number reported here. At p=1 that extra ε is dropped and the denominator is stock Adam's $\sqrt{\hat{v}_\theta}+\varepsilon$.

Algorithm 1 Adam-p: the preconditioner-anisotropy dial (§4). Setting p=1 recovers standard Adam; p=0 gives the gauge-equivariant scalar-Adam from Proposition A.1(2). Only the denominator moves with p: the gradient, moment estimates, bias corrections, step size, and (zero) weight decay are the same throughout. All operations on the two factors are elementwise: $\odot$, $\oslash$, $(\cdot)^{\odot q}$ denote elementwise multiplication/division/power. Line 8 is the only inter-factor cross-talk.

Require: factors U, V; step size η; $(\beta_1, \beta_2) = (0.9, 0.999)$; $\varepsilon = 10^{-8}$; dial $p \in [0,1]$; weight decay 0

1: $m_\theta \leftarrow 0, \quad v_\theta \leftarrow 0$ for each factor $\theta \in \{U, V\}$
2: **for** $t = 1, 2, \ldots$ **do**
3: **for** each factor $\theta \in \{U, V\}$ **do**
4: $g_\theta \leftarrow \nabla_\theta L(U,V)$ ▷ $L(U,V) = f(UV^\top)$; covariant: $g_\theta \mapsto g_\theta Q$
5: $m_\theta \leftarrow \beta_1 m_\theta + (1-\beta_1)\, g_\theta, \quad v_\theta \leftarrow \beta_2 v_\theta + (1-\beta_2)\, g_\theta^{\odot 2}$
6: $\hat{m}_\theta \leftarrow m_\theta / (1-\beta_1^t), \quad \hat{v}_\theta \leftarrow v_\theta / (1-\beta_2^t)$
7: **end for**
8: $\bar{s} \leftarrow \big(\mathrm{mean}(\hat{v}_U, \hat{v}_V)\big)^{1/2}$ ▷ one mean, pooled over *both* factors: gauge-invariant
9: **for** each factor $\theta \in \{U, V\}$ **do**
10: $D_\theta \leftarrow \big(\hat{v}_\theta^{\odot 1/2} + \varepsilon\big)^{\odot p} \odot (\bar{s} + \varepsilon)^{1-p}$ ▷ p=1: per-coordinate; p=0: one shared scalar
11: **if** $p < 1$ **then**
12: $\theta \leftarrow \theta - \eta\, \hat{m}_\theta \oslash (D_\theta + \varepsilon)$
13: **else**
14: $\theta \leftarrow \theta - \eta\, \hat{m}_\theta \oslash D_\theta$ ▷ $D_\theta = \hat{v}_\theta^{\odot 1/2} + \varepsilon$ here: stock Adam
15: **end if**
16: **end for**
17: **end for**

Denominator convention: at p=0 it is $\bar{s} + 2\varepsilon$ and at p=1 it is $\hat{v}_\theta^{\odot 1/2} + \varepsilon$; both are the ideal value up to $O(\varepsilon)$.

E.15 The row-pooled control

The dial of §4 changes two things at once. Going from p=1 to p=0 removes the anisotropy (one denominator for every coordinate of both factors) and restores gauge equivariance. This control arm keeps anisotropy, with one denominator per row, and restores only the symmetry, so the two effects can be told apart.

The rule. Pool the second moment over each row of a factor matrix, rather than over each coordinate. Run Adam's first and second moments as usual, with the same β_1=0.9, β_2=0.999 and ε=10^{-8} and the same bias correction. Then, for each factor $P \in \{U, V\}$ with rows indexed by i, use one denominator per row,

$$\Delta P_{i\cdot} \;=\; -\eta\, \frac{\hat{m}_{i\cdot}}{\sqrt{\frac{1}{k}\sum_{j=1}^{k} \hat{v}_{ij}} + \varepsilon}.$$

The denominator varies from row to row, so the rule is anisotropic. It is also gauge-equivariant. Under $(U,V) \mapsto (UQ, VQ)$ the gradient rows transform as $G_{i\cdot} \mapsto G_{i\cdot} Q$ (Lemma B.2), so $\sum_j \hat{v}_{ij}$, an average of past $\|G_{i\cdot}\|^2$, is unchanged and the update transforms the same way the parameters do. This is the matrix-shaped case of the vector pooling of Ling et al. (2022), and it is an instance of Definition 2.1 with σ_Q acting on the first moment alone.

Protocol. The testbed, seeds and rate grid are those of §4: n=40, planted rank 3, k=40, $m/\mathrm{dof} = 2.0$, initialization scale 10^{-3}, no weight decay, seeds $\{42, 123, 456\}$, rates $\{10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}\}$, and the 10^{-7} interpolation bar with a 3×10^4-step cap. Each arm is reported at the rate giving the best recovery among those that interpolate, the selection rule §4 applies to every method.

Gauge check. Running 60 steps at η=10^{-2} from a start and from its right-rotated twin, and comparing the products $UV^\top$, the row-pooled arm differs by 2.0×10^{-7} in relative Frobenius distance. This is the float32 noise floor of this harness (Appendix E.18). Adam under the same check differs by 0.34. So the arm is equivariant in practice and not only on paper.

Table 18: Row pooling reduces recovery error, and pooling both factors reduces it further. The three methods differ only in second-moment pooling. Each uses its best interpolating rate on the shared grid, evaluated over three sensing seeds. Lower recovery is better.

arm	pooled over	equivariant	recovery	s.d.	erank	lr
Adam (p=1)	nothing	no	0.5704	0.0505	14.49	3×10^{-3}
row-pooled	a factor row	yes	0.4609	0.1063	9.00	10^{-3}
scalar-Adam (p=0)	both factors	yes	0.2010	0.0381	5.43	10^{-3}

Result. Table 18 gives the three arms. Restoring equivariance while keeping the anisotropy buys back about 30% of the recovery distance from Adam to scalar-Adam, and more than half of the effective-rank distance. The remaining recovery gap needs the anisotropy to go as well.

