ANGLE-SCHEDULED MUON: REPLACING IMPLICIT STEP DECAY WITH AN EXPLICIT SCHEDULE

Anonymous authors
Paper under double-blind review

ABSTRACT

Muon orthogonalizes momentum so that the norm of each matrix update is fixed by the learning rate and the matrix shape. However, we observe that weight norms keep growing during training while update norms remain nearly constant, so each update moves the weights by a shrinking fraction even at a constant learning rate. We refer to this as *implicit step decay*: the angular step, i.e., the update norm relative to the weight norm, decays at a rate dictated by weight growth rather than by the learning-rate schedule. Based on this insight, we propose **A**ngle-**S**cheduled **M**uon (**ASM**), which replaces implicit step decay with an explicit schedule. ASM keeps Muon's orthogonalized update direction and sets each update's length to a prescribed fraction of the current weight norm, so that every weight matrix follows the same angular-step schedule. Crucially, ASM is a drop-in modification of Muon that adds only norm computations, requiring no additional orthogonalization or gradient evaluations. Experiments on language modeling and image classification show that ASM consistently improves over Muon and recent Muon variants with limited overhead.

1 INTRODUCTION

Muon updates matrix parameters with orthogonalized momentum and has become a strong alternative to AdamW for training language models (Jordan et al., 2024; Liu et al., 2025a). Because orthogonalization fixes the norm of each update up to a factor determined by the matrix shape, the learning rate alone determines how long each step is.

How much a step changes the network, however, depends not on its length alone but on its length relative to the weights it modifies. For scale-invariant weight matrices, such as a linear map followed directly by normalization, the output is unchanged by positive rescaling, so the same update changes the function less as the weights grow (van Laarhoven, 2017; Li & Arora, 2020; Wan et al., 2021). The relevant quantity is therefore the *angular step*

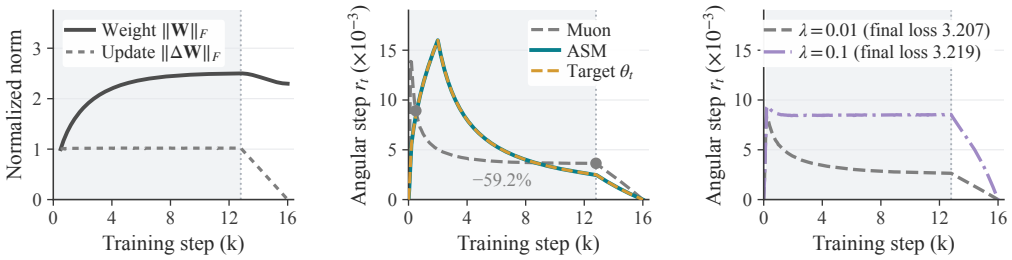
$$r_t = \frac{\|\Delta \mathbf{W}_t\|_F}{\|\mathbf{W}_t\|_F}, \tag{1}$$

which approximates the angle by which a weight matrix $\mathbf{W}_t$ rotates under a small update $\Delta \mathbf{W}_t$ nearly orthogonal to it (Kosson et al., 2024). Since Muon keeps $\|\Delta \mathbf{W}_t\|_F$ fixed for a given learning rate η_t, its angular step is proportional to $\eta_t / \|\mathbf{W}_t\|_F$: it shrinks whenever the weights grow.

Figure 1 shows that this happens in practice. In 127M language-model training, the weight norm keeps growing during the constant-learning-rate phase, while Muon's update norm stays nearly flat (Figure 1(a)). As a result, the angular step falls by 59.2% between steps 500 and 12,800 even before the learning rate begins to decay (Figure 1(b)). We refer to this as *implicit step decay*. In other words, the step schedule that Muon actually follows is not the learning-rate schedule alone: it is the learning-rate schedule multiplied by an extra decay that comes from weight growth, which the practitioner never chooses. Related growth of row and spectral norms under Muon has also been reported by Lion et al. (2026); here we track the Frobenius norm, which is the quantity that determines Muon's angular step.

Controlling this decay directly is not straightforward, for two reasons. Firstly, a single learning-rate schedule cannot generally compensate for every matrix. The weight norm is not known before

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.



(a) Norms (b) Angular step (c) Stronger weight decay

Figure 1: **Weight growth decays Muon's angular step at a constant learning rate** (127M language model). (a) Weight and update norms, normalized at step 500. (b) Angular step of Muon and ASM, with ASM's target θ_t. (c) Muon's angular step under weight decay 0.01 and 0.1; legends give the final validation loss. Shading marks the constant-learning-rate phase, which ends at the dotted line; gray dots mark steps 500 and 12,800.

training and grows at different rates across matrices, whereas the learning rate is a single scalar shared by all of them. Secondly, weight decay affects the angular step only indirectly. Stronger weight decay slows weight growth and keeps the angular step from falling (Figure 1(c)), but it does so by shrinking the weights themselves, which ties the step size to the strength of regularization. The level at which the angular step settles is then determined by the balance between decay and updates rather than chosen. Neither option lets us specify in advance how the angular step should evolve.

To this end, we propose *Angle-Scheduled Muon (ASM)*, which replaces implicit step decay with an explicit schedule. ASM keeps Muon's momentum and orthogonalized update direction and changes only the update length, through two components. Firstly, it realizes a target angular step θ_t exactly for every matrix by setting the update length to $\theta_t \|\mathbf{W}_t\|_F$ from the current weight norm, so that weight growth no longer shrinks the angular step. Secondly, it specifies how θ_t evolves through an explicit schedule that decays over training, as learning-rate schedules do, but whose shape is chosen rather than inherited from weight growth. Figure 1(b) contrasts the two behaviors: Muon's angular step drifts downward during the constant-learning-rate phase, whereas ASM follows its prescribed schedule. ASM adds only norm computations and a scalar rescaling per matrix, with no additional orthogonalization or gradient evaluations. Concurrently, Hübler et al. (2026) show that the Muown parameterization (Lion et al., 2026) implicitly decays its angular step size; ASM instead applies directly to standard Muon as a drop-in change, without reparameterizing the weights.

Our contributions are:

(i) We identify implicit step decay in Muon and decompose the change in the angular step into learning-rate, direction-norm, and weight-growth terms, showing that weight growth alone decays the angular step at a constant learning rate.

(ii) We propose ASM, a drop-in modification of Muon that realizes a prescribed angular-step schedule exactly for every weight matrix using only norm computations.

(iii) We validate ASM on language modeling and image classification, where it consistently improves over Muon and over controls that simply enlarge its steps, with limited computational overhead.

2 Related Work

Muon and its variants. Muon orthogonalizes momentum before updating matrix parameters (Jordan et al., 2024), which corresponds to steepest descent under the spectral norm (Bernstein & Newhouse, 2024); large-scale implementations add matrix-shape scaling, weight decay, and an auxiliary optimizer for non-matrix parameters (Liu et al., 2025a). Later variants refine the update through neuron-wise normalization (Li et al., 2025), element-wise second moments (Si et al., 2025), hybrid preconditioning (Liu et al., 2025b), or reshaping of its singular spectrum (Wu et al., 2026a;b), and numerical work studies faster and inexact orthogonalization (Amsel et al., 2026; Shulgin et al.,

2026). These methods change the update itself, whereas ASM changes only its length relative to the weights and is therefore complementary to them.

