Anytime Training with Schedule-Free Spectral Optimization

Anuj Apte* Pranav Deshpande Niraj Kumar
Shouvanik Chakrabarti Junhyung Lyle Kim†

Global Technology Applied Research, JPMorganChase
New York, NY 10001, USA

May 2026

Abstract

Standard neural network training relies on learning-rate schedules tied to a fixed horizon, leading to strong path dependence and costly re-tuning as data availability changes. Schedule-Free (SF) methods address this by removing explicit schedules, yet SF-AdamW, the current state-of-the-art anytime optimizer, consistently underperforms well-tuned AdamW baselines. We propose SF-NorMuon, a schedule-free spectral optimizer that closes this gap: with a single hyperparameter configuration, SF-NorMuon matches or exceeds tuned AdamW on 125M and 772M parameter language models across 1–8× Chinchilla horizons. On the theoretical side, we prove a stationarity guarantee for schedule-free spectral dynamics and identify weight decay at the fast iterate as essential for long-horizon stability. SF-NorMuon enables practitioners to obtain high-quality checkpoints at any point during training without committing to a horizon in advance. By closing the performance gap with tuned baselines, SF-NorMuon makes horizon-free optimization more practical, taking a step towards truly open-ended, continual learning.

1 Introduction

In the standard machine learning paradigm, after curating a dataset, a model is trained and then evaluated on a held-out, previously unseen portion of the data [1, 2, 3]. However, models are increasingly being deployed in scenarios where large amounts of data comparable to or even exceeding the training data are generated during the deployment period. Examples of this include large language models (LLMs) [4, 5, 6, 7] interacting with billions of users, and Vision-Language-Action (VLA) models controlling robots in industries [8, 9, 10]. As a result, judiciously leveraging the a priori unknown amount of data available at deployment time is crucial for improving model capabilities through continual learning [11, 12]. However, beyond addressing catastrophic forgetting [13, 14], there is also an optimization obstacle to overcome [15, 16, 17, 18].

Most existing training methods are not designed to be anytime, as they depend on learning rate schedules tied to a fixed training horizon and require extensive hyperparameter tuning under a predetermined compute budget [19, 20, 21]. For example, the standard cosine decay schedule has a strong path dependence. Consider training an LLaMA-2-style transformer with cosine decay as shown in Figure 1. The black dashed lines highlight the performance discrepancy: at the same token count, a run tuned for a shorter horizon significantly outperforms one tuned for a longer horizon, despite seeing identical data (e.g.: 2× vs 4× Chinchilla at 31B tokens for 772M model). This occurs because cosine decay ties the learning rate to the training horizon. At any intermediate point, runs with longer horizons have barely begun decaying, while shorter-horizon runs have already annealed to near zero, resulting in the big discrepancy in performances. Ideally, we want the model to learn as much as possible from a given set of training data regardless of how much data awaits in the future.

*anuj.apte@jpmchase.com
†lyle.kim@jpmchase.com

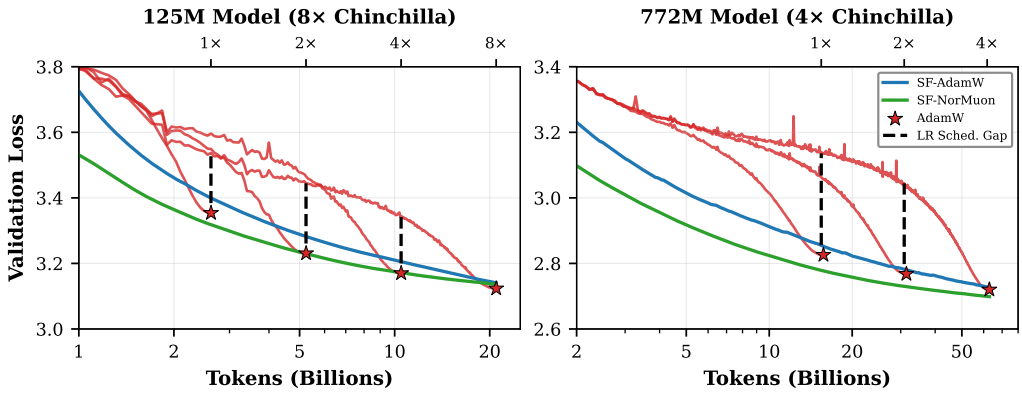

Figure 1: **Comparison of SF-NorMuon (this work), SF-AdamW, and *tuned* AdamW baselines for LLaMA-2-style transformers trained on FineWeb-100B.** *Left:* 125M model with AdamW learning rate tuned per horizon with cosine schedule. *Right:* 772M model with a single optimized AdamW configuration across horizons. Dashed lines highlight that learning rate schedule for a long horizon is sub-optimal for a smaller token budget. The grid of hyperparameters are summarized in Table 4, and validation losses for different learning rates are illustrated in Figure 6. SF-NorMuon consistently out performs SF-AdamW, and matches *tuned* AdamW baselines.

To that end, *Schedule-Free* (SF) methods [17] were recently proposed, removing the need for an explicit learning-rate schedule. The SF framework maintains three iterate sequences. In its most basic form, SF-SGD iterates as follows:

$$y_t = \beta x_t + (1-\beta) z_t, \tag{1}$$

$$z_{t+1} = z_t - \eta \nabla \mathcal{L}(y_t; \xi_t), \tag{2}$$

$$x_{t+1} = (1 - c_{t+1}) x_t + c_{t+1} z_{t+1}, \tag{3}$$

where $z_1 = x_1$ and $c_{t+1} = 1/(t+1)$. The y sequence is where the gradient is evaluated at interpolated points, in spirit of Nesterov's accelerated method [22, 23], the z sequence is the *fast iterate* taking the (stochastic) gradient direction with a constant step size η throughout the training, and the x sequence is the *evaluation* sequence where validation loss is computed. Rewriting the update, we obtain

$$x_{t+1} = x_t - \eta c_{t+1} \nabla \mathcal{L}(y_t; \xi_t) + c_{t+1}(z_{t+1} - x_t). \tag{4}$$

Thus, even though z moves at a constant rate, the effective learning rate for x is $\eta_{\text{eff}} = \eta/(t+1)$, which decays over the duration of the training. Note that since x is a linear combination of y and x it can be computed on the fly, and so it does not need additional memory overhead. An alternative but closely related approach for horizon-free training is simply training with a constant learning rate, and returning an some moving average[1] of the weights [24, 25, 16, 26]. Notably, SF-AdamW, the practical variant of SF-SGD, won the first prize in the self-tuning track of the `AlgoPerf` challenge [27], demonstrating a new level of effectiveness for anytime training. For completeness, Appendix B reviews SF-SGD and SF-AdamW, and summarizes the convergence guarantee for SF-SGD.

Despite this success, as observed in [28, 29, 30], SF-AdamW is unable to match the performance of well-tuned AdamW [31, 32], the primary workhorse for optimizing neural networks. This is also confirmed in Figure 1, where SF-AdamW (blue) consistently lies above the well-tuned AdamW baselines (red stars, tuning grid in Table 4) across all Chinchilla horizons. Since the schedule-free averaging mechanism in (1)–(3) is agnostic to the base update rule applied at (2), a natural question is whether a different choice of base optimizer can close this gap.

In modern neural networks, the vast majority of learnable parameters reside in weight matrices that act linearly on incoming activations. However, AdamW treats weight matrices as flat vectors, discarding how

[1] For instance, with $\beta = 0$ in (1), SF-SGD recovers constant SGD with uniform averaging.

they act on activations. The spectral norm provides a natural measure of the effect a weight matrix update has on the output activation, motivating recent interest in spectral optimization methods such as Muon [33, 34] and its variants [35, 36, 37, 38, 39, 40, 41, 42, 43, 44]. Muon performs steepest descent under the spectral norm via the polar decomposition of the gradient computed by Newton-Schulz iterations, and has been successfully scaled to trillions of tokens [45, 46, 47, 48]. Neural network loss surfaces are characterized by many nearly-flat directions corresponding to small Hessian eigenvalues [49, 50], and thus taking uniformly sized steps in all spectral directions via the polar decomposition enables Muon to extract more information from the training data per step [51]. This motivates the question:

Does optimization under the spectral norm improve upon existing schedule-free methods, and can it close the gap with well-tuned, horizon-dependent AdamW baselines?

Apart from the choice of base geometry, schedule-free methods exhibit a stability challenge that becomes critical for long horizon training. As shown in Figure 2, the validation loss increases for long runs in absence of weight decay. This is in sharp contrast to the convex optimization theory underlying schedule-free methods, where convergence holds without regularization [17, 52]. Moreover, weight decay at y as originally proposed in [17], is inadequate for the spectral case, where the instability is significantly amplified (right panel of Figure 2). Therefore, in schedule-free methods weight decay plays a qualitatively different role: it is not merely a regularizer that improves generalization [53, 54, 55], but is *necessary* for long-horizon stability.

Combining these two insights, spectral geometry to speed up training and weight decay at the fast iterate to ensure stability, our contributions are as follows:

- We propose SF-SpectralSGD (with momentum), a schedule-free spectral method for matrix optimization, and prove a stationarity guarantee matching the $\widetilde{O}(T^{-1/4})$ rate achieved by recent work on convergence of Muon [56, 57, 58, 59] (Theorem 2.1 and Corollary 2.1).
- For practical deep learning, we incorporate the per-neuron normalization of NorMuon [36, 35] and obtain SF-NorMuon, as summarized in Algorithm 1. SF-NorMuon substantially improves over the state-of-the-art anytime optimizer SF-AdamW, and matches tuned AdamW across training horizons (c.f., Figure 1, Table 1). In particular, including explicit momentum buffer and per-neuron normalization is crucial in achieving good practical performance (c.f., Figure 3).
- We identify the key role of weight decay in schedule-free optimization. Unlike in standard optimizers, the fast iterate z_t in (2) continues to move at constant learning rate throughout training, so weight decay is a necessity for long-horizon stability. For spectral optimization, the aggressive polar updates amplify the instability (Figure 2), but weight decay directly applied to z_t leads to stable training. We prove boundedness of all iterates (Lemma 3.1), and derive a closed-form steady-state characterization (Lemma 3.2) that closely matches empirical training dynamics (Figure 4).

2 Schedule-Free Spectral Optimization

The fundamental design choice in a schedule-free optimizer is the geometry used for the fast update. When inputs and outputs are measured in the Euclidean norm $\|\cdot\|_2$, the induced operator norm on a matrix $A \in \mathbb{R}^{m\times n}$ is

$$\|A\|_{2\to 2} := \sup_{\|u\|_2 \leqslant 1} \|Au\|_2 = \sigma_{\max}(A) = \|A\|_{\mathrm{op}}. \tag{5}$$

where $\sigma_{\max}(A)$ denotes the largest singular value of A, which is precisely the spectral norm. Geometrically, it measures the maximum factor by which A can stretch any unit vector, making it the natural measure of how much a weight perturbation can alter the network's activations. The analysis for other induced norms, and their corresponding optimizers is presented in Appendix A.