Two notes on reading this. First, the three arms interpolate at different rates and in different numbers of steps, which is the same confound the fixed-rate arm of Appendix E.13 addresses for the dial. At the matched rate 10^{-3} the three read 0.576, 0.461 and 0.201, so the split is not an artifact of the selection. Second, row pooling is one point in a space of pooling structures. This experiment separates row-level from coordinate-level pooling and says nothing about the ones in between.

E.16 The factor-aware preconditioner of Remark 3.5

Remark 3.5 needs a method that is gauge-equivariant and still recovers badly. That would show equivariance is not sufficient for good recovery. This subsection presents such a method. It is our own variant in the style of ScaledGD (Tong et al., 2021), not the published algorithm, and it carries none of that paper's guarantees.

The rule. With damping $\lambda > 0$ and rate η,

$$U_{t+1} = U_t - \eta\, G_t^U \left(V_t^\top V_t + \lambda I_k\right)^{-1}, \qquad V_{t+1} = V_t - \eta\, G_t^V \left(U_t^\top U_t + \lambda I_k\right)^{-1},$$

where G_t^U, G_t^V are the gradients of L at (U_t, V_t). It is equivariant: under $(U,V) \mapsto (UQ, VQ)$ we get $V^\top V \mapsto Q^\top V^\top V Q$ and $G^U \mapsto G^U Q$, so the whole update picks up one factor of Q on the right, as Definition 2.1 asks. Damping is what keeps it defined at the small initialization, where $V^\top V$ is near-singular.

Grid and selection. Same testbed, seeds and bar as Appendix E.15. Rates $\{10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}\}$ crossed with damping $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$, selected on best recovery among the cells that reach the 10^{-7} bar. Three cells diverge, all at η=10^{-1} with $\lambda \le 10^{-1}$.

Result. The selected cell is η=10^{-1}, λ=1: recovery 0.6092 (s.d. 0.0135; per seed 0.5945, 0.6210, 0.6119), erank13.68, train loss 8.4×10^{-12}, 200 steps.

Recovery is worse than Adam's 0.5734, so an equivariant method can finish outside the equivariant cluster of Figure 1 entirely. Every interpolating cell in the grid but one recovers between 0.61 and 0.94, so this is not one unlucky corner. The exception, η=3×10^{-2} at λ=10^{-3}, interpolates but has recovery error 3.54.

Damping is the axis that matters. At λ=1 every rate gives 0.61, and as λ falls toward 10^{-3} recovery degrades to 0.85–0.88 at the other rates, with effective rank rising from 13.5 to 30. The reading is the schedule axis of §6: preconditioning by the other factor's Gram matrix flattens the spectrum (effective rank 13.68 at the selected cell).

E.17 Breaking the head gauge on purpose

Silverstein et al. (2026) have already noted the head symmetry of §5, and they break it on purpose: they add a random (unlearned) bias to the q and v vectors each batch, in the spirit of a Hamiltonian/Noether framework for memory-efficient optimizers. Here we leave it alone and look at what happens with common choices of optimizer.

E.18 THE PROTOCOL IN FULL

Every experiment in the paper follows the rules below. We omit weight decay in all of our experiments studying the implicit bias of optimizers, as regularization can obscure implicit effects. All experiments are trained until interpolation is achieved, unless otherwise noted. For example, the apparently successful run of sign-descent in Appendix D.8 with stopping tolerance 5×10^{-2} only looks successful because it stops early. All learning rates are fixed or have a cosine schedule, except one linear-decay control in Appendix D.11. In the case of optimizers that have a fixed-norm update such as Muon, signum, and Lion, we use learning rate decay to give a more even comparison to other methods. This is a per-method criterion rather than a class-level one, informed by observations rather than theory. To ensure that this is not confounding the results, we also try learning rate decay (full schedule) for all optimizers in our experiments (Appendix D.1). To avoid outcome and speed bias, we show the full recovery vs. learning rate curves in our experiments (Appendices D.2 and D.3). Finally, we report recovery and final rank together, use the same random seeds for all methods, and use deterministic full-batch updates where relevant (§5).

E.19 ATTENTION TWINS AT SCALE AND UNDER MINI-BATCH NOISE

To remove confounders and show that gauge-breaking at step 1 is the effect of interest, we extend our toy modular addition task and train a larger-scale character-level language model on tiny-Shakespeare (Karpathy, 2015).

Gauge-breaking at scale. In Table 7, step-0 values are a rounding floor rather than a split, for the reason Appendix E.8 gives. The runs below are deterministic GPU training runs using `float64` precision, with $A{=}I$ twins exactly zero. We use mod-97 addition (the main text reports mod-47) at two sizes, 4 layers with $d{=}128$ and 6 layers with $d{=}256$. We run six initialization and rotation-draw pairs per configuration for Adam, and two per equivariant method. We report the relative difference between twin functions, which is informative at step 1 but is not so at convergence for these larger problems. Gauge-equivariant optimizers show a difference on par with their own noise twins, which is consistent with their late-time difference being amplification of floating-point precision effects and not a meaningful example of basis-dependence. This is a diagnostic reading, not an identification of the mechanism: we do not estimate Lyapunov exponents or measure a growth rate for the perturbation. Adam breaks the gauge at step 1 and grows well beyond its own noise twins.