Step sizes relative to the weights. LARS and LAMB scale each layer's update by the ratio between parameter and update norms (You et al., 2017; 2020), optimizer grafting combines the direction of one optimizer with the step length of another (Agarwal et al., 2020), and OrScale and TrasMuon bring trust-ratio scaling and energy-based trust-region clipping, respectively, to orthogonalized updates (Lou & You, 2026; Cheng et al., 2026). For scale-invariant weights, the weight norm and weight decay jointly set the effective learning rate (van Laarhoven, 2017; Zhang et al., 2019; Li & Arora, 2020), and their balance drives the angular update to a steady state (Wan et al., 2021; Kosson et al., 2024); Kosson et al. (2024) further control this angular update directly by fixing weight norms and setting the rotation of each neuron to its equilibrium value, and Hyperball fixes the Frobenius norms of weights and their updates, which also fixes the angular update (Wen et al., 2026). ASM shares this norm-relative, angular view, but applies it to Muon's matrix updates and prescribes the angular-step schedule directly: the target does not depend on weight decay, does not require fixing the weight norm, and requires no per-layer calibration. Most closely related, Muown separates the magnitudes of weight rows from their directions (Lion et al., 2026), and concurrent work shows that this parameterization implicitly decays its angular step and makes the step an explicit, schedulable multiplier (Hübler et al., 2026). ASM addresses the same phenomenon in standard Muon, without reparameterizing the weights.

Learning-rate schedules. Cosine annealing (Loshchilov & Hutter, 2017) and warmup-stable-decay schedules (Hu et al., 2024) set the learning rate over training, and recent analyses separate the roles of the sustained phase and the final decay (Wen et al., 2025). These schedules control the absolute size of each update. ASM keeps the recipe's schedule shape but applies it to the angular step, so that the schedule the practitioner chooses is the one the weights actually follow.

3 Preliminaries

Muon. We consider trainable weight matrices $\mathbf{W}_{i,t} \in \mathbb{R}^{m_i \times n_i}$, where i indexes matrices and t indexes training steps. Muon forms a momentum input $\mathbf{H}_{i,t}$ from stochastic gradients and orthogonalizes it with a fixed number of Newton–Schulz iterations, $\mathbf{Y}_{i,t} = \mathcal{N}(\mathbf{H}_{i,t})$ (Jordan et al., 2024). Its update is

$$\Delta \mathbf{W}_{i,t} = -c_i \eta_t \mathbf{Y}_{i,t}, \tag{2}$$

where η_t is the base learning rate and c_i is a fixed factor determined by the matrix shape, typically chosen so that the update magnitude matches that of AdamW (Liu et al., 2025a). With decoupled weight decay λ, the weights evolve as $\mathbf{W}_{i,t+1} = (1 - \eta_t \lambda)\mathbf{W}_{i,t} + \Delta \mathbf{W}_{i,t}$ (Loshchilov & Hutter, 2019). Because orthogonalization drives the singular values of $\mathbf{Y}_{i,t}$ toward one, $\|\mathbf{Y}_{i,t}\|_F$ is nearly constant for a given matrix shape.

Angular step. We measure the size of an update relative to the weights it modifies by the angular step of each matrix,

$$r_{i,t} = \frac{\|\Delta \mathbf{W}_{i,t}\|_F}{\|\mathbf{W}_{i,t}\|_F}, \tag{3}$$

defined for nonzero weights. Here $\Delta \mathbf{W}_{i,t}$ excludes weight decay, which we treat separately. For a small update nearly orthogonal to the weights, $r_{i,t}$ approximates the angle by which $\mathbf{W}_{i,t}$ rotates; Appendix B.3 gives the exact relation for general update directions. All norms are Frobenius norms.

4 Angle-Scheduled Muon

Section 1 showed that Muon's angular step decays as the weights grow, and that neither the learning-rate schedule nor weight decay controls this decay directly. As shown in Figure 2, ASM keeps Muon's momentum and orthogonalized direction and replaces only the update length. We first formalize the sources of implicit step decay (Section 4.1), then describe how ASM realizes a target angular step for every matrix (Section 4.2) and how the target is scheduled over training (Section 4.3).

4.1 Implicit step decay in Muon

Substituting Eq. (2) into Eq. (3) gives Muon's angular step $r_{i,t} = c_i \eta_t \|\mathbf{Y}_{i,t}\|_F / \|\mathbf{W}_{i,t}\|_F$. For positive learning rates and nonzero weights and directions at consecutive steps, its change decomposes as

$$\log \frac{r_{i,t+1}}{r_{i,t}} = \underbrace{\log \frac{\eta_{t+1}}{\eta_t}}_{\text{learning-rate schedule}} + \underbrace{\log \frac{\|\mathbf{Y}_{i,t+1}\|_F}{\|\mathbf{Y}_{i,t}\|_F}}_{\text{direction norm}} - \underbrace{\log \frac{\|\mathbf{W}_{i,t+1}\|_F}{\|\mathbf{W}_{i,t}\|_F}}_{\text{weight growth}}. \tag{4}$$

The shape factor c_i cancels. Of the three terms, only the first is prescribed by the practitioner. The second stays close to zero because orthogonalization keeps $\|\mathbf{Y}_{i,t}\|_F$ nearly constant. The third is negative whenever the weights grow, which happens throughout the constant-learning-rate phase (Figure 1(a)). Weight growth therefore decays the angular step even when the first term is zero, and once learning-rate decay begins, the two terms compound. We refer to the contribution of the third term as *implicit step decay*.

Weight decay acts only indirectly. Weight decay counteracts weight growth (van Laarhoven, 2017; Wan et al., 2021; Kosson et al., 2024). Dropping the matrix index, let $\alpha_t = 1 - \eta_t \lambda$, and let $\rho_t = \langle \mathbf{W}_t, \mathbf{Y}_t \rangle_F / (\|\mathbf{W}_t\|_F \|\mathbf{Y}_t\|_F)$ be the cosine between the weights and the orthogonalized direction. Expanding $\mathbf{W}_{t+1} = \alpha_t \mathbf{W}_t - c\,\eta_t \mathbf{Y}_t$ gives

$$\frac{\|\mathbf{W}_{t+1}\|_F^2}{\|\mathbf{W}_t\|_F^2} = \alpha_t^2 - 2\alpha_t r_t \rho_t + r_t^2. \tag{5}$$

The growth of the weights thus depends jointly on the decay factor, the angular step itself, and the alignment between update and weights. A larger λ shrinks the weights and can keep the angular step from falling (Figure 1(c)), but the level at which it settles emerges from this balance rather than being specified, and it cannot follow a prescribed trajectory. ASM instead specifies the angular step directly and leaves the treatment of the weight norm as a separate choice (Section 4.4).

Why Muon? Norm-dependent angular dynamics also arise in AdamW (Kosson et al., 2024), but there the update norm itself changes with the first- and second-moment estimates (Kingma & Ba, 2015). In Muon, orthogonalization keeps the direction norm nearly fixed, so the second term of Eq. (4) almost vanishes and weight growth becomes the dominant uncontrolled change in the angular step. This makes implicit step decay particularly clean to identify, and to correct, in Muon.