For matrix-valued weights acting as linear maps on activations, the spectral norm is the natural metric, since it equals the operator norm governing the largest change an update induces on activations. The steepest-descent

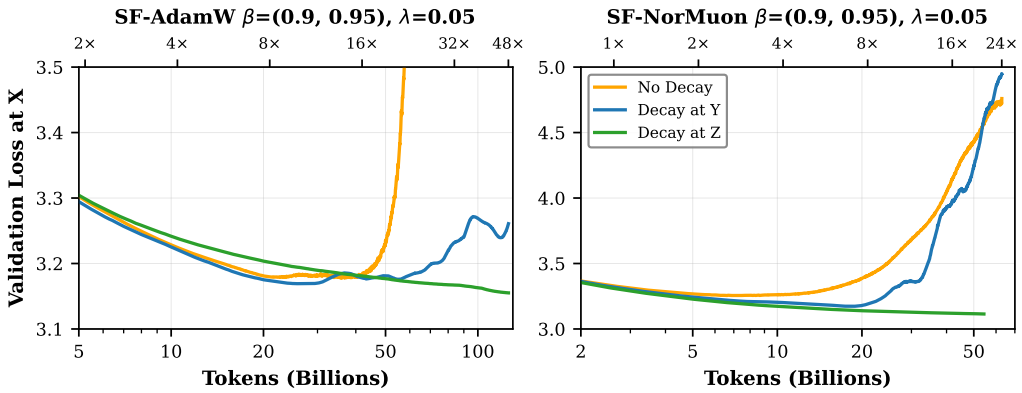

Figure 2: **Weight decay strategies for schedule-free optimizers.** *Left:* SF-AdamW with no decay (orange) diverges; decay at Y (blue) exhibits the best performance up to ∼30B tokens (c.f., Table 1), but eventually diverges. Decay at Z (green) yield stable training, but is suboptimal in the early phase. *Right:* SF-NorMuon with no decay (orange) diverges faster; decay at Y (blue) is also unstable. Decay at Z (green) maintains stable and performs the best throughout, motivating the analysis in Section 3.

direction under a spectral-norm constraint solves [60, 61]

$$\operatorname{argmin}_{\|\Delta W\|_{\mathrm{op}} \leqslant \eta} \langle \nabla f(W), \Delta W \rangle_F. \tag{6}$$

Let $\nabla f(W) = U\Sigma V^\top$ be the SVD with $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ orthonormal and $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_r)$. Decomposing $\Delta W = UQV^\top + R$ where R is orthogonal to the column/row space of the gradient, only Q contributes:

$$\langle \nabla f(W), \Delta W \rangle_F = \operatorname{tr}(\Sigma^\top Q) = \sum_{i=1}^{r} \sigma_i Q_{ii}. \tag{7}$$

Since $\|Q\|_{\mathrm{op}} \leqslant \eta$ implies $|Q_{ii}| \leqslant \eta$, the minimum is achieved at $Q^\star = -\eta I_r$. Thus, the polar factor is the steepest-descent direction under spectral norm:

$$\Delta W^\star = -\eta\, UV^\top = -\eta\, \operatorname{polar}(\nabla f(W)). \tag{8}$$

Unlike Adam [31] or Lion [62] which act entry-wise and distort the gradient's singular-vector structure, the polar factor $UV^\top$ preserves the left and right singular subspaces. The polar update extracts learning signal from all singular directions of the gradient and not just those with the largest entries, and by making all the singular values unity, it makes uniform progress along all singular directions. This uniform treatment is more aggressive than coordinate-wise updates, so we include an explicit momentum buffer M_t to smooth the gradient before taking its polar factor. We build on this spectral-norm geometry to design a schedule-free optimizer that respects how weight matrices act on activations, while obtaining implicit learning-rate decay through online averaging.

2.1 Schedule-Free Spectral Descent with Momentum

For parameters $\beta, \mu \in [0, 1)$ and step-size $\eta > 0$, update rules for SF-Spectral Descent with Momentum are given by

$$\begin{aligned} Y_t &= \beta X_t + (1-\beta) Z_t, & G_t &= \nabla f(Y_t; \xi_t), & M_t &= \mu M_{t-1} + (1-\mu) G_t, \\ P_t &= \operatorname{polar}(M_t), & Z_{t+1} &= Z_t - \eta P_t, & X_{t+1} &= (1-c_{t+1}) X_t + c_{t+1} Z_{t+1}, \end{aligned} \tag{9}$$

with $c_{t+1} = 1/(t+1)$, $X_1 = Z_1$ and $M_1 = G_1$. Here Z_t is the fast sequence with polar update, X_t is the returned schedule-free average, Y_t is the gradient-evaluation point, and M_t is an explicit momentum buffer, which is crucial in practical performance (c.f., Figure 3).

Assume the existence of $W^\star \in \mathbb{R}^{m\times n}$ such that $f(W^\star) = f^\star$ and $\nabla f(W^\star) = 0$, and define

$$r := \min\{m, n\}, \qquad \Delta := f(Y_1) - f^\star. \tag{10}$$

Our convergence analysis relies on the following standard assumptions:

Assumption 2.1 (Frobenius Lipschitz smoothness)**.** *There is a constant $L_F > 0$ such that for all $U, V \in \mathbb{R}^{m\times n}$,*

$$\|\nabla f(U) - \nabla f(V)\|_F \leqslant L_F \|U - V\|_F . \tag{11}$$

Assumption 2.2 (Unbiased gradient and bounded-variance noise)**.** *Let $\mathcal{F}_t$ be the sigma-field generated by all randomness up to the construction of Y_t. We assume, for some batch size $B \geqslant 1$ and $\sigma^2 \geqslant 0$,*

$$\mathbb{E}\big[G_t \mid \mathcal{F}_t\big] = \nabla f(Y_t), \qquad \mathbb{E}\Big[\|G_t - \nabla f(Y_t)\|_F^2 \mid \mathcal{F}_t\Big] \leqslant \sigma^2/B. \tag{12}$$

Assumption 2.3 (Bounded diameter)**.** *There exists a constant $D_F > 0$ such that*

$$\|W^\star\|_F \leqslant D_F/2, \qquad \|Z_t\|_F \leqslant D_F/2 \qquad \text{for all } t. \tag{13}$$

Note that bounded iterate assumption is typically employed in the convergence analysis of spectral methods (e.g., see [63, Theorem 7] and [39, Theorem 1]), and if necessary it can be enforced algorithmically by considering a projected variant that keep the fast iterates Z_t inside the corresponding norm ball. Further, with weight decay at Z, Assumption 2.3 can be lifted, as can be seen in Lemma 3.1.

Under these assumptions, we obtain the following convergence guarantee:

Theorem 2.1 (Stationarity of SF-Spectral Descent with Momentum)**.** *Under Assumption 2.1–2.3, for every $T \geqslant 1$, SF-Sepctral Descent with Momentum in (9) satisfies the following:*

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant{}& \frac{\Delta + 2L_F D_F^2\big(\log(eT) + \pi^2/6\big)}{(1-\beta)T\eta} + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\left(\sqrt{\frac{1-\mu}{1+\mu}} + \frac{1}{(1-\mu)T}\right)\\ &+ \frac{2\sqrt{r}\,L_F D_F \log(eT)}{(1-\beta)T}\left(\frac{1+\mu}{1-\mu}\right) + \frac{L_F r\,\eta}{(1-\beta)}\left(\frac{1+3\mu}{1-\mu}\right).\end{aligned} \tag{14}$$

The first term is the descent term, the next captures the stochastic noise, the third term is the price of schedule-free drift, and the final term reflects the constant spectral step and momentum tracking. The explicit momentum stabilizes the aggressive polar steps at the cost of a tracking error. By optimizing the terms that appear on the right hand side of the theorem above, we obtain the following corollary for any $\beta \in [0, 1)$, both with and without noise.

Corollary 2.1 (Optimized convergence rates for SF-Spectral Descent)**.** *Under Assumption 2.1–2.3, for any fixed $\beta \in [0, 1)$ there exist choices $\mu_T \in [0, 1)$ and $\eta_T > 0$ such that*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla f(Y_t)\|_* = \widetilde{O}(T^{-1/4}) \quad (\sigma > 0); \quad \text{and} \quad \frac{1}{T}\sum_{t=1}^{T}\|\nabla f(Y_t)\|_* = \widetilde{O}(T^{-1/2}) \quad (\sigma = 0). \tag{15}$$

These convergence rates match the ones derived in recent works on convergence of Muon [56, 57, 58, 59]. An analogous result can also be proven under Lipschitz smoothness measured in the spectral norm rather than the Frobenius norm. Since the Frobenius-smooth version is more directly comparable to existing matrix-optimization methods, we state it here and defer the spectral-smooth variant, together with the full proofs of both results, to the Appendix C.

Algorithm 1 Schedule-Free NorMuon (SF-NorMuon)

Input: base learning rate η, interpolation parameter β_1, variance parameter β_2, momentum parameter μ, perturbation parameter ε, weight decay λ, warmup steps T_{warmup}
for $t = 1$ **to** T **do**
  $Y_t \leftarrow (1-\beta_1)Z_t + \beta_1 X_t$ ▷ Interpolate weights
  $G_t \leftarrow \nabla_{\mathbf{W}} \mathcal{L}(Y_t, \zeta_t)$ ▷ Compute gradient at interpolation point
  $M_t \leftarrow \mu M_{t-1} + (1-\mu)G_t$ ▷ Explicit momentum buffer
  ---
  $P_t \leftarrow \text{polar}(M_t)$ ▷ Polar factor of momentum
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2)\,\text{mean}_{\text{cols}}(P_t \odot P_t)$ ▷ Row-wise second moment
  $V_t \leftarrow \text{ExpandRows}(v_t)$ ▷ Broadcast to matrix
  $\widehat{P}_t \leftarrow P_t \oslash (\sqrt{V_t} + \varepsilon)$ ▷ Adaptive row-wise normalization
  ---
  $\eta_t \leftarrow \eta \min(1, t/T_{\text{warmup}})$ ▷ Learning rate with warmup
  $\hat{\eta}_t \leftarrow 0.2\eta_t\sqrt{mn}/\|\widehat{P}_t\|_F$ ▷ Match Adam RMS scaling
  $s_t \leftarrow s_{t-1} + \eta_t^2$
  $c_{t+1} \leftarrow \eta_t^2 / s_t$ ▷ Averaging coefficient
  ---
  $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ ▷ Update with weight decay at Z_t
  $X_{t+1} \leftarrow (1-c_{t+1})X_t + c_{t+1}Z_{t+1}$ ▷ Update averaged iterate
end for
Return: X_T

2.2 Schedule-Free NorMuon for Neural Network Training

To make schedule-free spectral descent practical for neural network training, we incorporate three refinements below. The resulting algorithm is SF-NorMuon, which we summarize in Algorithm 1. A PyTorch implementation of this algorithm is provided in Appendix F.

Learning rate warm up and scaling. We warm up the learning rate linearly, which stabilizes early training [64] and enables larger learning rates [65]. After computing the adaptively normalized direction $\widehat{P}_t$, the algorithm scales the base learning rate by $\hat{\eta}_t = 0.2\eta_t\sqrt{\max(m,n)}/\|\widehat{P}_t\|_F$. This scaling serves two purposes [45]: (i) It normalizes the effective step size to be comparable to Adam's RMS-normalized updates, enabling η and λ to be shared with SF-AdamW; and (ii) it provides additional stability by accounting for the overall magnitude of the normalized update direction.

Weight decay at the fast iterate Z. Weight decay is applied to Z_t, rather than Y_t (as originally proposed in [66] for SF-AdamW). This choice treats weight decay as a modification to the base optimizer dynamics rather than as an ℓ_2 regularizer in the loss, which would correspond to decay at Y_t. The scaled weight decay term $\eta_t\lambda Z_t$ is subtracted from Z_t in the base update. Weight decay at Z_t is necessary for long-horizon training with schedule-free methods, as we explain in Section 3.

Row-wise Normalization. We incorporate row-wise adaptive normalization [36, 35]. The polar factor can produce orthogonalized updates with high variance in step sizes across individual neurons, as different rows of a weight matrix may have vastly different update magnitudes. To accomodate, SF-NorMuon maintains a running average $v_t \in \mathbb{R}^m$ of the squared row norms of the polar factor P_t, capturing the typical magnitude of updates for each neuron and enabling per-neuron step size adaptation. The exponential moving average with parameter β_2 provides stability while remaining responsive to changes in gradient statistics, analogous to the second-moment estimation in Adam, but operating at the neuron level rather than element-wise.