Gauge-breaking under mini-batch noise. To check that this survives stochastic training and confirm that Adam is basis-dependent while Muon and scalar-Adam are not, we train on tiny-Shakespeare, where inputs are minibatched and both twins receive the identical minibatch stream. We use a 6-layer $d{=}256$ transformer and compare attention twins trained with Muon, scalar-Adam, and Adam. All optimizers see the same minibatches each step. Results are shown in Table 8. Muon and scalar-Adam gauge twins exhibit relative differences comparable to their respective noise twins, that is, the same optimizer started from a small additive perturbation in the same basis, which is consistent with any late-time divergence being amplified finite-precision error rather than basis dependence. By contrast, Adam breaks the gauge immediately (step 1) and eventually achieves different behavior. Notably, the two instances of Adam converge to nearly-identical validation losses of 1.579 vs. 1.585, meaning that any functional differences are hard to observe from loss.

Two apparent instabilities are not basis-dependence:

- Muon can appear to be unstable for modular addition due to sensitivity to small singular values in its Newton–Schulz iteration.
- Scalar-Adam can appear to be unstable at depth, but this is due to low-precision effects amplified by minibatching and depth, not basis-dependence (Appendix D.4).

E.20 DO THE TWO METHODS FIT THE SAME MATRIX, OR ONLY SCORE THE SAME?

Table 19 suggests that GD and scalar-Adam might be finding similar predictors despite having very different convergence rates. We can test this hypothesis by measuring the distance between the fitted matrices directly. We compare GD to scalar-Adam, matching instance, initialization seed, mask, preprocessing, GPU `float64` execution, loss normalization, etc., and measuring at matching

Table 19: Scalar-Adam approaches GD's error and effective rank as its learning rate falls. Entries show held-out root-mean-square error (RMSE, $\times 10^{-2}$) / effective rank, averaged over four seeds in GPU `float64`. Training loss is matched at $\leq 10^{-5}$. Adam-variant rates span 3×10^{-3} to 10^{-6}. m/dof_{24} counts observations per degree of freedom at rank 24. Slope measures scalar-Adam's log–log approach to GD over the five smallest rates. Best RMSE is Adam's lowest grid error. Adam floor is its lowest grid effective rank, not a limiting value. Gap subtracts scalar-Adam's rank at its smallest reaching rate from Adam's lowest grid rank on the same seed (paired mean $\pm$ standard deviation). Bold floors have gaps whose 95% t-intervals exclude zero. Direct matrix comparisons: Appendix E.20.

		equivariant		coordinate-wise Adam			
dataset	m/dof_{24}	GD	scalar-Adam	slope	best RMSE	Adam floor	gap
Indian Pines	1.15	1.514 / 13.57	1.515 / 13.58	0.97	1.467	13.68	$+0.09 \pm 0.11$
	1.9	1.262 / 13.30	1.262 / 13.31	0.93	1.264	**13.75**	$+0.44 \pm 0.06$
Pavia Univ.	1.15	0.867 / 6.34	0.868 / 6.35	1.12	0.892	**6.88**	$+0.53 \pm 0.08$
	1.9	0.631 / 5.60	0.632 / 5.60	0.99	0.679	**6.21**	$+0.61 \pm 0.06$

95% t-intervals on the gap (3 degrees of freedom), top to bottom: $[-0.08, 0.26]$, $[0.35, 0.52]$, $[0.40, 0.65]$ and $[0.51, 0.70]$.

Table 20: Scalar-Adam approaches GD's fitted matrix, while Adam remains separated. Entries are mean relative Frobenius distances over four seeds, in percent, at matched training loss $\leq 10^{-5}$. $d_W = \|W - W_{\mathrm{GD}}\|_F / \|W_{\mathrm{GD}}\|_F$. m/dof_{24} counts observations per degree of freedom at rank 24. Floor is the distance between GD runs at two reference rates. Ratio divides Adam's distance by scalar-Adam's at 10^{-6}, before rounding.

			scalar-Adam			Adam			
dataset	m/dof_{24}	floor	$3{\cdot}10^{-3}$	10^{-4}	10^{-6}	$3{\cdot}10^{-3}$	10^{-4}	10^{-6}	ratio
Botswana	1.15	0.004	22.990	4.008	**0.059**	36.22	17.06	**17.80**	300×
	1.9	0.002	17.396	2.971	**0.031**	32.37	14.65	**15.28**	499×
Indian Pines	1.15	0.017	21.625	3.804	**0.045**	24.93	17.72	**17.48**	385×
	1.9	0.008	10.821	1.916	**0.028**	15.68	14.20	**13.50**	482×
Pavia Univ.	1.15	0.009	5.179	0.847	**0.004**	6.95	5.61	**6.85**	1610×
	1.9	0.003	1.830	0.303	**0.003**	3.58	3.87	**4.78**	1700×
Salinas	1.15	0.023	23.271	3.954	**0.044**	18.49	11.55	**12.79**	293×
	1.9	0.008	9.573	1.497	**0.020**	8.10	6.54	**7.84**	387×

training-loss values that agree to within $1/50$, taking GD's rate to be our previously-chosen reference rate. The distance is

$$d_W(\eta) \;=\; \frac{\left\|W_{\mathrm{scalar},\eta} - W_{\mathrm{GD,ref}}\right\|_F}{\left\|W_{\mathrm{GD,ref}}\right\|_F}.$$

We use four paired seeds, the Pavia University dataset and the two sampling densities of Table 19, with full code in `predictor_distance2.py`. The results, evaluated at the 10^{-5} loss value, appear in Table 22. As rate decreases, d_W (averaged over seeds) monotonically decreases by a factor of about 33 to 36 in both densities. Also, the held-out error and effective rank differences $|\Delta\mathrm{RMSE}|$ and $|\Delta\mathrm{erank}|$ fall with d_W, which provides an explanation for the matching values in Table 19: GD and scalar-Adam predictors are getting close. This rules out the possibility that Table 19's agreement on RMSE and effective rank is a coincidence. We can rule out noise in the GD reference because the average and maximum relative distance between predictors from different GD runs, at rate 10 versus rate 30 evaluated at matching thresholds, are 6×10^{-5} and 1.0×10^{-4}, at least fivefold smaller than the minimum d_W value. However, three limitations apply.