4.2 Realizing a target angular step

Let $\theta_t \geq 0$ be the target angular step at step t, shared by all matrices that Muon updates. ASM computes the same momentum input $\mathbf{H}_{i,t}$ and orthogonalized direction $\mathbf{Y}_{i,t}$ as Muon, and sets the length of the update to $\theta_t \|\mathbf{W}_{i,t}\|_F$:

$$\Delta \mathbf{W}_{i,t}^{\text{ASM}} = -\theta_t \frac{\|\mathbf{W}_{i,t}\|_F}{\|\mathbf{Y}_{i,t}\|_F} \mathbf{Y}_{i,t}. \tag{6}$$

Compared with Eq. (2), ASM replaces Muon's fixed coefficient $c_i \eta_t$ with $\theta_t \|\mathbf{W}_{i,t}\|_F / \|\mathbf{Y}_{i,t}\|_F$, recomputed at every step from the current norms. For nonzero weights and directions, taking norms gives

$$\frac{\|\Delta \mathbf{W}_{i,t}^{\text{ASM}}\|_F}{\|\mathbf{W}_{i,t}\|_F} = \theta_t. \tag{7}$$

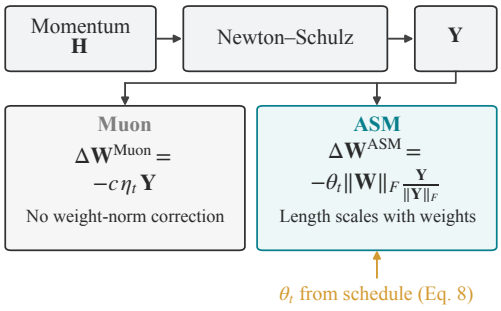


Figure 2: **ASM changes only the update length.** It keeps Muon's orthogonalized direction and sets the update length to $\theta_t \|\mathbf{W}\|_F$. Matrix and time indices are omitted.

The weight norm cancels: weight growth changes the absolute length of the update but no longer its angular step, so the third term of Eq. (4) is removed by construction, and every matrix follows the same target regardless of its scale. Because Eq. (7) uses the measured norm of the finite-step Newton–Schulz output, it holds without exact orthogonalization; approximation error affects only the direction (Shulgin et al., 2026). If either $\mathbf{W}_{i,t}$ or $\mathbf{Y}_{i,t}$ is zero, ASM applies no update.

Why not a learning-rate schedule? A learning-rate multiplier $k_t > 0$ shared by all matrices changes every Muon angular step to $k_t r_{i,t}$. It reproduces ASM only if $r_{i,t} = \theta_t / k_t$ for every i, that is, only if all matrices already share the same angular step. When weight norms grow at different rates, their angular steps differ, and no single multiplier can bring all of them to the target at once. ASM therefore determines both how update lengths are allocated across matrices and how they evolve over time, which a scalar learning-rate schedule cannot do.

4.3 Specifying the schedule

With the angular step realized exactly, the remaining choice is how θ_t evolves. We parameterize it as

$$\theta_t = \theta_{\text{peak}} \frac{q(t)}{q(t_p)} \begin{cases} (t/t_p)^a, & t < t_p, \\ (t/t_p)^{-b}, & t \geq t_p, \end{cases} \tag{8}$$

where $q(t)$ is the learning-rate shape of the training recipe, such as cosine annealing or warmup-stable-decay (Loshchilov & Hutter, 2017; Hu et al., 2024), t_p is a transition step, θ_{peak} is the target at t_p, and $a, b \geq 0$ are exponents. The factor $q(t)$ keeps the recipe's warmup and final annealing, the factor $(t/t_p)^a$ controls the branch before t_p, and $(t/t_p)^{-b}$ controls its later decay. Unlike implicit step decay, this decay is chosen in advance and is identical for all matrices. Setting $b = 0$ removes this decay.

Algorithm 1 One ASM update for a matrix

Input: step t, $\mathbf{W}_{i,t}$, gradient, momentum state
$\theta_t \leftarrow$ Eq. (8)
Update Muon momentum; form $\mathbf{H}_{i,t}$
$\mathbf{Y}_{i,t} \leftarrow \mathcal{N}(\mathbf{H}_{i,t})$
if $\mathbf{W}_{i,t} = \mathbf{0}$ or $\mathbf{Y}_{i,t} = \mathbf{0}$: $\Delta\mathbf{W} \leftarrow \mathbf{0}$
else: $\Delta\mathbf{W} \leftarrow -\theta_t \|\mathbf{W}_{i,t}\|_F \, \mathbf{Y}_{i,t} / \|\mathbf{Y}_{i,t}\|_F$
Apply $\Delta\mathbf{W}$ to $\mathbf{W}_{i,t}$
Output: $\mathbf{W}_{i,t+1}$ and the updated momentum

4.4 Implementation and cost

Algorithm 1 summarizes one ASM update. All matrices updated by Muon share θ_t, while the remaining parameters, such as embeddings and normalization layers, keep their AdamW updates. The update in Eq. (6) is computed from the weights before the step and is then combined with the recipe's treatment of the weight norm: decoupled weight decay at 127M and 300M, and norm preservation in the 170M and vision recipes (Appendix A.2). ASM is applied from the first update. Convolutional kernels are reshaped into matrices, which leaves their Frobenius norms unchanged.

Additional cost. Beyond Muon's orthogonalization, ASM computes two Frobenius norms and one scalar rescaling per matrix, adding $\mathcal{O}(\sum_i m_i n_i)$ arithmetic per step and no additional orthogonalization or gradient evaluations.

5 Experiments

We study three questions: whether ASM improves training across workloads, which parts of its update rule contribute to the improvement, and whether the gain offsets its computational cost.

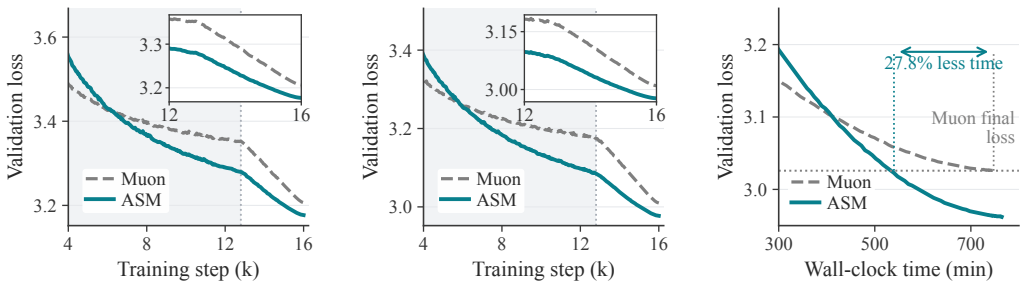
5.1 Experimental setup

Workloads and baselines. We train 127M and 300M GPT-style language models on FineWeb (Penedo et al., 2024) with a warmup-stable-decay schedule, and a 170M Llama-style model with a longer cosine-annealed recipe adapted from Liu et al. (2025a). For image classification, we train ResNet-50 (He et al., 2016) and ViT-S (Dosovitskiy et al., 2021) on CIFAR-100 (Krizhevsky, 2009). Baselines include AdamW (Loshchilov & Hutter, 2019), Muon, DynMuon (Wu et al., 2026a), NorMuon (Li et al., 2025), AdaMuon (Si et al., 2025), COSMOS (Liu et al., 2025b), SAMuon-lite (Wu et al., 2026b), and the official AngularMuown implementation (Hübler et al., 2026). Appendix A gives the architectures, training budgets, and hyperparameters.

ASM configurations and controls. Table 1 uses the selected ASM configuration of each workload, and every ASM run applies angular control from the first update. The mechanism studies in Section 5.3 instead hold one reference configuration fixed across their arms, so that each comparison changes a single factor; Appendix A.2 lists the configuration of each workload. The controls

Table 1: Results on language modeling and image classification: final validation loss (NLL) on FineWeb, and test top-1 accuracy (%) and cross-entropy (CE) on CIFAR-100, as mean and standard deviation over three seeds. Bold: best; underline: second best. Our method is shaded.