In Figure 3, we ablate the two key aspects of SF-NorMuon: (i) the explicit momentum buffer (by setting $\mu = 0$), which reduces to direct polar decomposition of the gradient, and (ii) the row-wise adaptive normalization (by removing the v_t computation and normalization step), which reduces to uniform spectral steps. These ablations help isolate the contribution of each component to the overall performance. We find that removing the row-wise adaptive normalization slows down convergence by around 12% at 8x Chinchilla ratio, but removing momentum leads to a significant drop in performance. Note that the original SF-SGD or SF-AdamW does

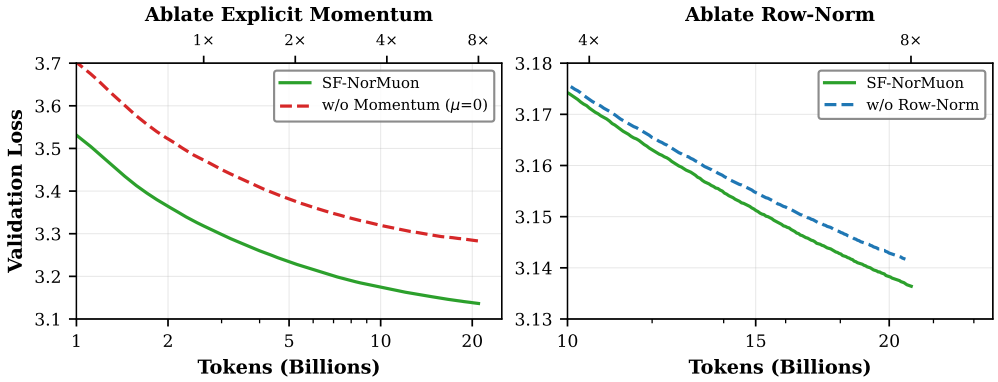

Figure 3: **Importance of explicit momentum and row-wise normalization for SF-NorMuon.** *Left:* Ablating explicit momentum ($\mu = 0$) significantly degrades performance, with final loss increasing from 3.14 to 3.28. This validates the importance of smoothing the gradient before computing the polar factor, as discussed in Section 2. *Right:* Ablating row-wise normalization leads to a smaller gap, with SF-NorMuon reaching the same loss approximately 12% faster.

not have an explicit momentum buffer. Therefore, the above ablation study illustrates how direct application of schedule-free dynamics to spectral optimzer (e.g., replacing $\nabla\mathcal{L}(y_t;\xi_t)$ in (2) with $\text{polar}(\nabla\mathcal{L}(y_t;\xi_t))$) would perform.

Remark 1 (Memory footprint)**.** *Consider a matrix parameter of shape $m \times n$. For SF methods, we need to track Y and Z (since X can be computed on the fly). For SF-AdamW the second moment buffer (per parameter) takes up memory mn, whereas SF-NorMuon uses a row-wise second-moment buffer of size m. Thus, there is room for an explicit momentum buffer matrix M of size $m \times n$ buffer unlike SF-AdamW. Hence, the total persistent memory is $3mn$ for SF-AdamW and $3mn + m$ for SF-NorMuon, compared with $3mn$ for AdamW and $2mn + m$ for NorMuon.*

3 Necessity of Weight Decay in Long Horizon Training

Modern language models are trained for very long horizons [67, 68, 69], and the training tokens for Mixture-of-Experts models can exceed 30 times the Chinchilla ratio to the number of active parameters [47, 48, 46]. Thus, to assess long-horizon behavior of schedule-free methods, we train the 125M LLaMa-2-style model for up to 48× Chinchilla with the default hyper-parameters.

Interestingly, for SF methods, the validation loss at X diverges without weight decaying *specifically at* Z. This is in sharp contrast to convex optimization theory, where convergence holds without regularization [17, 52]. In particular, for SF-AdamW in Figure 2 (left panel), decay at Y performs the best up to around 30B tokens, but eventually leads to divergence. Note that the original paper [17] recommended decaying at Y, which is sensible from the perspective of ℓ_2 regularization. In contrast, for SF-NorMuon in Figure 2 (right panel), decaying at Z is not only stable (see also Lemma 3.1) but also maintains the best validation loss throughout the long training exceeding 50B tokens.

To better understand this phenomenon, we consider a quasi steady-state analysis, following [70, 71]. Recall from (4) that the schedule-free method can be rewritten as

$$X_{t+1} = X_t - c_{t+1}\eta U_t + c_{t+1}(Z_{t+1} - X_t), \tag{16}$$

where $c_{t+1} = 1/(t+1)$, and U_t is either gradient divided by second moment buffer for SF-AdamW or polar transform of momentum buffer for SF-NorMuon, respectively. As can be seen, the evaluation sequence X_t evolves with an *effective* learning rate $\eta_t^{\text{eff}} = c_{t+1}\eta = \eta/(t+1)$ that decays to zero as $t \to \infty$, which makes the divergence surprising.

The culprit is in the fast Z_t sequence: it evolves with constant learning rate η throughout training, and in fact, we observe in practice that validation loss at Z_t often diverges. Since X_t averages all past $\{Z_s\}_{s\leqslant t}$, Z_t eventually contaminates the X_t sequence.

For SF-NorMuon, applying weight decay directly at Z_t addresses this issue, while maintaining good performance. To make this precise, we prove that weight decay at Z_t ensures all iterates to remain bounded, and characterize the precise steady-state norm to which Z_t converges.

Lemma 3.1. *Consider the update $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ and assume $0 < \eta\lambda < 1$ with $\hat{\eta} = 0.2\eta\sqrt{mn}/\|\widehat{P}_t\|_F$. Then, we have for all t,*

$$\|Z_t\|_F \leqslant \|Z_0\|_F(1-\eta\lambda)^t + \frac{0.2\sqrt{mn}}{\lambda} \implies \|Z_t\|_F \leqslant \frac{0.2\sqrt{mn}}{\lambda} \text{ as } t \to \infty. \tag{17}$$

Furthermore, since X_t and Y_t are convex combinations of $\{Z_s\}_{s\leqslant t}$, they satisfy the same bound.

In Figure 2, we see that weight decay at Y_t eventually leads to divergence, for both SF-NorMuon and SF-AdamW. We make the following remark regarding this observation:

Remark 2. *For weight decay at Y_t, the coupled dynamics (Z_t, D_t) where $D_t = Z_t - X_t$ satisfy:*

$$\begin{bmatrix} \|Z_{t+1}\|_F \\ \|D_{t+1}\|_F \end{bmatrix} \leqslant \begin{bmatrix} 1-\eta\lambda & \eta\lambda\beta \\ (1-c_{t+1})\eta\lambda & (1-c_{t+1})(1+\eta\lambda\beta) \end{bmatrix} \begin{bmatrix} \|Z_t\|_F \\ \|D_t\|_F \end{bmatrix} + 0.2\eta\sqrt{mn} \begin{bmatrix} 1 \\ 1-c_{t+1} \end{bmatrix}. \tag{18}$$

In the long-horizon limit $c_t \to 0$, the iteration matrix above has the leading eigenvalue

$$\lambda_1 = 1 + \frac{\eta\lambda}{2}\left(\sqrt{1+6\beta+\beta^2} - (1-\beta)\right) > 1 \tag{19}$$

for all $\beta > 0$, and so the triangle-inequality upper bound similar to (17) diverges.

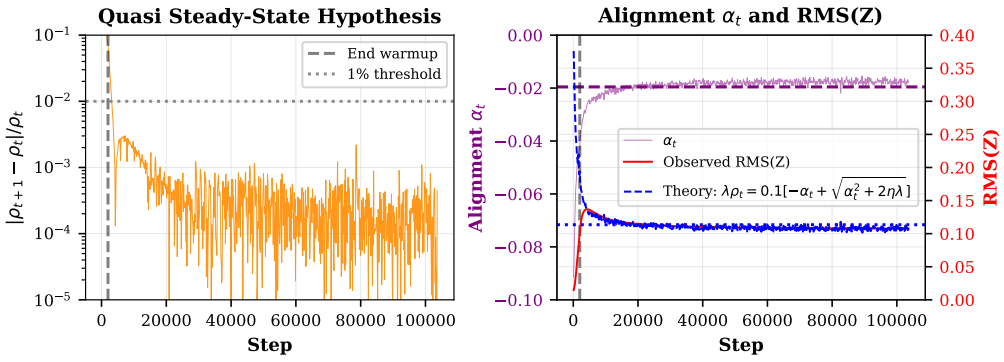

Figure 4: **Quasi steady-state analysis for SF-NorMuon with decay at Z (averaged over layers).** *Left:* The ratio $|\rho_{t+1} - \rho_t|/\rho_t$ remains below 1% after warmup, validating the quasi steady-state hypothesis (Hypothesis 3.1). *Right:* Alignment α_t (purple) and RMS(Z) (red) over training. The theoretical prediction $\lambda\rho_t = 0.1[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}]$ from Lemma 3.2 (blue dashed) closely tracks the observed values. This training run corresponds to $\eta = 0.01, \lambda = 0.05$.

Having established iterate boundedness in Lemma 3.1, we now study how the norm of Z evolves during the training. For this purpose, the analysis greatly simplifies if we make a quasi steady-state hypothesis following [70, 71], which states that change in norm of Z between two iterations is much smaller than the norm itself.

Hypothesis 3.1 (Quasi Steady-State)**.** *There exists an iteration count τ and a constant $\varepsilon \geqslant 0$ such that for all $t \geqslant \tau$,*

$$|\|Z_{t+1}\|_F - \|Z_t\|_F| \leqslant \varepsilon \|Z_t\|_F \ . \tag{20}$$

To enable uniform comparison across layers of different shapes, we define the RMS norm $\rho_t := \|Z_t\|_F/\sqrt{mn}$ for an $m \times n$ weight matrix, and the alignment $\alpha_t := \langle \widehat{P}_t, Z_t \rangle / (\|\widehat{P}_t\|_F \|Z_t\|_F) \in [-1, 1]$, which is the cosine of the angle between the preconditioned update and the current iterate. The left panel of Figure 4 shows that $|\rho_{t+1} - \rho_t|/\rho_t$ remains small after warmup, validating the hypothesis. Based on this, we derive the steady-state value of ρ_t :

Lemma 3.2 (Quasi Steady-State of Z)**.** *Consider the update* $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ *and assume* $0 < \eta\lambda \ll 1$ *with* $\hat{\eta} = 0.2\eta\sqrt{mn}/\|\widehat{P}_t\|_F$*. Then, the steady-state satisfies*

$$\lambda\rho_t = 0.1\Big[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\Big]. \tag{21}$$

After the end of the warm-up period, the alignment α_t approaches a small and negative value that persists for the rest of the training period as shown by the purple curve in right panel of Figure 4. Based on the Lemma 3.2, we expect that the RMS norm of Z (depicted in red) will asymptote to a constant value, which is also confirmed by the red curve sitting on top of the theoretical prediction shown in blue. Crucially, since the Z norm reaches a fixed value, the fast iterate remains controlled throughout training, and stabilizes other sequences as shown in Lemma 3.1.

4 Experiments on Training Language Models

Table 1: **Validation loss at X across Chinchilla horizons.** The best loss at each horizon is indicated with **bold**. *Speedup* is the percentage of steps saved by SF-NorMuon to reach the final loss of SF-AdamW. SF methods use hyperparameters tuned at 2× Chinchilla for both models. AdamW uses cosine schedule tuned per-horizon (c.f., Table 4) for 125M model, and the configuration for 8× was reused for the 772M model. SF-AdamW uses weight decay at Y as it performs the best in these horizons (c.f., Figure 2) and is the originally proposed option in [17].

	125M Model				**772M Model**			
Chinchilla Ratio	SF-NorMuon (decay@Z)	SF-AdamW (decay@Y)	AdamW (cosine)	Speedup	SF-NorMuon (decay@Z)	SF-AdamW (decay@Y)	AdamW (cosine)	Speedup
1×	**3.318**	3.399	3.354	35%	**2.777**	2.852	2.826	50%
2×	**3.229**	3.282	3.231	35%	**2.729**	2.780	2.768	52%
4×	**3.170**	3.205	3.170	36%	**2.698**	2.726	2.720	47%
8×	3.136	3.141	**3.124**	11%	—	—	—	—

We validate SF-NorMuon on auto-regressive language modeling by training decoder only transformers at two scales: a 125M parameter model and a 772M parameter model. Both architectures follow the LLaMA-2 design [72], with tied input-output embeddings, rotary positional embeddings (RoPE) [73], RMSNorm for pre-normalization [74], and squared ReLU activations in the MLP blocks [75]. Additional architecture details are provided in Appendix D. Our training code is based on the NanoGPT repository [76, 77].