1. Table 22's rates only go as far as 10^{-5}, one order of magnitude short of Table 19's minimum of 10^{-6}, so the smallest d_W here is small but not zero.
2. The log–log slope of d_W as a function of rate is 0.61 and 0.60 across the displayed rates and 0.76 and 0.73 across the three smallest. We do not make a first-order rate claim from these slopes, as Lemma A.6 concerns a different time interval and stopping value.
3. Looking at matching loss values across a finite number of checkpoints only shows predictor convergence over our range of rates, not convergence to a common limit.

Table 21: At the smallest rates, scalar-Adam's rank offsets are small while Adam's remain positive on these two scenes. Entries subtract GD's effective rank at matched training loss $\leq 10^{-5}$, averaged over four seeds. Each cell reports scalar-Adam / Adam at the same learning rate η. The grid extends Table 19. Its smallest rates resolve the Indian Pines gap left uncertain in Table 19. These are signed rank differences, not matrix distances (Table 22).

	Indian Pines		Pavia Univ.	
η	m/dof_{24}=1.15	1.9	1.15	1.9
3×10^{-3}	7.319 / 6.867	3.524 / 3.156	2.012 / 1.920	0.720 / 0.797
10^{-3}	3.696 / 3.385	1.732 / 1.883	1.001 / 0.979	0.349 / 0.607
3×10^{-4}	1.936 / 1.404	0.881 / 1.370	0.509 / 0.605	0.173 / 0.631
10^{-4}	1.028 / 0.503	0.465 / 1.083	0.268 / 0.533	0.092 / 0.684
3×10^{-5}	0.435 / 0.148	0.201 / 0.708	0.115 / 0.599	0.041 / 0.727
10^{-5}	0.167 / 0.108	0.081 / 0.442	0.045 / 0.847	0.017 / 0.931
3×10^{-6}	0.049 / 0.420	0.025 / 0.624	0.012 / 1.191	0.005 / 1.127
10^{-6}	0.012 / 0.778	0.007 / 0.802	0.001 / 1.349	0.001 / 1.180
3×10^{-7}	−0.001 / 1.208	−0.000 / 0.947	−0.003 / 1.452	−0.001 / 1.202
10^{-7}	**−0.005 / 1.492**	**−0.002 / 1.062**	**−0.004 / 1.539**	**−0.001 / 1.218**

Table 22: Scalar-Adam's fitted matrix approaches GD's as the learning rate falls. Pavia University runs share initialization and observation masks across four paired seeds, with training loss matched at $\leq 10^{-5}$. The sampling densities match Table 19. d_W is relative Frobenius distance to GD (mean $\pm$ standard deviation). GD uses its selected rate. $|\Delta\mathrm{RMSE}|$ and $|\Delta\mathrm{erank}|$ are absolute differences in root-mean-square error and effective rank, the statistics in Table 19. Table 20 adds Adam and extends the comparison to four scenes.

m/dof_{24}	scalar-Adam rate	d_W	$\lvert\Delta\mathrm{RMSE}\rvert$	$\lvert\Delta\mathrm{erank}\rvert$
1.15	3×10^{-3}	0.0518 ± 0.0042	3.4×10^{-3}	2.01
	1×10^{-3}	0.0282 ± 0.0024	1.7×10^{-3}	1.00
	3×10^{-4}	0.0153 ± 0.0014	8.4×10^{-4}	0.51
	1×10^{-4}	0.0085 ± 0.0008	4.4×10^{-4}	0.27
	3×10^{-5}	0.0037 ± 0.0004	1.8×10^{-4}	0.12
	1×10^{-5}	$\mathbf{0.0015\pm0.0002}$	7.1×10^{-5}	0.04
1.9	3×10^{-3}	0.0183 ± 0.0011	8.1×10^{-4}	0.72
	1×10^{-3}	0.0099 ± 0.0006	3.7×10^{-4}	0.35
	3×10^{-4}	0.0054 ± 0.0004	1.7×10^{-4}	0.17
	1×10^{-4}	0.0030 ± 0.0002	8.7×10^{-5}	0.09
	3×10^{-5}	0.0014 ± 0.0001	3.8×10^{-5}	0.04
	1×10^{-5}	$\mathbf{0.0006\pm0.0001}$	1.5×10^{-5}	0.02

E.21 The matched-loss trajectories and the density sweep

Figure 4 further investigates the learning dynamics. GD's fitting is consistent with a greedy pattern that fits the largest singular values first, steadily decreasing the test error. Adam's test error actually increases from 0.0253 to 0.0271 when the training loss decreases from 3×10^{-4} to 3×10^{-5}, and so does the effective rank, from 22 to 28. Muon's effective rank stays close to the model's rank cap until quite late in training, then decreases from 46 to 36, and its held-out error improves little until then. These observations suggest that deep fitting in the interpolation regime with a gauge-breaking optimizer can hurt generalization performance on the real data.

Adam at a vanishing learning rate. In §6, we include the result of a sweep of the learning rate on Pavia (density 0.28), in order to understand the importance of tuning. At all learning rates on this grid, Adam has a higher effective rank than GD (Table 21). As the learning rate η approaches 0, Adam's held-out error deteriorates from 0.00892 to 0.00961 and its effective rank consistently increases from 6.88 to 7.69. That is only one to two standard deviations on its own, but the direction holds in all 8 paired comparisons (four seeds, two statistics).