	FineWeb		ResNet-50		ViT-S	
Method	127M ↓	300M ↓	Top-1 ↑	CE ↓	Top-1 ↑	CE ↓
AdamW	3.2501 ±0.006	3.0646 ±0.005	77.65 ±0.28	0.913 ±0.011	66.95 ±0.62	1.403 ±0.021
Muon	3.2017 ±0.004	3.0086 ±0.004	77.72 ±0.31	0.939 ±0.009	71.30 ±0.55	1.216 ±0.016
AdaMuon	3.2590 ±0.007	3.0924 ±0.006	77.20 ±0.24	0.964 ±0.013	70.05 ±0.71	1.299 ±0.024
NorMuon	3.1998 ±0.004	3.0051 ±0.003	77.62 ±0.35	0.952 ±0.008	71.03 ±0.48	1.240 ±0.014
COSMOS	3.2500 ±0.005	3.0552 ±0.004	78.14 ±0.22	0.937 ±0.010	50.71 ±0.83	1.872 ±0.028
SAMuon-lite	3.2076 ±0.005	3.0147 ±0.004	79.43 ±0.30	0.866 ±0.012	69.16 ±0.59	1.333 ±0.019
DynMuon	3.1927 ±0.004	2.9902 ±0.003	78.84 ±0.26	0.903 ±0.009	70.23 ±0.66	1.287 ±0.022
AngularMuown	3.1896 ±0.005	2.9916 ±0.003	80.57 ±0.33	0.819 ±0.014	71.97 ±0.52	1.199 ±0.015
ASM (ours)	**3.1728** ±0.004	**2.8725** ±0.003	**81.26** ±0.27	**0.789** ±0.010	**73.03** ±0.57	**1.155** ±0.017



(a) 127M (b) 300M (c) 170M: elapsed time

Figure 3: **Validation loss over training.** (a,b) The last 75% of training at 127M and 300M, with insets for the final 25%; shading marks the constant-learning-rate phase, which ends at the dotted line. (c) Validation loss against wall-clock time on the 170M recipe; the arrow marks the time ASM saves in reaching the end point of Muon's curve (horizontal line).

in Table 3 separate amplitude, matrix allocation, and temporal shape. Additional controls that keep dynamic magnitude rules are reported in Appendix A.4.

Evaluation. For language models, we report the final validation loss after training, the negative log-likelihood (NLL) on held-out tokens. Vision configurations and checkpoints are selected by validation accuracy, with cross-entropy breaking ties, and test metrics are read out afterward. Tuning budgets vary across methods, a relevant factor in optimizer comparisons (Schmidt et al., 2021). All results are averaged over three seeds, and Appendix A specifies the hyperparameters and the computing platform.

5.2 Main Results

Language modeling and image classification. As shown in Table 1, ASM lowers validation loss from 3.2017 to 3.1728 at 127M and from 3.0086 to 2.8725 at 300M. ASM also improves on DynMuon, the second-best method at 300M, by 0.020 at 127M and 0.118 at 300M. The gain over Muon is larger at 300M (0.136) than at 127M (0.029), and every gap above exceeds twice the largest standard deviation across seeds in Table 1 (0.007). On CIFAR-100, ASM achieves the best result on every metric, reaching 81.26% top-1 accuracy with ResNet-50 and 73.03% with ViT-S, 3.54 and 1.73 percentage points above Muon, and it also attains the lowest test cross-entropy on both architectures (0.789 and 1.155).

A longer cosine-annealed recipe. The 170M recipe tests transfer to a different tokenizer, a cosine schedule, and a longer training horizon. ASM reaches a final validation loss of 2.9988, compared

with 3.0243 for AngularMuown and 3.0622 for Muon (Table 2; Table 10 in the appendix adds further controls). Absolute losses are compared within each recipe.

Comparison with AngularMuown. AngularMuown also makes the angular step explicit, but it reparameterizes each weight matrix into row magnitudes and normalized directions and optimizes them separately (Hübler et al., 2026). ASM obtains an explicit angular step for standard Muon by rescaling the update alone. ASM is ahead on every workload: by 0.017 and 0.119 in validation loss at 127M and 300M, by 0.69 and 1.06 points in top-1 accuracy on the two vision architectures, and by 0.026 on the 170M recipe. Making the angular step explicit therefore does not require changing the parameterization.

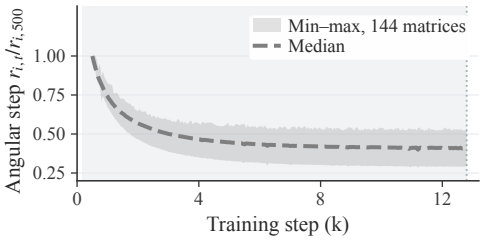
Table 2: Results on the 170M cosine-annealed recipe. We report the final validation loss. Our method is shaded.

Method	Loss
Muon	3.0622
DynMuon	3.0620
AngularMuown	3.0243
ASM (ours)	**2.9988**

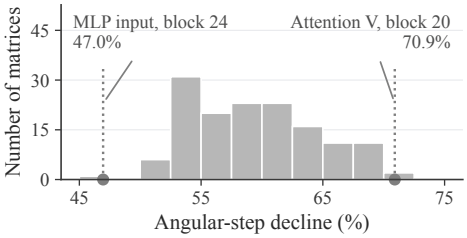
How the advantage develops. As shown in Figure 3(a,b), ASM finishes below Muon at both scales despite not leading throughout training. In Figure 1(b), its angular step rises above Muon's early in training, then falls below it before the plateau ends. ASM thus changes when updates are large. The curves track validation loss during training, whereas Table 1 reports the evaluation after training.

5.3 What drives the improvement

Separating the interventions. We compare a 2×2 set of 127M controls that share the same weight decay, auxiliary optimizer, and schedule hyperparameters (Appendix A.4). One factor is the target's temporal shape: the ASM profile or the outer learning-rate shape alone. The other is allocation: a target enforced separately for every matrix, or one global multiplier that preserves Muon's allocation. The two shapes have the same sum of squared target angular steps over training. This controls the prescribed budget, while allowing weight norms and absolute update lengths to evolve differently.



(a) Trajectories across matrices



(b) Distribution of declines

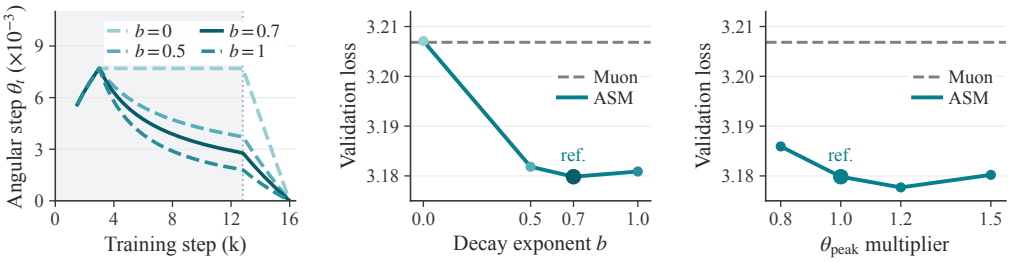
Figure 4: **Matrices decay at different rates under Muon.** (a) Normalized angular steps of all 144 matrices at 127M: min–max band and median. (b) Declines from step 500 to 12,800; the two extremes are labeled.

Both interventions matter. As shown in Table 3, per-matrix control lowers validation loss under either temporal shape, and the ASM shape lowers it under either allocation. The two effects are nearly additive: alone, they lower the loss by 0.024 and 0.028, and together by 0.048. Intuitively, the temporal shape decides when updates are large, whereas per-matrix control decides which matrices receive them. Their combination reaches 3.1728, compared with 3.1976 for ordinary Muon and 3.1954 for Muon with its updates amplified by a fixed factor of 1.5, so the gain is not explained by larger steps. Appendix A.4 gives their exact construction. The benefit of per-matrix control is consistent with how unevenly Muon's angular step decays: across the 144 matrices of the 127M model, it falls by 47.0% to 70.9% over the constant-learning-rate phase (Figure 4), so a single global multiplier cannot keep every matrix on target.