Dataset and training setup. We train on the FineWeb-100B dataset [78] tokenized with the GPT-2 tokenizer [79]. All experiments use batch size of 512 sequences and sequence length of 1024 tokens, yielding approximately 524K tokens per gradient step. We adopt the Chinchilla scaling [20], where $N\times$ Chinchilla denotes training on $\sim 20N$ tokens per model parameter. For our 125M model, this corresponds to 2.5B tokens at 1× (5,000 steps), scaling up to 21B tokens at 8× (40,000 steps).

Hyperparameter tuning. We start with the 150M model. For schedule-free optimizers, we use 2,000 warm up steps and tune hyper-parameters at 2× Chinchilla, and reuse these for all settings. The best hyperparameters for SF-AdamW are $\eta = 0.01$, $(\beta_1, \beta_2) = (0.95, 0.99)$, and $\lambda = 0.05$. For SF-NorMuon, we reuse the same learning rate and weight decay parameter, following the $0.2\eta_t\sqrt{mn}/\|\widehat{P}_t\|_F$ rescaling factor from [46]. The other hyper-parameters were set to $(\beta_1, \beta_2) = (0.9, 0.95)$, momentum $\mu = 0.8$. For the non-matrix

parameters (embeddings, layer norms), both schedule-free methods run SF-AdamW. For the AdamW baseline, we tune the learning rate *separately at each horizon*, using 2,500 warm up steps followed by cosine decay to zero; comparison of these runs can be found in Figure 6 in Appendix D. Other hyperparameters were tuned at 2× Chinchilla. This represents a well-tuned baseline with per-horizon optimization, using optimized $\eta \in \{0.004, 0.008\}$, and $\beta_1 = 0.9$, $\beta_2 = 0.95$, and $\lambda = 0.1$. Additional hyper-parameter tuning details and learning rate sweeps are provided in Table 4 in Appendix D.

For the 772M parameter model, we reuse the optimal 8× Chinchilla hyper-parameters from AdamW and the 2×-tuned hyper-parameters for schedule-free methods. This approach is motivated by the observation that losses near the optimal learning rates exhibit only small variation, and recent work has shown similar optimal learning rates across small and large Llama-2 model scales (see Table 7 and Table 35 in [30]). We omit the 8× Chinchilla runs for the large model due to compute limitations.

Results. Table 1 reports validation losses (evaluated at X) across training horizons. Note that the SF-NorMuon uses exactly the same learning rate and weight decay as SF-AdamW. For both models, SF-NorMuon with weight decay at Z consistently outperforms SF-AdamW across all horizons. Crucially, SF-NorMuon matches the per-horizon tuned AdamW baselines (except at 8× Chinchilla where it lags marginally), while SF-AdamW lags behind consistency. This demonstrates that SF-NorMuon delivers competitive performance with AdamW at any stopping point, without a horizon-specific learning rate schedule. For the 772M model, the speedup advantage of SF-NorMuon over SF-AdamW is even more pronounced: 50% at 1× and 52% at 2× Chinchilla.

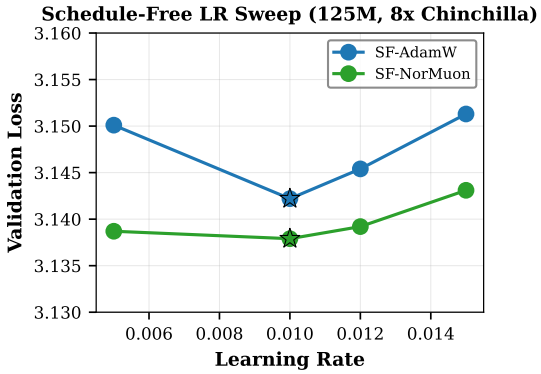

Figure 5: Learning rate sweep comparison between SF-AdamW and SF-NorMuon.

Improvement across learning rates. A key practical advantage of SF-NorMuon is robustness to the choice of learning rate. Figure 5 shows a learning-rate sweep at 8× Chinchilla for both schedule-free methods: SF-NorMuon achieves lower validation loss than SF-AdamW across all learning rates tested. Importantly, SF-NorMuon's performance degrades gracefully away from the optimum, confirming that the spectral geometry provides a favorable optimization landscape even without careful per-run tuning.

Other findings. We present the full convergence proofs under both Frobenius and spectral smoothness in Appendix C. Additional material includes: a self-contained derivation of spectral and sign update rules from first principles (Appendix A), a review of SF-SGD and SF-AdamW (Appendix B), detailed model architectures and hyperparameter search spaces (Appendix D), an ablation study with a comparison against tuned NorMuon across all horizons (Appendix E), and a PyTorch implementation of SF-NorMuon in Algorithm 1(Appendix F).

5 Conclusion

In this work, we proposed SF-NorMuon, a schedule-free spectral optimizer that matches per-horizon tuned AdamW across Chinchilla horizons without requiring schedule or horizon specification in advance. We proved a $\widetilde{O}(T^{-1/4})$ stationarity guarantee for the underlying schedule-free spectral dynamics for smooth non-convex functions and identified weight decay at the fast iterate is a necessity for long-horizon stability, which is a departure from the convex optimization theory. A limitation of this work is the diversity of model architectures and datasets; extending the validation of SF-NorMuon to more diverse settings, and considering larger scales and multi-stage continual learning pipelines are therefore a natural direction for future work.

Acknowledgments

We thank Anthony Ashmore for insightful discussions on the importance of learning rate scheduling. We thank Aaron Defazio for valuable guidance on schedule-free methods. We are grateful to Pragna Subrahmanya for assistance with computing infrastructure. We thank Rob Otter for the executive support of the work, and our colleagues at the Global Technology Applied Research center of JPMorganChase for support.

Author Contributions

A.A. and J.L.K. conceived the algorithm. A.A. developed the codebase, conducted the language modeling experiments, and established the convergence and steady-state theory. P.D. performed additional experiments. J.L.K. carried out exploratory experiments, shaped the overall research direction and oversaw the presentation of the material. S.C. and N.K. provided feedback and contributed to discussions throughout the project. All authors contributed to writing the manuscript.

Disclaimer

References

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. ISBN: 9780387848587. DOI: 10.1007/978-0-387-84858-7. URL: http://dx.doi.org/10.1007/978-0-387-84858-7.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[3] Christopher M. Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer International Publishing, 2024. ISBN: 9783031454684. DOI: 10.1007/978-3-031-45468-4. URL: http://dx.doi.org/10.1007/978-3-031-45468-4.

[4] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[5] Hugo Touvron et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. arXiv: 2302.13971 [cs.CL]. URL: https://arxiv.org/abs/2302.13971.

[6] Wayne Xin Zhao et al. *A Survey of Large Language Models*. 2023. eprint: arXiv:2303.18223.

[7] Shervin Minaee et al. *Large Language Models: A Survey*. 2024. eprint: arXiv:2402.06196.

[8] Anthony Brohan et al. *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*. 2023. eprint: arXiv:2307.15818.

[9] Yueen Ma et al. *A Survey on Vision-Language-Action Models for Embodied AI*. 2024. eprint: arXiv:2405.14093.

[10] Moo Jin Kim et al. *OpenVLA: An Open-Source Vision-Language-Action Model*. 2024. eprint: arXiv:2406.09246.

[11] German I. Parisi et al. "Continual lifelong learning with neural networks: A review". In: *Neural Networks* 113 (May 2019), pp. 54–71. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.01.012. URL: http://dx.doi.org/10.1016/j.neunet.2019.01.012.

[12] Liyuan Wang et al. *A Comprehensive Survey of Continual Learning: Theory, Method and Application*. 2024. arXiv: 2302.00487 [cs.LG]. URL: https://arxiv.org/abs/2302.00487.

[13] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114.13 (Mar. 2017), pp. 3521–3526. ISSN: 1091-6490. DOI: 10.1073/pnas.1611835114. URL: http://dx.doi.org/10.1073/pnas.1611835114.

[14] Matthias Delange et al. "A continual learning survey: Defying forgetting in classification tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3057446. URL: http://dx.doi.org/10.1109/TPAMI.2021.3057446.

[15] Thomas Degris et al. "Step-size optimization for continual learning". In: *arXiv preprint arXiv:2401.17401* (2024).

[16] Alexandru Meterez et al. "Anytime Pretraining: Horizon-Free Learning-Rate Schedules with Weight Averaging". In: *arXiv preprint arXiv:2602.03702* (2026).

[17] Aaron Defazio et al. "The road less scheduled". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 9974–10007.

[18] Priya Kasimbeg et al. "How far away are truly hyperparameter-free learning algorithms?" In: *arXiv preprint arXiv:2505.24005* (2025).

[19] Ilya Loshchilov and Frank Hutter. "SGDR: Stochastic Gradient Descent with Warm Restarts". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=Skq89Scxx.

[20] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. URL: https://arxiv.org/abs/2203.15556.

[21] Shengding Hu et al. *MiniCPM: Unveiling the Potential of Small Language Models with Scalable Training Strategies*. 2024. arXiv: 2404.06395 [cs.CL]. URL: https://arxiv.org/abs/2404.06395.

[22] Yurii Nesterov. "A method for solving the convex programming problem with convergence rate O (1/k2)". In: *Dokl akad nauk Sssr*. Vol. 269. 1983, p. 543.

[23] Guanghui Lan. "An optimal method for stochastic composite optimization". In: *Mathematical Programming* 133.1 (2012), pp. 365–397.

[24] Alexander Hägele et al. *Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations*. 2024. arXiv: 2405.18392 [cs.LG]. URL: https://arxiv.org/abs/2405.18392.

[25] Yunshui Li et al. *Model Merging in Pre-training of Large Language Models*. 2025. arXiv: 2505.12082 [cs.CL]. URL: https://arxiv.org/abs/2505.12082.

[26] Pavel Izmailov et al. "Averaging weights leads to wider optima and better generalization". In: *arXiv preprint arXiv:1803.05407* (2018).

[27] Priya Kasimbeg et al. "Accelerating neural network training: An analysis of the AlgoPerf competition". In: *The Thirteenth International Conference on Learning Representations*. 2025. URL: https://openreview.net/forum?id=CtM5xjRSfm.

[28] Alexander Hägele et al. "Scaling laws and compute-optimal training beyond fixed training durations". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 76232–76264.

[29] Minhak Song et al. "Through the River: Understanding the Benefit of Schedule-Free Methods for Language Model Training". In: *High-dimensional Learning Dynamics 2025*. 2025. URL: https://openreview.net/forum?id=b5HYeRzG9M.

[30] Andrei Semenov, Matteo Pagliardini, and Martin Jaggi. *Benchmarking Optimizers for Large Language Model Pretraining*. 2026. URL: https://openreview.net/forum?id=Jw7khYzYzl.

[31] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. 2015. URL: https://arxiv.org/abs/1412.6980.

[32] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

[33] Keller Jordan et al. *Muon: An optimizer for hidden layers in neural networks*. https://kellerjordan.github.io/posts/muon/. Accessed: 2026-01-25. 2024.

[34] Keller Jordan et al. *Muon (GitHub repository): An optimizer for hidden layers in neural networks*. GitHub repository, master branch. 2024. URL: https://github.com/KellerJordan/Muon (visited on 01/25/2026).

[35] Chongjie Si, Debing Zhang, and Wei Shen. *AdaMuon: Adaptive Muon Optimizer*. 2025. arXiv: 2507.11005 [cs.LG]. URL: https://arxiv.org/abs/2507.11005.

[36] Zichong Li et al. *NorMuon: Making Muon more efficient and scalable*. 2025. arXiv: 2510.05491 [cs.LG]. URL: https://arxiv.org/abs/2510.05491.