The two scenes, and Muon's inversion. Next we comment on the differences between the Pavia and the (preprocessed) Indian Pines data, which exhibit an inversion of Muon performance: Muon

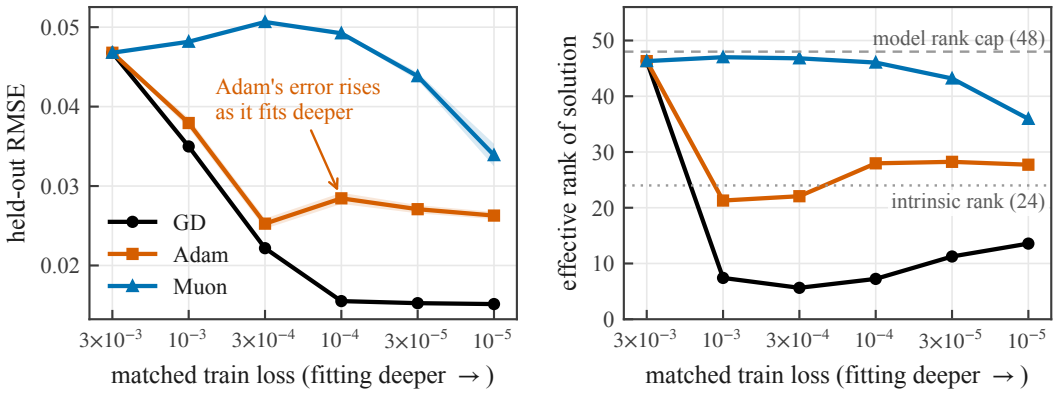

Figure 4: Held-out performance against training loss on real data (Indian Pines, $m/\text{dof} \approx 1.15$, 4 seeds, GPU `float64`; learning rates from the train-only rule in Appendix D.5). GD sits below Adam at every matched loss past the shared starting point. *Left:* held-out error, with ± 1 s.d. bands over the 4 seeds. The bands sit inside the line width wherever the seeds agree, which is the whole path for GD. *Right:* effective rank. GD settles between 5.6 and 13.6. Adam climbs to 28, against the scene's intrinsic rank of 24. Muon tracks the rank cap until convergence, then drops sharply.

performs worse than Adam, and does not outperform GD anywhere on the latter. Measured on the preprocessed matrices, Indian Pines carries 2.6 to 5.5 times Pavia's tail amplitude at every reference rank we have tested, so the ordering of the two scenes does not depend on the convention, even though the absolute placement does. If we naively trust the synthetic sweep, Muon should perform better than Adam at the observed value of $\tau = 0.251$ for Indian Pines, since Adam is worst at every τ of Table 15. However, on Indian Pines Muon's schedule has not begun to concentrate the spectrum, and that is not the schedule the phase diagram describes. Appendix D.5 discusses the likely cause and leaves it open.

The density sweep. Next we investigate the effect of the sampling density on the observed gap between Adam and GD. At higher density, GD's relative advantage becomes smaller, from $+44\%$ at $m/\text{dof} = 1.15$ to $+28\%$ at $m/\text{dof} = 1.9$, consistently over four seeds. In an earlier interpolation-based evaluation, which we do not tabulate, this gap appeared to approach zero at $m/\text{dof} \approx 3$; we have since replaced that protocol with the protocol of training to comparable loss.

Where the comparison is taken. At training loss $\leq 10^{-5}$, the relative advantages are $+42.4\%$ and $+27.5\%$ for Indian Pines, which is comparable to the gap at training loss $\leq 3 \times 10^{-5}$ of $+44\%$ and $+28\%$. This shows that the observed gap does not significantly depend on how far into interpolation one evaluates.

E.22 The FlowAdam repair

A recent paper (Singh & Sheel, 2026) proposed a variant of Adam they call FlowAdam, which incorporates the (clipped) gradient-flow velocity $v \mapsto \text{clip}(v, -c, c)$ directly into Adam's momentum mechanism, reflecting the bias dynamics that Theorem 3.4 describes in its memoryless, idealized form, though the stateful hybrid falls outside the theorem's formal reach. Running this method without weight decay achieves an error of 0.347 on our recovery task, which is better than Adam's 0.573 but still far from GD's 0.131 benchmark. We can explain this gap as follows. Clipping each coordinate of v separately breaks gauge symmetry per Proposition 3.1. We can repair FlowAdam by clipping the norm of the entire vector instead, which limits step-size but does not change direction. This operation is gauge-symmetric and yields improved results. Applied in isolation with a fixed learning rate, it improves FlowAdam's recovery error by 37%, reducing it to 0.219 when the training loss reaches 6×10^{-7} (see Appendix C). Raising the threshold c further improves recovery to 0.113, equal to the error achieved by GD at the lowest interpolating learning rate in Table 5. However, unlike GD, this method does not achieve the 10^{-7} interpolation bar at these learning rates, so neither number is an interpolating result. The reason for the gap at 0.219 is that Adam's coordinate-wise preconditioning also breaks gauge symmetry, so with clipping alone FlowAdam as a whole stays gauge-breaking.