The useful change is how the step decays. Using a fixed reference configuration (Appendix A.5), Figure 5(a,b) compares decay exponents. Removing the power-law decay ($b = 0$) raises validation loss from 3.1798 to 3.2071. The target is then flat after t_p during the plateau, while keeping fi-

Table 3: Effect of temporal shape and matrix allocation on 127M (final validation loss). All cells share the same squared angular-step budget; reference rows use Muon without angular control.

Target shape	Global multiplier	Per-matrix control
Learning-rate shape only	3.2207	3.1964
ASM shape	3.1928	**3.1728**
References		
Ordinary Muon	3.1976	
Ordinary Muon $\times 1.5$	3.1954	



(a) Target trajectories (b) Decay exponent (c) Schedule amplitude

Figure 5: **Sensitivity to the schedule at 127M.** (a) Target trajectories for decay exponents b. (b,c) Final validation loss when varying b or the peak amplitude alone; dashed lines mark Muon (3.2068).

nal learning-rate annealing. Positive exponents finish within 0.0020 of one another. Figure 5(c) also shows that increasing the amplitude does not improve loss monotonically. These observations support choosing the decay profile, rather than eliminating decay or uniformly enlarging the steps.

Schedule and weight-decay choices. With the same configuration, removing the rising branch gives a validation loss of 3.1796, close to the reference 3.1798, so this experiment does not establish a benefit from that branch. Weight-decay coefficients 0, 0.01, and 0.1 give 3.1806, 3.1798, and 3.5613. Direct angular-step control thus separates the target from weight growth, but does not remove the need to choose suitable regularization. Conversely, for Muon itself, raising weight decay from 0.01 to 0.1 keeps the angular step from falling (Figure 1(c)) but raises the loss from 3.2068 to 3.2192, so weight decay cannot replace an explicit schedule. Appendix A.5 reports these comparisons.

5.4 Practical considerations

The gain offsets a small per-step cost. Table 5 reports the end-to-end training cost on the same hardware. ASM adds 2.46% median step time at 127M and 0.88% at 300M, with unchanged peak memory at the reported precision. The angular control adds no state, since its target depends only on the step and the weight norm. The extra work consists of two Frobenius norms and one rescaling per matrix, whose cost does not grow with the batch size; the relative overhead is therefore smaller at 300M, where each update does more computation. As shown in Figure 3(c), the 170M ASM recipe first reaches the end point of Muon's validation curve in 539.9 minutes, compared with 747.4 minutes for Muon, a 27.8% reduction. The target is chosen from Muon's curve after training, and elapsed time includes evaluation. This saving comes from reaching the target in fewer updates; full-

Table 4: Time to reach validation-loss targets taken from Muon's trajectory on the 170M recipe.

Target loss	Muon (min)	ASM (min)	Time saved
3.1167	374.7	389.4	−3.9%
3.0457	594.6	502.3	+15.5%
3.0258	747.4	539.9	+27.8%

Table 5: Training cost on the same hardware: median time per update, change relative to Muon, and peak memory. Our method is shaded.

	127M			300M		
Method	ms/update	vs. Muon	GiB	ms/update	vs. Muon	GiB
AdamW	1,203.7	−0.90%	13.16	2,693.1	−1.16%	23.78
Muon	1,214.5	–	12.88	2,724.9	–	22.94
ASM	1,244.4	+2.46%	12.88	2,748.8	+0.88%	22.94

budget training takes 764.5 minutes for ASM and 747.8 minutes for Muon. Table 4 reports all three targets; the saving grows from -3.9% at the highest target to 27.8% at the end point of Muon's curve, because ASM trails Muon early in training and overtakes it later (Figure 3(c)).

Robustness to the nominal learning rate. ASM prescribes the angular step directly, so the nominal learning rate no longer sets the update length; it affects only decoupled weight decay. Figure 6 varies the nominal learning rate over 0.005, 0.01, and 0.02 at the 127M reference configuration (Appendix A.5). The final validation loss of ASM changes by only 0.0036 across these rates, less than a fifth of Muon's range of 0.0200 and below DynMuon's 0.0061. ASM thus makes training much less sensitive to the nominal learning rate. In practice, the nominal learning rate can be kept from the base Muon recipe, and tuning can focus on the schedule, which does set the update length.

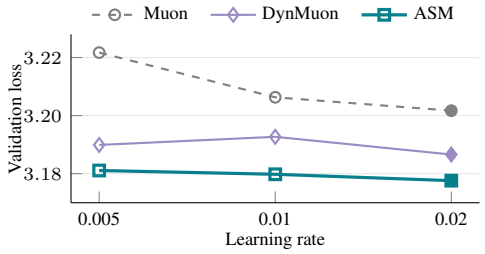


Figure 6: **ASM is robust to the nominal learning rate.** Final validation loss at three nominal learning rates; filled markers indicate each method's best rate.

Choosing the schedule. The selected language-model schedules change little with scale: at 127M and 300M they share $\theta_{\text{peak}} = 0.016$, $t_p = 2{,}000$, and $a = 0.46$, and differ only in b (1.0 versus 1.2; Table 6). Given the flat amplitude response in Figure 5(c), a new workload can start from these values and tune mainly b. Across all five workloads in Table 6, the rising exponent stays at $a = 0.46$, so in practice only θ_{peak}, t_p, and b are set per workload, while the learning-rate shape $q(t)$ is inherited from the recipe.

Summary. Across language modeling and image classification, ASM improves over Muon on every workload and is the best method on every metric in Table 1. The controlled comparisons attribute this gain to both the temporal shape of the schedule and per-matrix control, rather than to larger steps. On the 170M recipe, ASM also reaches the end point of Muon's validation curve in 27.8% less wall-clock time. These benefits come at a small per-step cost and with reduced sensitivity to the nominal learning rate.

6 Conclusion

We showed that Muon's angular step decays implicitly as the weights grow, and proposed ASM, which replaces this implicit decay with an explicit schedule by setting each update's length from the current weight norm. ASM is a drop-in change to Muon that requires only norm computations. It improves over Muon across language modeling and image classification, and controlled comparisons show that both the temporal shape of the schedule and per-matrix control contribute to the gain.

Limitations. The evaluation covers language models up to 300M parameters and CIFAR-100. Search budgets differ across methods. Extending the comparisons to larger models and broader vision tasks is a natural next step. The analysis describes the update geometry and sources of implicit decay; convergence guarantees for the full momentum-based training process remain open.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and refinement of the manuscript. We also used them to assist with figure preparation, literature searches, and the inspection and interpretation of experimental outputs. We have not used generative AI tools for conducting experiments, generating training data or experimental results, originating the core research idea, or making final scientific conclusions. We have reviewed and verified all AI-assisted work, including manually checking the generated text, figures, and cited references for correctness and consistency. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We make efforts to ensure the reproducibility of our work by providing detailed descriptions of the proposed method and experimental protocols. The main paper specifies the ASM update in Equation (6), the angular schedule in Equation (8), and a single ASM update in Algorithm 1. Detailed training recipes, hyperparameters, schedule parameters, evaluation protocol, and the timing protocol for every workload are provided in Appendix A, together with the complete training procedure in Algorithm 2. Appendix B derives the update identities. The source code will be released upon acceptance to facilitate further research and reproduction of our results.

ETHICS STATEMENT

This work proposes a modification to an optimizer for training neural networks and does not involve human subjects, personal data or new datasets. All experiments use publicly available datasets under their respective licenses. We do not foresee ethical concerns specific to this work beyond those that apply to training large models in general.

REFERENCES

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020. URL https://arxiv.org/abs/2002.11803.

Noah Amsel, David Persson, Christopher Musco, and Robert M. Gower. The polar express: Optimal matrix sign methods and their application to the Muon algorithm. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2505.16932.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024. URL https://arxiv.org/abs/2409.20325v2.