[37] Kwangjun Ahn et al. "Dion: Distributed Orthonormalized Updates". In: *arXiv preprint: 2504.05295* (2025).

[38] Kwangjun Ahn, Noah Amsel, and John Langford. *Dion2: A Simple Method to Shrink Matrix in Muon*. 2025. arXiv: 2512.16928 [cs.LG]. URL: https://arxiv.org/abs/2512.16928.

[39] Kang An et al. *ASGO: Adaptive Structured Gradient Optimization*. 2025. arXiv: 2503.20762 [cs.LG]. URL: https://arxiv.org/abs/2503.20762.

[40] Liliang Ren et al. *Rethinking Language Model Scaling under Transferable Hypersphere Optimization*. 2026. arXiv: 2603.28743 [cs.LG]. URL: https://arxiv.org/abs/2603.28743.

[41] Noah Amsel et al. *The Polar Express: Optimal Matrix Sign Methods and Their Application to the Muon Algorithm*. 2026. arXiv: 2505.16932 [cs.LG]. URL: https://arxiv.org/abs/2505.16932.

[42] Ziyue Liu et al. *Muon^2: Boosting Muon via Adaptive Second-Moment Preconditioning*. 2026. arXiv: 2604.09967 [cs.LG]. URL: https://arxiv.org/abs/2604.09967.

[43] Ahmed Khaled et al. *MuonBP: Faster Muon via Block-Periodic Orthogonalization*. 2025. arXiv: 2510.16981 [cs.LG]. URL: https://arxiv.org/abs/2510.16981.

[44] Alexey Kravatskiy et al. *The Ky Fan Norms and Beyond: Dual Norms and Combinations for Matrix Optimization*. 2025. arXiv: 2512.09678 [math.OC]. URL: https://arxiv.org/abs/2512.09678.

[45] Jingyuan Liu et al. *Muon is Scalable for LLM Training*. 2025. arXiv: 2502.16982 [cs.LG]. URL: https://arxiv.org/abs/2502.16982.

[46] Kimi Team. *Kimi K2.5: Visual Agentic Intelligence*. 2026. arXiv: 2602.02276 [cs.CL]. URL: https://arxiv.org/abs/2602.02276.

[47] GLM-5-Team. *GLM-5: from Vibe Coding to Agentic Engineering*. 2026. arXiv: 2602.15763 [cs.LG]. URL: https://arxiv.org/abs/2602.15763.

[48] Varun Singh et al. *Arcee Trinity Large Technical Report*. 2026. arXiv: 2602.17004 [cs.LG]. URL: https://arxiv.org/abs/2602.17004.

[49] Levent Sagun et al. *Empirical Analysis of the Hessian of Over-Parametrized Neural Networks*. 2017. eprint: arXiv:1706.04454.

[50] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. "An Investigation into Neural Net Optimization via Hessian Eigenvalue Density". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2232–2241. URL: https://proceedings.mlr.press/v97/ghorbani19b.html.

[51] Damek Davis and Dmitriy Drusvyatskiy. *When do spectral gradient updates help in deep learning?* 2026. arXiv: 2512.04299 [cs.LG]. URL: https://arxiv.org/abs/2512.04299.

[52] Fabian Schaipp et al. "The surprising agreement between convex optimization theory and learning-rate scheduling for large model training". In: *arXiv preprint arXiv:2501.18965* (2025).

[53] Anders Krogh and John A. Hertz. "A simple weight decay can improve generalization". In: *Proceedings of the 5th International Conference on Neural Information Processing Systems*. NIPS'91. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1991, pp. 950–957. ISBN: 1558602224.

[54] Francesco D'Angelo et al. *Why Do We Need Weight Decay in Modern Deep Learning?* 2024. arXiv: 2310.04415 [cs.LG]. URL: https://arxiv.org/abs/2310.04415.

[55] Shikai Qiu et al. *Hyperparameter Transfer Enables Consistent Gains of Matrix-Preconditioned Optimizers Across Scales*. 2026. arXiv: 2512.05620 [cs.LG]. URL: https://arxiv.org/abs/2512.05620.

[56] Da Chang, Yongxiang Liu, and Ganzhao Yuan. *On the Convergence of Muon and Beyond*. 2026. arXiv: 2509.15816 [cs.LG]. URL: https://arxiv.org/abs/2509.15816.

[57] Wei Shen et al. *On the Convergence Analysis of Muon*. 2026. arXiv: 2505.23737 [stat.ML]. URL: https://arxiv.org/abs/2505.23737.

[58] Naoki Sato, Hiroki Naganuma, and Hideaki Iiduka. *Convergence Bound and Critical Batch Size of Muon Optimizer*. 2025. arXiv: 2507.01598 [cs.LG]. URL: https://arxiv.org/abs/2507.01598.

[59] Jiaxiang Li and Mingyi Hong. *A Note on the Convergence of Muon*. 2025. arXiv: 2502.02900 [math.OC]. URL: https://arxiv.org/abs/2502.02900.

[60] Jeremy Bernstein and Laker Newhouse. *Old Optimizer, New Norm: An Anthology*. 2024. arXiv: 2409.20325 [cs.LG]. URL: https://arxiv.org/abs/2409.20325.

[61] Thomas Pethick et al. *Training Deep Learning Models with Norm-Constrained LMOs*. 2025. arXiv: 2502.07529 [cs.LG]. URL: https://arxiv.org/abs/2502.07529.

[62] Xiangning Chen et al. "Symbolic discovery of optimization algorithms". In: *Advances in neural information processing systems* 36 (2023), pp. 49205–49233.

[63] Vineet Gupta, Tomer Koren, and Yoram Singer. *Shampoo: Preconditioned Stochastic Tensor Optimization*. 2018. arXiv: 1802.09568 [cs.LG]. URL: https://arxiv.org/abs/1802.09568.

[64] Priya Goyal et al. "Accurate, large minibatch sgd: Training imagenet in 1 hour". In: *arXiv preprint arXiv:1706.02677* (2017).

[65] Dayal Singh Kalra and Maissam Barkeshli. "Why warmup the learning rate? underlying mechanisms and improvements". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 111760–111801.

[66] Aaron Defazio and Konstantin Mishchenko. "Learning-rate-free learning by d-adaptation". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 7449–7479.

[67] Noam Shazeer et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer". In: *International Conference on Learning Representations*. 2017.

[68] Dmitry Lepikhin et al. "GShard: Scaling giant models with conditional computation and automatic sharding". In: *International Conference on Learning Representations*. 2021.

[69] William Fedus, Barret Zoph, and Noam Shazeer. "Switch Transformers: scaling to trillion parameter models with simple and efficient sparsity". In: *Journal of Machine Learning Research* 23.120 (2022), pp. 1–39.

[70] Twan van Laarhoven. *L2 Regularization versus Batch and Weight Normalization*. 2017. arXiv: 1706.05350 [cs.LG]. URL: https://arxiv.org/abs/1706.05350.

[71] Aaron Defazio. *Why Gradients Rapidly Increase Near the End of Training*. 2025. arXiv: 2506.02285 [cs.LG]. URL: https://arxiv.org/abs/2506.02285.

[72] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[73] Jianlin Su et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. arXiv: 2104.09864 [cs.CL]. URL: https://arxiv.org/abs/2104.09864.

[74] Biao Zhang and Rico Sennrich. *Root Mean Square Layer Normalization*. 2019. arXiv: 1910.07467 [cs.LG]. URL: https://arxiv.org/abs/1910.07467.

[75] Zhengyan Zhang et al. *ReLU^2 Wins: Discovering Efficient Activation Functions for Sparse LLMs*. 2024. arXiv: 2402.03804 [cs.LG]. URL: https://arxiv.org/abs/2402.03804.

[76] Andrej Karpathy. *NanoGPT*. https://github.com/karpathy/nanoGPT. 2022.

[77] Keller Jordan et al. *modded-nanogpt: Speedrunning the NanoGPT baseline*. 2024. URL: https://github.com/KellerJordan/modded-nanogpt.

[78] Guilherme Penedo et al. *The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale*. 2024. arXiv: 2406.17557 [cs.CL]. URL: https://arxiv.org/abs/2406.17557.

[79] Alec Radford et al. "Language models are unsupervised multitask learners". In: (2019).

[80] Jack Choquette et al. "NVIDIA A100 Tensor Core GPU: Performance and Innovation". In: *IEEE Micro* 41.2 (Mar. 2021), pp. 29–35. ISSN: 1937-4143. DOI: 10.1109/mm.2021.3061394. URL: http://dx.doi.org/10.1109/MM.2021.3061394.

[81] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. ASPLOS '24. ACM, Apr. 2024, pp. 929–947. DOI: 10.1145/3620665.3640366. URL: http://dx.doi.org/10.1145/3620665.3640366.

[82] Kaichao You et al. *How Does Learning Rate Decay Help Modern Neural Networks?* 2019. arXiv: 1908.01878 [cs.LG]. URL: https://arxiv.org/abs/1908.01878.

Technical Appendices and Supplementary Material for "Anytime Training with Schedule-Free Spectral Optimization"

In this appendix, we provide more thorough preliminaries and background information that supplement the main text. We also state the formal version of the informal theorems from the main text, as well as their proofs. This appendix is organized as follows:

- In Appendix A, we review metrized deep learning, and derive optimizers from norms on weight matrices.
- In Appendix B , we review the Schedule-free method. In particular, we discuss SF-SGD and its convergence bound, and the algorithm for SF-AdamW.
- In Appendix C proves in the detail the convergence of SF-Spectral Descent, for noisy Lipschitz smooth functions in both the Frobenius and spectral norm. In addition, we prove the lemmas used in the steady-state analysis.
- In Appendix D, we review the architecture of the language models used in our experiments and present the hyper-parameter selection procedure.
- In Appendix E, studies the effect of ablating momentum term for SF-NorMuon (setting $\mu = 0$), and ablating the row-normalization. Furthermore, we present a detailed comparison against NorMuon with cosine decay of learning rate.
- In Appendix F, we present a PyTorch implementation of SF-NorMuon algorithm.

A Optimizers from Weight-Space Matrix Geometry

For the reader's convenience, we collect here the basic derivations of the update rules discussed in the main text. The starting point is always the same: given a current iterate W_t and gradient $G_t := \nabla f(W_t)$, we choose an update ΔW_t by minimizing the first-order model of the loss subject to a constraint, or equivalently a quadratic penalty, in the geometry of interest. Our treatment follows the work of Bernstein and Newhouse [60].

A.1 Steepest descent in a prescribed geometry

Let $\|\cdot\|_{\mathcal{X}}$ be any norm on the space of matrices. The corresponding norm-constrained steepest-descent step is

$$\Delta W_t \in \arg\min_{\|\Delta W\|_{\mathcal{X}} \leqslant \eta} \langle G_t, \Delta W \rangle. \tag{22}$$

Since the objective is linear, the solution lies on the boundary of the ball and points in an extremal descent direction for the chosen norm. Writing

$$\|G_t\|_{\mathcal{X},*} := \sup_{\|U\|_{\mathcal{X}} \leqslant 1} \langle G_t, U \rangle \tag{23}$$

for the dual norm, (22) is equivalent to

$$\Delta W_t = -\eta\, U_t, \qquad U_t \in \arg\max_{\|U\|_{\mathcal{X}} \leqslant 1} \langle G_t, U \rangle, \tag{24}$$

and the optimal value is $-\eta \|G_t\|_{\mathcal{X},*}$.

In practice one often uses the equivalent regularized local model

$$\Delta W_t \in \arg\min_{\Delta W} \left\{ \langle G_t, \Delta W \rangle + \frac{1}{2\eta} \|\Delta W\|_{\mathcal{X},t}^2 \right\}, \tag{25}$$

where $\|\cdot\|_{\mathcal{X},t}$ may depend on time through a momentum estimate or a preconditioner. The constrained form (22) produces normalized directions such as sign and polar updates, while the regularized form (25) produces the preconditioned-gradient forms used by practical optimizers.