FlowAdam-p. Next we combine FlowAdam's norm-clipping with the preconditioner-anisotropy dial from §4 to get a gauge-symmetric method (at p=0) that, at the dial's fixed rate 10^{-3}, gives the best recovery among the Adam-type variants meeting the 10^{-7} interpolation bar. At a learning rate of 10^{-3}, this method, which we call FlowAdam-p, achieves a 0.169 recovery error, improving on the 0.201 achieved by the dial alone at the same rate (+15.9%). It also has a final effective rank between 4.8 and 5.4 depending on the seed, which is closer to GD's 4.51 than any of the methods we've seen that start from Adam. Unlike the 0.113 run above, FlowAdam-p reaches our standard 10^{-7} threshold at this learning rate once its step budget is extended, with the same 0.169 (Appendix D.9), so its number is an interpolating result. We find that on finer learning-rate grids with smaller rates, FlowAdam-p's empirical limit on recovery error converges to the limit of the dial alone, suggesting that clipping offers an improvement for a fixed learning rate, but that the preconditioner-anisotropy knob can achieve comparable results given a small enough learning rate. For instance, at n=40, the dial alone achieves a 0.143 recovery error, while FlowAdam-p achieves 0.148. At n=128, they achieve errors of 0.40 and 0.42, respectively. We report additional details about FlowAdam-p in Appendix D.9.

Interpolation and early stopping. To ensure that FlowAdam-p's n=40 cell is compared at our standard threshold for interpolation of 10^{-7} training loss, we increase its training steps from the sweep's 3×10^4 to 6×10^4. All ten seeds interpolate by this point, with an average recovery error of 0.1475 ± 0.0357. Finally, to ensure that the 0.169 at 10^{-3} was not artificially low by stopping before convergence to the local minimizer, we train that run for 1.2×10^5 steps (Appendix D.9). We find that the recovery error stays approximately constant once the training loss goes below 10^{-4}, well before our interpolation threshold of 10^{-7}.

E.22.1 Where the bias succeeds and where it does not

Importantly, the implicit bias does not come with its own coefficient, and so its effective regularization varies along with other hyperparameters such as the learning rate, the initialization scale, and the dial of §4.

Starting from the setup of Appendix E.22, FlowAdam has been empirically tested along three axes: the density of available observations, the conditioning of the target signal, and the need for explicit regularization (Appendix C). Overall, a more flow-based/equivariant bias helps beyond what is offered by weight decay only when the amount of explicit regularization the task needs stays small and structured. On well-conditioned and densely-sampled tasks, it delivers an improvement of 3–6% over Adam across all five random initializations at a given amount of L2 regularization (results not tabulated). However, if the task requires strong regularization, say, due to sparser sampling or an intrinsically low-rank and ill-conditioned target, FlowAdam can be beaten by Adam. The comparison at a given amount of explicit L2 regularization should not be misconstrued as contradicting the weight-decay-free comparisons of §§4–6.

The two limitations that follow are conjectures based on the procedure's theoretical properties, not on empirical observation.

Grokking. Another caveat regards tasks in the manner of grokking (Power et al., 2022): our conjecture is that FlowAdam can hurt them. Intuitively, the optimal trajectory might want to generalize only after memorization, when task-specific gradient signals have died off: at such a point, regularization-type signals, for example, weight decay, become key, while geometric flow can be less helpful. We have not investigated grokking empirically, nor do we assert that it does in fact require weight decay.

LoRA fine-tuning. Finally, let us discuss LoRA (Hu et al., 2022), a form of low-rank adaptation. It posits an update of the form $\Delta W = BA$ with smaller-than-W factors B and A, thereby limiting the rank of ΔW from above. Such an architectural decision impacts the rank-selection bias measured in §4, so LoRA is not directly comparable with our full-recovery experiment. Nevertheless, BA may converge to have rank below its theoretical maximum. Moreover, the gauge freedom of §2 is retained verbatim, in that ΔW is unchanged by a reparameterization $(B,A)\mapsto(BQ, Q^\top A)$ with $Q\in \mathrm{O}(r)$, and so Proposition A.2 still applies to the first step. Overall, LoRA forecloses one empirical implication of our effect, but the effect itself, tracked in §5, should be unaffected.

Table 23: Prior work studies related symmetries, while this paper tests which fitted solutions optimizers select.

work	setting and result	relation to this paper
VectorAdam (Ling et al., 2022); Xie et al. (2025)	Examine ambient rotations of vector parameters under general loss functions. Show Adam is sensitive to such rotations, whereas shared scalar updates restore equivariance.	We focus instead on an internal factorization gauge and connect the same principle to solution selection among interpolants, rather than optimization speed.
LoRA-RITE (Yen et al., 2025); DePavia et al. (2025)	Study transformations of LoRA factors and rotations in data or feature space. Demonstrate improved optimization through invariance and altered implicit bias in Adam under rotations.	We isolate the intrinsic gauge of the factorization itself and evaluate recovery performance at matched training loss across a range of optimizers.
Silverstein et al. (2026); Zhang (2026)	Analyze per-head query–key (QK) rotations and Schur multiplicity bases. Highlight that optimizer design can intentionally break or preserve symmetry.	We maintain the gauge structure intact and use it to analyze solutions chosen by standard, deployed optimizers.
Lau & Su (2026); Shirodkar (2026)	Investigate architectural symmetry groups and construct optimizers that exactly respect equivariance.	These works support symmetry-aware design; we instead assess the practical consequences of preserving or violating such symmetry.
Kang et al. (2026); Dragutinović et al. (2026)	Study Muon's spectral dynamics and its impact on simplicity bias.	We extend this by mapping conditions under which uniform spectral growth helps or harms performance as the target develops a spectral tail.

F Extended related work

Table 23 sorts the nearest work by which symmetry it acts on.

Low-rank bias of gradient flow. Implicit bias of gradient flows in factored models has been extensively studied. In particular, in the context of the matrix sensing problem, in the underdetermined regime, it was first conjectured by Gunasekar et al. (2017), and shown for commuting measurement matrices, that the overparameterized solution converges to the minimum-nuclear-norm interpolant. Their setting factorizes the target as $UU^\top$ and optimizes U by gradient flow from an infinitesimal random initialization. Later, Arora et al. (2019) showed that such implicit bias extends to deep matrix factorizations. They advocated a dynamical, rather than norm-based, point of view, later refined in, e.g., Razin & Cohen (2020), as well as in the context of the greedy low-rank learning dynamics in Li et al. (2021).