Peng Cheng, Jiucheng Zang, Qingnan Li, Liheng Ma, Yufei Cui, Yingxue Zhang, Boxing Chen, Ming Jian, and Wen Tong. TrasMuon: Trust-region adaptive scaling for orthogonalized momentum optimizers. *arXiv preprint arXiv:2602.13498*, 2026. URL https://arxiv.org/abs/2602.13498v2. Version 2.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016. URL https://arxiv.org/abs/1512.03385.

Shengding Hu, Yuge Tu, Xu Han, et al. MiniCPM: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024. URL https://arxiv.org/abs/2404.06395.

Florian Hübler, Kai Lion, Antonio Orvieto, and Niao He. Muown implicitly performs angular step-size decay. *arXiv preprint arXiv:2606.23637*, 2026. URL https://arxiv.org/abs/2606.23637.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Atli Kosson, Bettina Messmer, and Martin Jaggi. Rotational equilibrium: How weight decay balances learning across neural networks. In *International Conference on Machine Learning*, 2024.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://cave.cs.toronto.edu/kriz/cifar.html.

Zhiyuan Li and Sanjeev Arora. An exponential learning rate schedule for deep learning. In *International Conference on Learning Representations*, 2020.

Zichong Li, Liming Liu, Chen Liang, Weizhu Chen, and Tuo Zhao. NorMuon: Making Muon more efficient and scalable. *arXiv preprint arXiv:2510.05491*, 2025. URL https://arxiv.org/abs/2510.05491.

Kai Lion, Florian Hübler, Bingcong Li, Antonio Orvieto, and Niao He. Muown: Row-norm control for Muon optimization. *arXiv preprint arXiv:2605.10797*, 2026. URL https://arxiv.org/abs/2605.10797.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025a. URL https://arxiv.org/abs/2502.16982.

Liming Liu, Zhenghao Xu, Zixuan Zhang, Hao Kang, Zichong Li, Chen Liang, Weizhu Chen, and Tuo Zhao. COSMOS: A hybrid adaptive optimizer for memory-efficient training of LLMs. *arXiv preprint arXiv:2502.17410*, 2025b. URL https://arxiv.org/abs/2502.17410.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

Yuxuan Lou and Yang You. OrScale: Orthogonalised optimization with layer-wise trust-ratio scaling. *arXiv preprint arXiv:2605.07815*, 2026. URL https://arxiv.org/abs/2605.07815.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

Robin M. Schmidt, Frank Schneider, and Philipp Hennig. Descending through a crowded valley – benchmarking deep learning optimizers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9367–9376, 2021. URL https://proceedings.mlr.press/v139/schmidt21a.html.

Egor Shulgin, Sultan AlRashed, Peter Richtárik, and Francesco Orabona. Beyond the ideal: Analyzing the inexact Muon update. In *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300 of *Proceedings of Machine Learning Research*, pp. 5167–5175, 2026. URL https://proceedings.mlr.press/v300/shulgin26a.html.

Chongjie Si, Debing Zhang, and Wei Shen. AdaMuon: Adaptive Muon optimizer. *arXiv preprint arXiv:2507.11005*, 2025. URL https://arxiv.org/abs/2507.11005.

Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017. URL https://arxiv.org/abs/1706.05350.

Ruosi Wan, Zhanxing Zhu, Xiangyu Zhang, and Jian Sun. Spherical motion dynamics: Learning dynamics of normalized neural network using SGD and weight decay. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/326a8c055c0d04f5b06544665d8bb3ea-Abstract.html.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.05192.

Kaiyue Wen, Xingyu Dang, Kaifeng Lyu, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them II: Hyperball optimization. *arXiv preprint arXiv:2606.16899*, 2026. URL https://arxiv.org/abs/2606.16899.

Fangzhou Wu, Rikhav Shah, Sandeep Silwal, and Qiuyi Zhang. DynMuon: A dynamic spectral shaping view of Muon. *arXiv preprint arXiv:2605.17109*, 2026a. URL https://arxiv.org/abs/2605.17109v3.

Xiaodong Wu, Wenyi Yu, Chao Zhang, and Philip Woodland. Spectral allocation: Why Muon outperforms Adam, and how to improve Muon. *arXiv preprint arXiv:2608.25990*, 2026b. URL https://arxiv.org/abs/2608.25990.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017. URL https://arxiv.org/abs/1708.03888v3.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

Guodong Zhang, Chaoqi Wang, Bowen Xu, and Roger Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations*, 2019.

A Experimental Protocol and Hyperparameters

A.1 Evaluation

For language models, all reported losses are the final validation NLL, evaluated on held-out tokens after training, and losses are compared only within the same recipe. Vision uses test accuracy and cross-entropy at the checkpoint selected by validation accuracy, with validation cross-entropy breaking ties.

ASM hyperparameters are selected by validation performance, and test metrics are never used for selection.

Computing platform. All experiments run on a single platform with GPUs of 96 GB memory and a shared software environment. Timing measurements use the 127M and 300M training settings and report the median over 250 timed updates after 50 warmup updates, with angular control active.

A.2 Schedule and norm settings

The schedule is exactly Eq. (8), with θ_{peak} denoting its value at t_p. This notation absorbs any separate amplitude multiplier. When $q(t)$ varies around t_p, t_p need not be the maximum of the combined profile. The target uses the absolute training step and is applied from the first update.

Table 6: Hyperparameters of ASM. The upper rows are used for the main results in Table 1 and the 170M comparison; the last row is the reference configuration of the 127M sensitivity studies. θ_{peak} is the target angular step at the transition step t_p in Eq. (8).

Workload	Main LR	t_p	θ_{peak}	a	b	Preserve norm
127M	0.02	2,000	0.016	0.46	1.0	No
300M	0.02	2,000	0.016	0.46	1.2	No
170M	0.00095	9,537	0.0077	0.46	0.7	Yes
ResNet-50	0.001	3,262	0.0154	0.46	0.9	Yes
ViT-S	0.001	3,262	0.01155	0.46	0.7	Yes
127M sensitivity	0.01	3,000	0.0077	0.46	0.7	No

The 127M and 300M runs use decoupled weight decay $\lambda = 0.01$ and no norm preservation. Their update is

$$\mathbf{W}_{i,t+1} = (1 - \eta_t \lambda)\mathbf{W}_{i,t} + \Delta\mathbf{W}_{i,t}^{\text{ASM}}. \tag{9}$$

The 170M and vision configurations instead omit weight decay on controlled matrices and preserve their pre-update norms:

$$\mathbf{Z}_{i,t} = \mathbf{W}_{i,t} + \Delta\mathbf{W}_{i,t}^{\text{ASM}}, \qquad \mathbf{W}_{i,t+1} = \|\mathbf{W}_{i,t}\|_F \frac{\mathbf{Z}_{i,t}}{\|\mathbf{Z}_{i,t}\|_F}. \tag{10}$$

A zero intermediate matrix is left unchanged. In both cases, the prescribed angular step concerns the optimization update before weight decay or norm preservation.

A.3 Training recipes

The 127M transformer has 24 blocks, eight attention heads, and hidden width 512; the 300M transformer has 32 blocks, 12 heads, and width 768. Both use vocabulary size 50,304, sequence length 1,024, and a global batch of 512 sequences, so the 16,000 updates cover 8.39B tokens. Each block contributes six controlled matrices. Token embeddings and output weights use AdamW with base learning rate 0.001 and zero weight decay. The learning-rate factor follows a warmup-stable-decay shape (Hu et al., 2024): it has 160 warmup steps, stays at one until step 12,800, and then decreases linearly to zero at step 16,000. The same factor supplies $q(t)$ in Eq. (8). The training loader uses 99 data shards, with a separate validation shard.