A.2 The sign geometry: $p = 1$, $q = \infty$

Take the matrix norm

$$\|A\|_{\ell_1 \to \ell_\infty} = \sup_{\|u\|_1 \leqslant 1} \|Au\|_\infty. \tag{26}$$

A direct calculation shows that this is simply the entrywise max norm:

$$\|A\|_{\ell_1 \to \ell_\infty} = \max_{i,j} |A_{ij}|. \tag{27}$$

Indeed, for each row i,

$$|(Au)_i| = \left| \sum_j A_{ij} u_j \right| \leqslant \max_j |A_{ij}| \sum_j |u_j| \leqslant \max_j |A_{ij}|, \tag{28}$$

so $\|Au\|_\infty \leqslant \max_{i,j} |A_{ij}|$. Equality is achieved by choosing u to be a signed coordinate vector supported on an entry attaining the maximum.

Substituting (27) into (22) gives

$$\min_{\max_{i,j} |\Delta W_{ij}| \leqslant \eta} \sum_{i,j} (G_t)_{ij} (\Delta W)_{ij}. \tag{29}$$

This decouples coordinatewise, and for each entry the minimizer is

$$(\Delta W_t)_{ij} = -\eta \operatorname{sign}\big((G_t)_{ij}\big). \tag{30}$$

Hence the steepest-descent rule is

$$\Delta W_t = -\eta \operatorname{sign}(G_t), \tag{31}$$

which is the base update underlying signSGD:

$$W_{t+1} = W_t - \eta \operatorname{sign}(G_t). \tag{32}$$

A.3 Momentum in the sign geometry

To overcome the noise from the computing the gradient on a mini-batch a standard practical modification is to replace the raw gradient by a smoothed momentum estimate. Let

$$M_t = \beta M_{t-1} + (1 - \beta) G_t, \qquad 0 \leqslant \beta < 1. \tag{33}$$

Applying the same steepest-descent rule to M_t instead of G_t yields

$$W_{t+1} = W_t - \eta \operatorname{sign}(M_t), \tag{34}$$

which is the update for signSGD with momentum.

Lion. Lion uses a sign update applied to a momentum direction that directly incorporates current gradient as well. In the standard implementation, one first forms

$$C_t = \beta_1 M_{t-1} + (1-\beta_1) G_t, \tag{35}$$

then updates

$$W_{t+1} = W_t - \eta \operatorname{sign}(C_t), \tag{36}$$

and finally refreshes the state

$$M_t = \beta_2 M_{t-1} + (1-\beta_2) G_t. \tag{37}$$

Thus Lion can be viewed as a two-timescale version of the basic sign steepest-descent rule.

Adam with $\beta_2 = 0$. Adam augments the gradient with both a momentum buffer and a variance buffer:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) G_t, \qquad v_t = \beta_2 v_{t-1} + (1-\beta_2) G_t^{\odot 2}, \tag{38}$$

where $G_t^{\odot 2}$ denotes the entrywise square. Its coordinatewise update is

$$W_{t+1} = W_t - \eta \, \frac{m_t}{\sqrt{v_t} + \varepsilon}. \tag{39}$$

When $\beta_2 = 0$, the variance buffer collapses to the current squared gradient,

$$v_t = G_t^{\odot 2}, \tag{40}$$

and so

$$W_{t+1} = W_t - \eta \operatorname{sign}(M_t). \tag{41}$$

Thus, Adam with $\beta_2 = 0$ keeps the momentum buffer but normalizes it coordinatewise by the current gradient magnitude. This leads us back to signSGD with momentum.

A.4 The spectral geometry: $p = q = 2$

Now take the matrix norm

$$\|A\|_{\ell_2 \to \ell_2} = \|A\|_{\mathrm{op}}, \tag{42}$$

the spectral norm. The norm-constrained step becomes

$$\Delta W_t \in \arg\min_{\|\Delta W\|_{\mathrm{op}} \leqslant \eta} \langle G_t, \Delta W \rangle. \tag{43}$$

Let the singular value decomposition of G_t be

$$G_t = U_t \Sigma_t V_t^\top. \tag{44}$$

By von Neumann's trace inequality,

$$\langle G_t, U \rangle \leqslant \|G_t\|_* \|U\|_{\mathrm{op}} \qquad \text{for all } U, \tag{45}$$

so over the unit spectral-norm ball the maximum of $\langle G_t, U \rangle$ is $\|G_t\|_*$, attained by

$$U = U_t V_t^\top. \tag{46}$$

This matrix is the polar factor of G_t, which we denote by $\operatorname{polar}(G_t)$. Therefore

$$\Delta W_t = -\eta \operatorname{polar}(G_t), \tag{47}$$

and the corresponding update for spectral descent is

$$W_{t+1} = W_t - \eta \operatorname{polar}(G_t). \tag{48}$$

A.5 Momentum in the spectral geometry

Muon. Exactly as in the sign case, we may replace G_t by a momentum-smoothed matrix

$$M_t = \beta M_{t-1} + (1-\beta)G_t. \tag{49}$$

Applying spectral steepest descent to M_t gives the update for Muon

$$W_{t+1} = W_t - \eta\,\mathrm{polar}(M_t). \tag{50}$$

In practical implementations, this polar factor is not computed by an exact SVD but is approximated by a small number of Newton-Schulz iterations. This minimizes the computational overhead as they can be implemented using only matrix multiplications (GEMMs).

Shampoo. Shampoo maintains left and right second-moment buffers for each matrix parameter and updates by preconditioning with their inverse quarter-powers

$$M_t = \beta_1 M_{t-1} + (1-\beta_1)G_t, \qquad L_t = \beta_2 L_{t-1} + (1-\beta_2)M_t M_t^\top, \qquad R_t = \beta_2 R_{t-1} + (1-\beta_2)M_t^\top M_t, \tag{51}$$

followed by the update

$$W_{t+1} = W_t - \eta\, L_t^{-1/4} M_t R_t^{-1/4}. \tag{52}$$

In the limiting case $\beta_2 = 0$, these buffers collapse to the instantaneous matrices

$$L_t = M_t M_t^\top, \qquad R_t = M_t^\top M_t. \tag{53}$$

If $M_t = U\Sigma V^\top$ is the singular value decomposition, then

$$L_t^{-1/4} M_t R_t^{-1/4} = U\Sigma^{-1/2}U^\top\,(U\Sigma V^\top)\,V\Sigma^{-1/2}V^\top = UV^\top = \mathrm{polar}(M_t), \tag{54}$$

where the identity is exact in the idealized full-rank case (and otherwise understood with pseudo-inverses on the support of M_t). Hence, when the second-moment buffers are collapsed in this way, Shampoo reduces to the same polar transform of the momentum matrix that underlies spectral descent and Muon.

B Review of SF-SGD and SF-AdamW

Schedule-free (SF) methods were introduced to remove the need for a prescribed training horizon in learning-rate scheduling. The basic idea is to keep a *fast* sequence z_t, on which the base optimizer runs at a nearly constant learning rate, while returning an *averaged* sequence x_t whose effective learning rate decays automatically through online averaging. Gradients are evaluated at an interpolation point y_t between these two states. In this way, schedule-free optimization replaces explicit time-based annealing by implicit annealing through online weight averaging [17].

SF-SGD. In its simplest form, Schedule-Free SGD maintains three sequences,

$$y_t = \beta x_t + (1-\beta)z_t, \qquad g_t = \nabla\mathcal{L}(y_t;\xi_t), \qquad z_{t+1} = z_t - \eta g_t, \tag{55}$$

together with the online average

$$x_{t+1} = (1-c_{t+1})x_t + c_{t+1}z_{t+1}, \qquad c_{t+1} = \frac{1}{t+1}. \tag{56}$$

Here z_t is the base iterate, x_t is the returned iterate, and y_t is the point at which the stochastic gradient is computed. The parameter $\beta \in [0,1]$ interpolates between Polyak-Ruppert averaging ($\beta = 0$) and primal averaging ($\beta = 1$).

Implicit decay of the effective learning rate. The averaging update can be rewritten as

$$x_{t+1} - x_t = c_{t+1}(z_{t+1} - x_t) = c_{t+1}(z_t - x_t) - c_{t+1}\eta g_t. \tag{57}$$

Using

$$y_t = \beta x_t + (1-\beta)z_t \qquad \Longrightarrow \qquad z_t - x_t = \frac{y_t - x_t}{1-\beta}, \tag{58}$$

we obtain

$$x_{t+1} - x_t = \frac{c_{t+1}}{1-\beta}(y_t - x_t) - c_{t+1}\eta g_t. \tag{59}$$

Thus the returned iterate x_t moves as though it were taking a gradient step with *effective* learning rate

$$\eta_t^{\text{eff}} \approx c_{t+1}\eta = \frac{\eta}{t+1}. \tag{60}$$

Thus, the effective learning rate decays like $1/t$ even though the base update on z_t uses a constant step size. This is the basic mechanism by which schedule-free methods replace an explicit schedule. In the convex Lipschitz setting, this implicit learning rate decay is sufficient to retain the optimal worst-case convergence rate of $O(T^{-1/2})$. For a convex G-Lipschitz function with $D = \|x_1 - x_\star\|$, the choice of $\eta = D/(G\sqrt{T})$ for any $\beta \in [0,1]$ leads to

$$\mathbb{E}\big[F(x_T) - F(x_\star)\big] \leqslant \frac{DG}{\sqrt{T}}. \tag{61}$$

SF-AdamW. For deep-learning applications, the raw gradient is replaced by a preconditioned update together with warmup and decoupled weight decay.

Algorithm 2 Schedule-Free AdamW

Input: base learning rate η, interpolation parameter β_1, variance parameter β_2, weight decay λ, warmup steps T_{warmup}, numerical constant ε, decay location $\tilde{y}_t \in \{y_t, z_t\}$
for $t = 1, 2, \ldots, T$ **do**
  $y_t \leftarrow (1-\beta_1)z_t + \beta_1 x_t$ ▷ Momentum via interpolation
  $g_t \leftarrow \nabla\mathcal{L}(y_t; \xi_t)$ ▷ Gradient is evaluated at y_t
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^{\odot 2}$
  $\eta_t \leftarrow \eta\sqrt{1-\beta_2^t}\,\min(1, t/T_{\text{warmup}})$ ▷ Warmup and Adam bias-correction
  $z_{t+1} \leftarrow z_t - \eta_t\, g_t/(\sqrt{v_t} + \varepsilon) - \eta_t\lambda\tilde{y}_t$
  $s_t \leftarrow s_{t-1} + \eta_t^2$
  $c_{t+1} \leftarrow \eta_t^2/s_t$
  $x_{t+1} \leftarrow (1-c_{t+1})x_t + c_{t+1}z_{t+1}$ ▷ Update weighted iterate average
end for
return x_T

The choice $\tilde{y}_t = y_t$ corresponds to applying weight decay at the gradient location, which matches the interpretation of decay as an ℓ_2-regularizer in the loss, while $\tilde{y}_t = z_t$ applies decay at the fast iterate. A useful practical feature is that SF-AdamW does not require more memory than AdamW. One stores z_t and y_t in memory, and reconstructs

$$x_t = \frac{y_t - (1-\beta_1)z_t}{\beta_1} \tag{62}$$

when evaluation is needed. Note that unlike classical momentum, which introduces a separate momentum buffer, schedule-free momentum is implemented through the interpolation $y_t = (1-\beta)z_t + \beta x_t$ in weight space.

C Convergence of Schedule-Free Spectral optimization

We first state the iterations of the schedule-free spectral optimization and the assumptions of Lipschitz smoothness and bounded diameter.