Gradient flow has been studied in factored models by Arora et al. (2018); Du et al. (2018), who noticed that the balancedness $U^\top U - V^\top V$ is a conserved quantity. The corresponding symmetry is the general linear gauge symmetry $(U,V) \mapsto (UA, VA^{-\top})$ for $A \in \mathrm{GL}(k)$, which, in the Noether sense of Kunin et al. (2021), is what conserves it, rather than the orthogonal gauge we study. Our internal orthogonal symmetry is the subgroup $A \in \mathrm{O}(k)$, which acts on the shared latent index of both factors. Whether an optimizer's update commutes with it bears on which interpolant is selected. Recent work by Chou et al. (2024a) studied the dynamics of deep factorizations and showed that the low-rank bias becomes more pronounced with depth. Chou et al. (2024b) showed that weight-normalized models exhibit a low-rank bias robust to the initialization scale. All this work considers gradient descent or flow, and does not discuss optimizers that are deployed in practice.

Margins and norms. Implicit bias of Adam and other adaptive optimizers was first considered by Wilson et al. (2017), who showed that adaptive methods find solutions with worse generalization. Recent work has made progress on AdamW (Loshchilov & Hutter, 2019), Adam, and related algorithms. See, for example, Xie & Li (2024), who characterize AdamW through an ℓ_∞-norm

constraint, Zhang et al. (2024), who show Karush–Kuhn–Tucker convergence for Adam, or Baek et al. (2026), who give per-sample refinements. Most of this work builds on Soudry et al. (2018) and considers the classification margin in separable classification problems. Another line of work, which is not margin-based, includes backward error analyses of Adam and RMSProp, showing that the corresponding modified flows penalize a perturbed one-norm of the loss gradients or, more often, slow down its decrease (Cattaneo et al., 2024). Li et al. (2025) identify a sharpness that Adam reduces near the minimizer manifold.

Corollary 3.3 places our criterion next to this body of work, whose characterizations are norms on the parameters of an unfactored classification model. These characterizations say something about the asymptotic direction of the iterates, but not about the gauge symmetry within a factored model, nor about the selection of an interpolant, which is what we consider. We borrow from this body of work only the idea of associating particular norms to particular optimizers.

Finally, Xie et al. (2025) study the effect of rotations of the parameter space on gradient-based optimization. They show that SGD is rotationally invariant but Adam is not, and propose an interpretation of Adam's efficacy as coming from its alignment with the coordinate system. This is related to the recent work of DePavia et al. (2025), who also consider rotations. Theirs are small external rotations of the data and feature space, rather than internal rotations within the latent space as we consider, and they study their effect on Adam's richness bias in classification. Our twin protocol allows us to quantitatively measure the degree to which Adam depends on the latent basis, and our planted ground truth lets us validate our theory's predictions directly. Both of these, along with the spectral-tail phase diagram, have no counterpart in this literature.

Continuous-time arguments and time rescaling. A note is in order regarding the novelty of Theorem 3.4. The change of time therein is nothing but a standard reparametrization of autonomous flows, a device that is used over and over in continuous-time optimization. In particular, rescaling time is usually the way to show that trajectories are accelerated by certain changes, or that convergence rates can be inferred from Lyapunov functions. Here, on the contrary, convergence rates are discarded. What matters is the condition that the clock satisfies, which makes sure that, under a rescaled flow with clock $a(t)$ computed only in terms of gauge-invariant quantities associated with the trajectory, the path itself, as well as its limit, is invariant. This ensures that any statement regarding the implicit bias (defined in terms of the path and its limit) transfers to the rescaled flow.

Connection to equivariant architectures. Lawrence et al. (2022) show that for a family of equivariant linear G-CNN architectures indexed by any finite group G, gradient descent converges to a solution whose Fourier coefficients are low rank under a Schatten norm. There the symmetry is a property of the problem and not of the parameterization, and it is built into the model through an equivariant layer. On the other hand, the group acting on the parameterization in our setting is a redundancy: multiplying both factors in the matrix factorization by the same $\mathrm{O}(k)$ transformation, corresponding to acting on the shared index of $W = UV^\top$, does not affect the model's predictions. More closely related is the work of Ravanbakhsh et al. (2017), who give an if-and-only-if characterization of the groups G of transformations under which a given parameter-sharing layer is equivariant, in terms of G being a group of symmetries of the weights. Similarly, Theorem 3.2 characterizes which memoryless update rules are equivariant under the gauge group of our parameterization. The notion of equivariance is different, though. We are asking whether a particular update commutes with a group of transformations, and the results do not imply each other.

Connection to SGD and symmetry. Ziyin et al. (2024) study continuous symmetries of the parameters and show that the noise introduced by stochasticity systematically biases SGD to move along these symmetries and converge to a common point that is independent of the initialization. In our setting, this would be reflected in the diffusion term. The drift along degenerate directions studied by Ziyin et al. (2024) is the aspect of SGD dynamics that we study empirically but do not characterize, and it is the question we leave open in Appendix A.9. We are concerned with deterministic updates and symmetries and ask if a given update commutes with these, while Ziyin et al. (2024) ask how SGD moves along symmetry orbits.