The 170M recipe derives from our Llama-style benchmark, whose starting configuration follows the dense-model scaling experiments of Liu et al. (2025a). Their Table 2 specifies a 12-layer, 12-head

model of width 1,536, with learning rate 9.503×10^{-4} and batch size 96. Our benchmark uses the rounded base rate 9.5×10^{-4} and 96 sequences per update, then reduces the model depth and width for the 170M experiment. We use 2,048-token rather than 8K-token sequences and a 10B-token rather than 8.92B-token budget. These choices change both the model scale and the number of tokens per update. It has 10 blocks, six attention heads, hidden width 768, and SwiGLU width 2,048, with tied token embeddings and output weights. Its 169,295,616 parameters include the 128,256-token embedding. The shared training protocol uses the Llama 3.1 tokenizer, sequence length 2,048, global batch size 96, and 50,863 updates. Its learning-rate factor uses 5% linear warmup followed by cosine decay (Loshchilov & Hutter, 2017) to 0.1 of the maximum. Gradient clipping uses threshold 1. The vision recipe uses 17,400 updates over 200 epochs, with 870 warmup updates followed by cosine decay to 0.1. These recipe-specific factors also supply $q(t)$; the target is normalized by $q(t_p)$ in every case.

The CIFAR-100 ResNet-50 replaces the original stem with a 3×3 convolution of stride one and removes the initial max-pooling layer. ViT-S uses 4×4 patches, hidden width 384, 12 blocks, six attention heads, and an MLP expansion ratio of four on 32×32 images. These architecture adaptations are shared by the corresponding optimizer comparisons.

The schedule target depends on training progress and the selected hyperparameters, rather than a stored norm trajectory.

A.4 Magnitude and allocation controls

Dynamic controls. DynScalar $\times 1.5$ and $\times 1.0$ preserve the scalar magnitude obtained from the momentum norm and spectral exponent; the multiplier acts on the negative-exponent branch. A unit multiplier therefore does not recover ordinary Muon. DynRealloc $\times 1.5$ and $\times 1.0$ additionally use row-equilibrated directions, split-K8 norm calibration, and bounded positive gradient-projection weights. RowEq-dynamic $\times 1.0$ uses uniform allocation in the same row-equilibrated construction. These controls change more than a constant Muon learning-rate multiplier. Table 7 reports their results under the evaluation protocol of Table 1.

Table 7: Results of the additional dynamic controls, with the same metrics as Table 1.

	FineWeb		ResNet-50		ViT-S	
Method	127M ↓	300M ↓	Top-1 ↑	CE ↓	Top-1 ↑	CE ↓
DynScalar $\times 1.5$	3.1817	2.9894	79.32	0.877	70.17	1.272
DynRealloc $\times 1.5$	3.1805	2.9884	79.20	0.880	69.99	1.315

For calibrated base updates $\mathbf{D}_i$ and allocation weights $v_i \in [1, 1.2]$, DynRealloc applies $(\gamma v_i/\zeta)\mathbf{D}_i$, where γ is 1 or 1.5 and

$$\zeta = \left(\frac{\sum_i \|\mathbf{D}_i\|_F^2 v_i^2}{\sum_i \|\mathbf{D}_i\|_F^2} \right)^{1/2}. \tag{11}$$

This makes the concatenated update norm γ times the calibrated base norm.

Plain-Muon controls. All arms in Table 3 use main learning rate 0.02, weight decay 0.01, and no norm preservation. Matrix control uses Eq. (6). Global control multiplies every current ordinary-Muon update by

$$k_t = \theta_t \frac{(\sum_i \|\mathbf{W}_{i,t}\|_F^2)^{1/2}}{(\sum_i \|\Delta \mathbf{W}_{i,t}^{\mathrm{Muon}}\|_F^2)^{1/2}}. \tag{12}$$

This enforces the concatenated angular step and preserves Muon's allocation across matrices. The learning-rate-only shape sets $a = b = 0$ and chooses θ_{peak} before training so that the sum of θ_t^2 over training matches the ASM shape. Absolute update norms can differ because the training trajectories and weight norms differ. The separate amplification control uses $k_t = 1.5$ without changing weight decay. Zero directions remain zero.

Table 8: Effect of the decay exponent b at 127M. We report the final validation loss, with all other settings fixed.

Decay exponent b	Validation loss ↓	Difference from $b = 0.7$
0	3.2071	+0.0272
0.5	3.1818	+0.0020
0.7	**3.1798**	—
1.0	3.1809	+0.0010

Table 9: Sensitivity to the nominal learning rate at 127M. We report the final validation loss; the range is the difference between the largest and smallest loss. The smallest range is in bold. Our method is shaded.

Method	LR 0.005	LR 0.01	LR 0.02	Range
Muon	3.2217	3.2063	3.2017	0.0200
DynMuon	3.1899	3.1927	3.1866	0.0061
DynScalar $\times 1.5$	3.1864	3.1817	3.1870	0.0052
DynRealloc $\times 1.5$	3.1855	3.1805	3.1868	0.0064
ASM	3.1811	3.1798	3.1776	**0.0036**

A.5 Schedule sensitivity

The sensitivity studies in Figures 5 and 6, Tables 8 and 9, and the rising-branch and weight-decay comparisons in Section 5.3 start from the 127M reference configuration in the last row of Table 6 and change one setting at a time, with the same 16,000-update budget. Their losses are therefore compared with each other rather than with Table 1.

A.6 Sensitivity to the nominal learning rate

Figure 6 and Table 9 vary the nominal learning rate while keeping ASM's angular schedule fixed. The target fixes the optimization-update length, while the nominal rate still affects weight decay. The observed NLL range of 0.0036 therefore describes stability with respect to this remaining role of the nominal rate. Sensitivity to ASM's controlled update amplitude is tested directly in Figure 5(c).

A.7 Time to a validation target

Table 4 uses the first validation evaluation during training at which each target is reached, without interpolation. Both trajectories use the 170M recipe described in Appendix A.3. Targets come from Muon's trajectory after training; elapsed times include evaluation. Full-budget times are 747.8 minutes for Muon and 764.5 minutes for ASM. The validation curves in Figure 3 and the targets in Table 4 come from evaluations on a held-out slice made periodically during training. The final losses in Tables 1 and 2 additionally include an evaluation after training on a separate held-out set, so the two measurements differ in absolute value: on the 170M recipe, for example, Muon's curve ends at 3.0258, whereas its final loss in Table 2 is 3.0622. All methods are compared under the same measurement.

A.8 Additional results and complete procedure

Table 10 reports the longer cosine-annealed recipe. Table 5 gives absolute step times and peak memory. Algorithm 2 collects the complete training procedure.

Table 10: Results on the 170M recipe with additional controls. We report the final validation loss. Best results are in bold and second-best results are underlined. Our method is shaded.

Method	Validation loss ↓
Muon	3.0622
DynMuon	3.0620
AngularMuown	<u>3.0243</u>
DynScalar $\times 1.0$	3.0590
DynRealloc $\times 1.0$	3.0588
RowEq-dynamic $\times 1.0$	3.0598
ASM (ours)	**2.9988**

Algorithm 2 Training procedure for ASM

Inputs: schedule $(\theta_{\text{peak}}, t_p, a, b, q)$; weight decay λ or norm preservation.

1. At step t, compute gradients.
2. Compute θ_t using Eq. (8).
3. For each matrix, update momentum, form $\mathbf{H}_{i,t}$, and compute $\mathbf{Y}_{i,t} = \mathcal{N}(\mathbf{H}_{i,t})$.
4. Form $\Delta\mathbf{W}_{i,t}^{\text{ASM}}$ using Eq. (6); use zero for a zero weight or direction.
5. Apply Eq. (9) for weight decay, or Eq. (10) for norm preservation.
6. Update auxiliary parameters with the recipe's optimizer.