Iterates. For parameters $0 \leqslant \beta < 1$, $0 \leqslant \mu < 1$, stepsize $\eta > 0$, and batch size $B \geqslant 1$, consider

$$Y_t = \beta X_t + (1-\beta) Z_t, \tag{63}$$

$$G_t = \frac{1}{B} \sum_{i=1}^{B} \nabla f(Y_t; \xi_{t,i}), \tag{64}$$

$$M_0 = G_0, \qquad M_t = \mu M_{t-1} + (1-\mu) G_t \quad (t \geqslant 1), \tag{65}$$

$$P_t = \mathrm{polar}(M_t), \tag{66}$$

$$Z_{t+1} = Z_t - \eta P_t, \tag{67}$$

$$X_{t+1} = \frac{t}{t+1} X_t + \frac{1}{t+1} Z_{t+1}, \qquad X_0 = Z_0. \tag{68}$$

Assume there exists $W^\star \in \mathbb{R}^{m \times n}$ such that

$$f(W^\star) = f^\star, \qquad \nabla f(W^\star) = 0,$$

and let

$$r := \min\{m, n\}, \qquad \Delta := f(Y_0) - f^\star.$$

Throughout, the stochastic oracle is unbiased with bounded Frobenius variance:

$$\mathbb{E}[G_t \mid \mathcal{F}_t] = \nabla f(Y_t) =: g_t, \qquad \mathbb{E}\big[\|G_t - g_t\|_F^2 \mid \mathcal{F}_t\big] \leqslant \frac{\sigma^2}{B}. \tag{69}$$

Smoothness assumptions. For the spectral proof, we assume there exists a constant L_S such that

$$\|\nabla f(A) - \nabla f(B)\|_* \leqslant L_S \|A - B\|_{\mathrm{op}} \qquad \text{for all } A, B \in \mathbb{R}^{m \times n}. \tag{70}$$

For the Frobenius proof, we assume there exists a constant L_F such that

$$\|\nabla f(A) - \nabla f(B)\|_F \leqslant L_F \|A - B\|_F \qquad \text{for all } A, B \in \mathbb{R}^{m \times n}. \tag{71}$$

Bounded diameter assumptions. For the spectral proof, we assume there exists a constant $D_S > 0$ such that

$$\|W^\star\|_{\mathrm{op}} \leqslant D_S/2, \qquad \|Z_t\|_{\mathrm{op}} \leqslant D_S/2 \qquad \text{for all } t, \tag{72}$$

while for the Frobenius proof, we assume there exists a constant $D_F > 0$ such that

$$\|W^\star\|_F \leqslant D_F/2, \qquad \|Z_t\|_F \leqslant D_F/2 \qquad \text{for all } t. \tag{73}$$

Consequently, $\|X_t\|_{\mathrm{op}}, \|Y_t\|_{\mathrm{op}} \leqslant D_S/2$ for all t, and hence

$$\|Y_t - W^\star\|_{\mathrm{op}} \leqslant D_S, \qquad \|Z_t - W^\star\|_{\mathrm{op}} \leqslant D_S \qquad \text{for all } t,$$

and a similar statement holds for the Frobenius norm.

These are standard bounded-iterate assumptions (see Theorem 7 in [63] and Theorem 1 in [39]), and they can be enforced algorithmically by considering a projected variant that keep the fast iterates Z_t inside the corresponding norm ball. The iterates X, Y are unweighted and weighted averages of current and previous Z iterates and hence remain within the ball.

C.1 Reusable identities and bounds

Lemma C.1 (Iterate identities)**.** *Define*

$$S_t := Z_t - Y_t.$$

Then $S_t = \beta(Z_t - X_t)$*, and for every* $t \geqslant 0$*,*

$$S_{t+1} = \frac{t}{t+1}\big(S_t - \beta\eta P_t\big), \tag{74}$$

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)P_t. \tag{75}$$

Consequently, if $\|\cdot\|_\diamond$ *is any norm for which* $\|Y_t - W^\star\|_\diamond \leqslant D_\diamond$ *and* $\|Z_t - W^\star\|_\diamond \leqslant D_\diamond$*, then*

$$\|S_t\|_\diamond \leqslant 2D_\diamond, \qquad \|Y_{t+1} - Y_t\|_\diamond \leqslant \frac{2D_\diamond}{t+1} + \eta\|P_t\|_\diamond. \tag{76}$$

In particular,

$$\|Y_{t+1} - Y_t\|_{\mathrm{op}} \leqslant \frac{2D_S}{t+1} + \eta, \tag{77}$$

$$\|Y_{t+1} - Y_t\|_F \leqslant \frac{2D_F}{t+1} + \eta\sqrt{r}. \tag{78}$$

Proof. Since $Y_t = \beta X_t + (1-\beta)Z_t$, we have

$$S_t = Z_t - Y_t = \beta(Z_t - X_t).$$

Using (67) and (68),

$$\begin{aligned}
S_{t+1} &= Z_{t+1} - Y_{t+1} \\
&= \beta(Z_{t+1} - X_{t+1}) \\
&= \beta\left(Z_{t+1} - \frac{t}{t+1}X_t - \frac{1}{t+1}Z_{t+1}\right) \\
&= \frac{t}{t+1}\beta(Z_{t+1} - X_t) \\
&= \frac{t}{t+1}\beta\big((Z_t - \eta P_t) - X_t\big) \\
&= \frac{t}{t+1}\big(S_t - \beta\eta P_t\big),
\end{aligned}$$

which proves (74). Then

$$\begin{aligned}
Y_{t+1} - Y_t &= (Z_{t+1} - S_{t+1}) - (Z_t - S_t) \\
&= -\eta P_t - S_{t+1} + S_t \\
&= \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)P_t,
\end{aligned}$$

which is (75). The norm bounds follow from

$$\|S_t\|_\diamond = \|Z_t - Y_t\|_\diamond \leqslant \|Z_t - W^\star\|_\diamond + \|Y_t - W^\star\|_\diamond \leqslant 2D_\diamond,$$

together with $\|P_t\|_{\mathrm{op}} \leqslant 1$ and $\|P_t\|_F \leqslant \sqrt{r}$. □

Lemma C.2 (Polar alignment)**.** *For every* $t \geqslant 0$*,*

$$\langle g_t, P_t\rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*. \tag{79}$$

Proof. Since $P_t = \mathrm{polar}(M_t)$,

$$\langle M_t, P_t \rangle = \|M_t\|_*.$$

Also,

$$\langle g_t - M_t, P_t \rangle \geqslant -\|g_t - M_t\|_* \|P_t\|_{\mathrm{op}} \geqslant -\|g_t - M_t\|_*.$$

Therefore

$$\langle g_t, P_t \rangle = \langle M_t, P_t \rangle + \langle g_t - M_t, P_t \rangle \geqslant \|M_t\|_* - \|g_t - M_t\|_*.$$

Finally,

$$\|M_t\|_* \geqslant \|g_t\|_* - \|g_t - M_t\|_*,$$

which yields (79). □

Lemma C.3 (Stochastic EMA error)**.** *Let*

$$C_0 = g_0, \qquad C_t = \mu C_{t-1} + (1-\mu) g_t, \qquad E_t := C_t - M_t.$$

Then

$$E_t = \mu E_{t-1} + (1-\mu)(g_t - G_t), \tag{80}$$

and

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|E_t\|_* \leqslant \sqrt{r} \left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}} \right). \tag{81}$$

Proof. The recursion (80) is immediate from the definitions. Using (69),

$$\mathbb{E}\big[\|E_t\|_F^2 \mid \mathcal{F}_t\big] \leqslant \mu^2 \|E_{t-1}\|_F^2 + (1-\mu)^2 \frac{\sigma^2}{B}.$$

Solving this recursion and applying Jensen gives

$$\mathbb{E}\|E_t\|_F \leqslant \mu^t \frac{\sigma}{\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}}.$$

Since $\|A\|_* \leqslant \sqrt{r}\,\|A\|_F$, averaging over $t = 0, \ldots, T-1$ yields (81). □

Lemma C.4 (Tracking bound under spectral smoothness)**.** *Assume* (70) *and* (72)*. Then*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|g_t - M_t\|_* \leqslant \frac{\mu L_S \eta}{1-\mu} + \frac{2\mu L_S D_S \log(eT)}{(1-\mu)T} + \sqrt{r} \left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}} \frac{\sigma}{\sqrt{B}} \right). \tag{82}$$

Proof. Write

$$g_t - M_t = (g_t - C_t) + (C_t - M_t), \qquad A_t := \|g_t - C_t\|_*.$$

Then

$$\begin{aligned} A_t &= \|g_t - \mu C_{t-1} - (1-\mu) g_t\|_* \\ &= \mu \|g_t - C_{t-1}\|_* \\ &\leqslant \mu \|g_t - g_{t-1}\|_* + \mu A_{t-1} \\ &\leqslant \mu L_S \|Y_t - Y_{t-1}\|_{\mathrm{op}} + \mu A_{t-1} \\ &\leqslant \mu L_S \left(\frac{2D_S}{t} + \eta \right) + \mu A_{t-1}, \end{aligned}$$

where we used (77). Iterating and averaging gives

$$\frac{1}{T} \sum_{t=0}^{T-1} A_t \leqslant \frac{\mu L_S \eta}{1-\mu} + \frac{2\mu L_S D_S \log(eT)}{(1-\mu)T}.$$

Combining this with Lemma C.3 proves (82). □

Lemma C.5 (Tracking bound under Frobenius smoothness)**.** *Assume* (71) *and* (73)*. Then*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|g_t - M_t\|_* \leqslant \frac{\mu r L_F \eta}{1-\mu} + \frac{2\mu\sqrt{r}\,L_F D_F \log(eT)}{(1-\mu)T} + \sqrt{r}\left(\frac{\sigma}{(1-\mu)T\sqrt{B}} + \sqrt{\frac{1-\mu}{1+\mu}}\frac{\sigma}{\sqrt{B}}\right). \tag{83}$$

Proof. Again write

$$g_t - M_t = (g_t - C_t) + (C_t - M_t), \qquad A_t := \|g_t - C_t\|_*.$$

Then

$$\begin{aligned} A_t &\leqslant \mu\|g_t - g_{t-1}\|_* + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\,\|g_t - g_{t-1}\|_F + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\,L_F\|Y_t - Y_{t-1}\|_F + \mu A_{t-1} \\ &\leqslant \mu\sqrt{r}\,L_F\left(\frac{2D_F}{t} + \eta\sqrt{r}\right) + \mu A_{t-1}, \end{aligned}$$

where we used (78). Iterating and averaging gives

$$\frac{1}{T}\sum_{t=0}^{T-1} A_t \leqslant \frac{\mu r L_F \eta}{1-\mu} + \frac{2\mu\sqrt{r}\,L_F D_F \log(eT)}{(1-\mu)T}.$$

Combining with Lemma C.3 proves (83). □

C.2 Spectral-smooth analysis

Theorem C.1 (Stationarity at the Y-iterates under spectral smoothness)**.** *Assume* (69)*,* (70)*, and* (72)*. Then for every $T \geqslant 1$,*

$$\begin{aligned} \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant{}& \frac{\Delta + 2L_S D_S^2\left(\log(eT) + \frac{\pi^2}{6}\right)}{(1-\beta)T\eta} + \frac{L_S\eta}{2(1-\beta)} + \frac{2\mu L_S\eta}{(1-\beta)(1-\mu)} \\ &+ \frac{2L_S D_S\log(eT)}{(1-\beta)T}\left(1 + \frac{2\mu}{1-\mu}\right) + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} \\ &+ \frac{2\sqrt{r}\,\sigma}{(1-\beta)(1-\mu)T\sqrt{B}}. \end{aligned} \tag{84}$$

Proof. By (70),

$$f(Y_{t+1}) \leqslant f(Y_t) + \langle g_t, Y_{t+1} - Y_t\rangle + \frac{L_S}{2}\|Y_{t+1} - Y_t\|_{\mathrm{op}}^2.$$

Using Lemma C.1,

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)P_t,$$

hence

$$f(Y_{t+1}) \leqslant f(Y_t) + \frac{1}{t+1}\langle g_t, S_t\rangle - \eta\left(1 - \beta + \frac{\beta}{t+1}\right)\langle g_t, P_t\rangle + \frac{L_S}{2}\|Y_{t+1} - Y_t\|_{\mathrm{op}}^2.$$