Transformation invariance in optimizer design. Table 23 places VectorAdam (Ling et al., 2022) and LoRA-RITE (Yen et al., 2025) beside this paper. Ling et al. (2022) study the rotation-equivariant

case for vector-valued geometric parameters, observing that Adam's per-coordinate moments are not rotation-equivariant and proposing VectorAdam, which uses a single scalar per vector, restoring equivariance. Our p=0 endpoint differs by using a single scalar for both factors, yielding an isotropic preconditioner, whereas VectorAdam aggregates second moments over the entries of each vector and can scale different vectors differently. Both methods are nonetheless gauge-compatible, indicating that gauge-compatibility need not forfeit anisotropy. Adam and our p=0 endpoint delimit two ends of our pooling axis, with VectorAdam giving an intermediate, more pooled, point.

Independently, Yen et al. (2025) propose LoRA-RITE, a transformation-invariant preconditioner for LoRA factors. Like us, they observe that failure to be transformation-invariant is a source of error in the optimizer. They focus on speedup, whereas we additionally find that lack of invariance can introduce a bias in the resulting solution (§4, §5). ScaledGD (Tong et al., 2021) preconditions with the Gram matrix of the other factor. We include our ScaledGD-style variant (Appendix E.16) not as a strong baseline, as it is not tuned, but as a control and representative of our class of equivariant optimizers, and as our within-class counterexample (Remark 3.5).

Relation to equivariant deep learning. Zhang (2026) finds that, in group-equivariant networks, Muon's polar step is equivariant to orthogonal transformations when parameters are written in the Schur (multiplicity) basis, whereas AdamW is not, and proposes PolarAdamW, which violates equivariance in order to allow coordinate-wise adaptation. Their observation that coordinate-wise adaptation is most useful when the multiplicity gauge is trivial echoes our boundary analysis (Appendix E.22.1). Lau & Su (2026) consider optimizers which respect permutation and shift symmetries in neural network parameters, providing a framework which encompasses spectral descent, Muon, Scion, and other optimizers including polar-type updates. Shirodkar (2026) considers how to derive equivariant optimizers in a variety of architectural gauges. The paper shows that a version of Adam resists AdamW's over-training collapse and that a Muon variant groks at previously unreachable depth, both consistent with the mechanism we propose here. Closer to our setting, Stupariu & Manolache (2026) compare Muon to Adam in an equivariant setting.

Sophia and GaLore. While not tested here, and outside this line of designs, two popular optimizers have noteworthy relationships with our results. Sophia (Liu et al., 2024) involves two changes with respect to SGD: scaling by an estimate of the diagonal of the Hessian, and clipping. Both are done per coordinate. The former resembles Adam's $1/\sqrt{\hat{v}}$ and thus has the same coordinate-wise character rather than a two-sided one. The latter is an entrywise nonlinearity applied to the preconditioned update, a fixed function, exactly the kind of thing shown by Proposition 3.1 to be inadmissible in the memoryless case. Since it is stateful, however, this does not automatically apply to Sophia (cf. Remark A.4). While using second-order information might seem to make Sophia gauge-invariant, unfortunately, as long as the choice of basis (the coordinates in which the parameters happen to be stored) is arbitrary, a diagonal-only Hessian estimate will be gauge-dependent.

GaLore (Zhao et al., 2024) is different in that it performs a two-sided operation, projecting the gradient onto an (estimated) low-rank subspace of gradients. Neither of our results speaks to this directly, though Theorem 3.2 allows for two-sidedness, and its behavior will depend on details of how often and how the subspace is estimated. Nonetheless, GaLore is a natural object to try in the four-twin setup of §5, which we leave to future work.

Implicit bias from stochastic symmetries. Aladrah et al. (2026) combine continuous symmetries of the loss with SGD's stochasticity to obtain a geometric correction, based on quotient volume, to the effective loss. In particular, they consider the symmetry (UD, VD^{-1}) for positive rescaling of the feature map, which is a subgroup of $\mathrm{GL}(k)$ but not of our orthogonal gauge. Our framework is complementary to their noise-dependent mechanism: it offers a deterministic and optimizer-dependent account of implicit biases. Moreover, equivariance by itself does not decide which gauge-invariant solution is selected. It provides a useful diagnosis of basis-dependence, but is orthogonal to understanding the spectral schedule.

Structured optimizers: Muon and beyond. Recent work has explored the implicit bias of structured optimizers such as Shampoo (Gupta et al., 2018) and Muon (Jordan et al., 2024; Liu et al., 2025), the latter performing orthogonalization by the matrix-sign function applied to the momentum. Fan et al. (2025); Gronich & Vardi (2026) attribute a spectral max-margin bias to Muon, and Kang

et al. (2026) prove uniform growth of singular values under Muon for low-rank adaptation (LoRA) factorizations. Our phase diagram recasts these insights through the lens of Theorem 3.2. It shows that Muon succeeds when the target is exactly low-rank and falls behind GD on problems with a heavy spectral tail, consistent with the dual behavior that Dragutinović et al. (2026) observe. Concurrently, Beneventano et al. (2026) give an alternative justification of Muon's polar step as one-step entropy maximization and analyze Muon's exact singular-value dynamics in underdetermined regression. Shen et al. (2026) report a consistent trade-off in experiments on mixed-spectrum sensing. Finally, Dong & Sawin (2026) propose a within-class family, Muon with a fractional power of the gradient singular values, interpolating between Muon and GD along the spectral-transfer axis of Proposition A.3. Our two-axis organization provides a simple lens through which to understand these results. However, we caution that placing these methods along the axes is an interpretation of their behavior, not a direct implication of Theorem 3.2, especially for stateful optimizers along the schedule axis.

Mini-batches. Our theoretical results apply to the full-batch setting. We do not aim to understand the effects of noise on the gauge-splitting dynamics. Mini-batch noise adds a diffusion term that sits outside our results (Appendix A.9). Empirically, the step-1 gauge split under Adam that we measure in Appendix D.4 is preserved when mini-batches are used.