B Properties of the Update

This section establishes properties of the update rule in exact arithmetic. The statements concern each controlled matrix at an individual step and do not require exact orthogonalization.

B.1 Angular-step control

Remark. For nonzero $\mathbf{W}$ and $\mathbf{Y}$, homogeneity of the Frobenius norm gives $\|\Delta\mathbf{W}\|_F = \theta\|\mathbf{W}\|_F$ directly from Eq. (6). Thus the angular step equals θ by construction, while the direction remains $-\mathbf{Y}/\|\mathbf{Y}\|_F$ when $\theta > 0$.

When $\mathbf{Y} = 0$, the implementation applies a zero update. A zero weight matrix also receives a zero update, and its angular step is undefined. These cases are handled without dividing by a zero norm.

B.2 Geometry of the experimental update settings

For nonzero $\mathbf{W}$ and $\mathbf{Y}$, let $\rho = \langle \mathbf{W}, \mathbf{Y} \rangle_F / (\|\mathbf{W}\|_F \|\mathbf{Y}\|_F)$. Expanding the squared norm of the trial update gives

$$\frac{\|\mathbf{W} + \Delta\mathbf{W}\|_F^2}{\|\mathbf{W}\|_F^2} = 1 - 2\theta\rho + \theta^2. \tag{13}$$

This follows from $\|\mathbf{W} + \Delta\mathbf{W}\|_F^2 = \|\mathbf{W}\|_F^2 + 2\langle \mathbf{W}, \Delta\mathbf{W} \rangle_F + \|\Delta\mathbf{W}\|_F^2$. It describes the trial update before the experimental postprocessing in Eqs. (9) and (10).

Proposition 1 (Well-defined norm preservation)**.** *If $\mathbf{W} \neq 0$ and $0 \leq \theta < 1$, the intermediate matrix $\mathbf{Z} = \mathbf{W} + \Delta\mathbf{W}$ is nonzero. The rescaled update $\mathbf{W}^+ = \|\mathbf{W}\|_F \mathbf{Z}/\|\mathbf{Z}\|_F$ satisfies $\|\mathbf{W}^+\|_F = \|\mathbf{W}\|_F$ and has the same direction as $\mathbf{Z}$.*

Proof. The reverse triangle inequality gives $\|\mathbf{Z}\|_F \geq \|\mathbf{W}\|_F - \|\Delta\mathbf{W}\|_F = (1 - \theta)\|\mathbf{W}\|_F > 0$. The remaining statements follow from multiplication by the positive scalar $\|\mathbf{W}\|_F/\|\mathbf{Z}\|_F$. □

More generally, the reverse triangle inequality gives $\|\mathbf{Z}\|_F \geq |1 - \theta| \|\mathbf{W}\|_F$. This lower bound is reached at $\mathbf{Y} = \mathbf{W}$, so it is exact in the worst case over directions. The normalization is therefore well defined for every direction whenever $\theta \neq 1$; at $\theta = 1$, a zero intermediate matrix is possible. The reported schedules use the small-step regime $\theta < 1$. The implementation uses a guarded

normalization for a zero intermediate matrix. Floating-point norm reductions and rescaling can introduce rounding error relative to the exact identities.

B.3 Relation to the geometric rotation angle

Let $\widehat{\mathbf{W}} = \mathbf{W}/\|\mathbf{W}\|_F$ and $\widehat{\mathbf{Y}} = \mathbf{Y}/\|\mathbf{Y}\|_F$, and write $\rho = \langle \widehat{\mathbf{W}}, \widehat{\mathbf{Y}} \rangle_F \in [-1, 1]$. Let $\alpha = 1 - \eta\lambda$ for decoupled weight decay and $\alpha = 1$ for the norm-preserving configuration, so that $\mathbf{Z} = \|\mathbf{W}\|_F(\alpha\widehat{\mathbf{W}} - \theta\widehat{\mathbf{Y}})$ is the update before any norm restoration. Whenever $\mathbf{Z} \neq 0$, the angle $\phi \in [0, \pi]$ between $\mathbf{W}$ and $\mathbf{Z}$ satisfies

$$\cos\phi = \frac{\alpha - \theta\rho}{\sqrt{\alpha^2 - 2\alpha\theta\rho + \theta^2}}. \tag{14}$$

This follows by taking the Frobenius inner product with $\widehat{\mathbf{W}}$ and computing the norm of $\alpha\widehat{\mathbf{W}} - \theta\widehat{\mathbf{Y}}$. Positive radial rescaling leaves ϕ unchanged, so Eq. (14) also applies after norm preservation.

With norm preservation enabled, $\alpha = 1$. For a tangent direction ($\rho = 0$), $\phi = \arctan\theta$, which agrees with θ to first order at small step sizes. For a fixed $|\rho| < 1$, expansion around $\theta = 0$ gives $\phi = \theta\sqrt{1 - \rho^2} + O(\theta^2)$. A radial direction can instead produce zero rotation even with a nonzero norm ratio. Thus, the controlled angular step is a relative-update measure and equals the geometric angle only approximately under additional conditions.

The final change under norm preservation has relative norm

$$\frac{\|\mathbf{W}^+ - \mathbf{W}\|_F}{\|\mathbf{W}\|_F} = \sqrt{2 - 2\cos\phi} = 2\sin(\phi/2). \tag{15}$$

Equation (7) therefore concerns the optimization update before the norm-preserving rescaling, rather than this final chord length.

B.4 Scope of the schedule

At $t = t_p$, both branches in Eq. (8) equal one, hence $\theta_{t_p} = \theta_{\text{peak}}$ provided $q(t_p) > 0$. The branch factor is continuous at the transition, and continuity of the combined schedule also requires continuity of q. For constant positive q and $b > 0$, the second branch decreases monotonically. The implemented warmup and final annealing remain part of the recipe. These finite-horizon schedules are selected empirically; a power-law exponent alone does not imply convergence of ASM with momentum and approximate orthogonalization.

C Diagnostic Trajectories

Figure 1(a,b) uses ASM with the 127M configuration in Table 6 and Muon, both at learning rate 0.02 and weight decay 0.01. Update norms exclude decoupled weight decay. The logged global norms concatenate the controlled matrices: $w_t = (\sum_i \|\mathbf{W}_{i,t}\|_F^2)^{1/2}$ and $d_t = (\sum_i \|\Delta\mathbf{W}_{i,t}\|_F^2)^{1/2}$. The plotted angular step is d_t/w_t. If every nonzero controlled matrix realizes the same target, the global ratio equals that target as well. Norms in panel (a) are normalized at step 500. The learning rate is constant from step 160 through step 12,800.

For Muon, the global angular step decreases from 0.0089226964 at step 500 to 0.0036408571 at step 12,800, a 59.2% decline. The weight norm grows by a factor of 2.501 over the same interval. For ASM, the measured and target curves use logged observations from the first update. Figure 1(c) uses a separate matched comparison at learning rate 0.01: weight decay 0.01 and 0.1 yield final validation NLL 3.2068 and 3.2192. This pair changes only weight decay and is not combined with the learning-rate-0.02 trajectory to estimate its effect.

C.1 Differences across matrices

Figure 4 in Section 5.3 shows the angular step of every controlled matrix in the same ordinary-Muon run used in Figure 1(a,b). Over the constant-learning-rate interval from step 500 to step 12,800, the median decline is 58.9%, ranging from 47.0% for the MLP input matrix in block 24 to 70.9% for the attention value matrix in block 20. The band in Figure 4(a) describes variation across matrices.