Now $\|g_t\|_* \leqslant L_S D_S$, $\|S_t\|_{\mathrm{op}} \leqslant 2D_S$, $\|Y_{t+1} - Y_t\|_{\mathrm{op}} \leqslant 2D_S/(t+1) + \eta$, and Lemma C.2 gives

$$\langle g_t, P_t\rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*.$$

Therefore

$$f(Y_{t+1}) \leqslant f(Y_t) - \eta(1-\beta)\|g_t\|_* + 2\eta\|g_t - M_t\|_* + \frac{2L_S D_S^2}{t+1} + \frac{L_S}{2}\left(\frac{2D_S}{t+1} + \eta\right)^2. \tag{85}$$

Summing (85) from $t = 0$ to $T - 1$, taking expectations, and using $f(Y_T) \geqslant f^\star$, we obtain

$$\eta(1-\beta)\sum_{t=0}^{T-1}\mathbb{E}\|g_t\|_* \leqslant \Delta + 2\eta\sum_{t=0}^{T-1}\mathbb{E}\|g_t - M_t\|_* + 2L_S D_S^2\sum_{t=0}^{T-1}\frac{1}{t+1} + \frac{L_S}{2}\sum_{t=0}^{T-1}\left(\frac{2D_S}{t+1}+\eta\right)^2.$$

Using

$$\sum_{t=0}^{T-1}\frac{1}{t+1} \leqslant \log(eT), \qquad \sum_{t=0}^{T-1}\frac{1}{(t+1)^2} \leqslant \frac{\pi^2}{6},$$

and substituting Lemma C.4 yields (84). □

C.3 Frobenius-smooth analysis

Theorem C.2 (Stationarity at the Y-iterates under Frobenius smoothness)**.** *Assume* (69), (71), *and* (73). *Then for every* $T \geqslant 1$,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta + 2L_F D_F^2\left(\log(eT)+\frac{\pi^2}{6}\right)}{(1-\beta)T\eta} + \frac{L_F r\,\eta}{2(1-\beta)} + \frac{2\mu L_F r\,\eta}{(1-\beta)(1-\mu)} + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{(1-\beta)T}\left(1+\frac{2\mu}{1-\mu}\right) + \frac{2\sqrt{r}\,\sigma}{(1-\beta)\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\beta)(1-\mu)T\sqrt{B}}. \tag{86}$$

Proof. By (71),

$$f(Y_{t+1}) \leqslant f(Y_t) + \langle g_t, Y_{t+1} - Y_t\rangle + \frac{L_F}{2}\|Y_{t+1} - Y_t\|_F^2.$$

Using Lemma C.1,

$$Y_{t+1} - Y_t = \frac{S_t}{t+1} - \eta\left(1-\beta+\frac{\beta}{t+1}\right)P_t.$$

Hence

$$f(Y_{t+1}) \leqslant f(Y_t) + \frac{1}{t+1}\langle g_t, S_t\rangle - \eta\left(1-\beta+\frac{\beta}{t+1}\right)\langle g_t, P_t\rangle + \frac{L_F}{2}\|Y_{t+1} - Y_t\|_F^2.$$

Since $\|g_t\|_F \leqslant L_F D_F$, $\|S_t\|_F \leqslant 2D_F$, $\|Y_{t+1} - Y_t\|_F \leqslant 2D_F/(t+1) + \eta\sqrt{r}$, and Lemma C.2 gives

$$\langle g_t, P_t\rangle \geqslant \|g_t\|_* - 2\|g_t - M_t\|_*,$$

we obtain

$$f(Y_{t+1}) \leqslant f(Y_t) - \eta(1-\beta)\|g_t\|_* + 2\eta\|g_t - M_t\|_* + \frac{2L_F D_F^2}{t+1} + \frac{L_F}{2}\left(\frac{2D_F}{t+1}+\eta\sqrt{r}\right)^2. \tag{87}$$

Summing (87) from $t = 0$ to $T - 1$, taking expectations, and using $f(Y_T) \geqslant f^\star$, we get

$$\eta(1-\beta)\sum_{t=0}^{T-1}\mathbb{E}\|g_t\|_* \leqslant \Delta + 2\eta\sum_{t=0}^{T-1}\mathbb{E}\|g_t - M_t\|_* + 2L_F D_F^2\sum_{t=0}^{T-1}\frac{1}{t+1} + \frac{L_F}{2}\sum_{t=0}^{T-1}\left(\frac{2D_F}{t+1}+\eta\sqrt{r}\right)^2.$$

Using

$$\sum_{t=0}^{T-1} \frac{1}{t+1} \leqslant \log(eT), \qquad \sum_{t=0}^{T-1} \frac{1}{(t+1)^2} \leqslant \frac{\pi^2}{6},$$

and substituting Lemma C.5 yields (86). □

C.4 Corollaries with optimized hyperparameters

Corollary C.1 (Tuned rates under spectral smoothness)**.** *Assume the hypotheses of Theorem C.1, and fix* $\beta \in [0, 1)$.

(i) ***Noisy case.*** *There exist choices* $\mu_T \in [0, 1)$ *and* $\eta_T > 0$ *such that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right). \tag{88}$$

(ii) ***Noiseless case.*** *If* $\sigma = 0$*, then there exist choices* $\mu_T \in [0, 1)$ *and* $\eta_T > 0$ *such that*

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right). \tag{89}$$

(iii) ***Log-free specialization when*** $\beta = 0$**.** *If* $\beta = 0$*, then the schedule-free drift disappears, and the rates improve to*

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(Y_t)\|_* = O\left(T^{-1/4}\right) \qquad \textit{in the noisy case,} \tag{90}$$

and

$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla f(Y_t)\|_* = O\left(T^{-1/2}\right) \qquad \textit{in the noiseless case.} \tag{91}$$

Proof. Let

$$A_{T,S} := \Delta + 2L_S D_S^2 \left(\log(eT) + \frac{\pi^2}{6}\right).$$

Then $A_{T,S} = O(\log T)$.

For part (i), write

$$\alpha := 1 - \mu \in (0, 1].$$

Then Theorem C.1 becomes

$$\frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta} \left[\frac{A_{T,S}}{T\eta} + \frac{L_S(4-3\alpha)}{2\alpha}\eta + \frac{2L_S D_S \log(eT)}{T} \frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}} \sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}} \right]. \tag{92}$$

For fixed α, the exact optimizer in η is

$$\eta^\star_{T,S}(\alpha) = \sqrt{\frac{2A_{T,S}\alpha}{L_S T(4-3\alpha)}}. \tag{93}$$

Substituting (93) into (92) gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\Bigg[\sqrt{\frac{2A_{T,S}L_S}{T}\cdot\frac{4-3\alpha}{\alpha}}+\frac{2L_SD_S\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}}+\frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\Bigg]. \tag{94}$$

Choose

$$\alpha_{T,S} := \min\left\{1,\ 2\sqrt{\frac{A_{T,S}L_SB}{r\sigma^2T}}\right\}, \qquad \mu_{T,S} := 1-\alpha_{T,S}, \qquad \eta_{T,S} := \eta^\star_{T,S}(\alpha_{T,S}). \tag{95}$$

This is the leading-order optimal choice obtained by balancing the first and third terms in (94). With this choice,

$$\sqrt{\frac{2A_{T,S}L_S}{T}\cdot\frac{4-3\alpha_{T,S}}{\alpha_{T,S}}} = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right),$$

and

$$\frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha_{T,S}}{2-\alpha_{T,S}}} = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right).$$

The remaining terms are lower order:

$$\frac{2L_SD_S\log(eT)}{T}\frac{2-\alpha_{T,S}}{\alpha_{T,S}} = O\left(\sqrt{\frac{\log T}{T}}\right), \qquad \frac{2\sqrt{r}\,\sigma}{\alpha_{T,S}T\sqrt{B}} = O\left(\frac{1}{\sqrt{T\log T}}\right).$$

Hence

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{A_{T,S}}{T}\right)^{1/4}\right) = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right),$$

which proves part (i).

For part (ii), when $\sigma = 0$, the stochastic terms vanish from (92). The remaining bound is increasing in μ, hence minimized by

$$\mu_{T,S} = 0 \qquad \text{equivalently} \qquad \alpha_{T,S} = 1.$$

Then the exact optimizer in η is

$$\eta_{T,S} = \sqrt{\frac{2A_{T,S}}{L_ST}}. \tag{96}$$

Substituting $\mu_{T,S} = 0$ and (96) into Theorem C.1 gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2L_SA_{T,S}}{T}}+\frac{2L_SD_S\log(eT)}{T}\right].$$

Since $A_{T,S} = O(\log T)$, this implies

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right),$$

which proves part (ii).

For part (iii), suppose $\beta = 0$. Then $Y_t = Z_t$, hence

$$S_t = Z_t - Y_t = 0 \qquad \text{for all } t,$$

and therefore

$$Y_{t+1} - Y_t = -\eta P_t.$$

Thus the harmonic-drift term disappears from both the descent and tracking arguments. Repeating the proof of Theorem C.1 with $S_t \equiv 0$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S\eta}{2} + \frac{2\mu L_S\eta}{1-\mu} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\mu)T\sqrt{B}}. \tag{97}$$

Writing $\alpha := 1-\mu$, this becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}.$$

For fixed α, the exact optimizer in η is

$$\eta^\star = \sqrt{\frac{2\Delta\,\alpha}{L_S T(4-3\alpha)}}.$$

Substituting this back and balancing the first two leading terms gives

$$\alpha_T \asymp \min\left\{1,\ 2\sqrt{\frac{\Delta L_S B}{r\sigma^2 T}}\right\}, \qquad \mu_T := 1-\alpha_T, \qquad \eta_T := \sqrt{\frac{2\Delta\,\alpha_T}{L_S T(4-3\alpha_T)}}.$$

With this choice,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O(T^{-1/4}),$$

which proves the noisy claim in (90).

If in addition $\sigma = 0$, then (97) reduces to

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_S\eta}{2}.$$

Its exact optimizer is

$$\mu_T = 0, \qquad \eta_T = \sqrt{\frac{2\Delta}{L_S T}}.$$

Substituting yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \sqrt{\frac{2L_S\Delta}{T}} = O(T^{-1/2}),$$

which proves (91). □

Corollary C.2 (Tuned rates under Frobenius smoothness)**.** *Assume the hypotheses of Theorem C.2, and fix* $\beta \in [0,1)$.

(i) ***Noisy case.*** *There exist choices* $\mu_T \in [0,1)$ *and* $\eta_T > 0$ *such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right). \tag{98}$$

(ii) ***Noiseless case.*** *If* $\sigma = 0$*, then there exist choices* $\mu_T \in [0,1)$ *and* $\eta_T > 0$ *such that*

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right). \tag{99}$$

(iii) ***Log-free specialization when*** $\beta = 0$**.** *If $\beta = 0$, then the schedule-free drift disappears, and the rates improve to*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\Big(T^{-1/4}\Big) \qquad \textit{in the noisy case,} \tag{100}$$

and

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\Big(T^{-1/2}\Big) \qquad \textit{in the noiseless case.} \tag{101}$$

Proof. Let

$$A_{T,F} := \Delta + 2L_F D_F^2\left(\log(eT) + \frac{\pi^2}{6}\right).$$

Then $A_{T,F} = O(\log T)$.

For part (i), write

$$\alpha := 1 - \mu \in (0, 1].$$

Then Theorem C.2 becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\frac{A_{T,F}}{T\eta} + \frac{L_F r(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\right]. \tag{102}$$

For fixed α, the exact optimizer in η is

$$\eta^\star_{T,F}(\alpha) = \sqrt{\frac{2A_{T,F}\alpha}{L_F r\,T(4-3\alpha)}}. \tag{103}$$

Substituting (103) into (102) gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2A_{T,F}L_F r}{T}\cdot\frac{4-3\alpha}{\alpha}} + \frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha}{\alpha} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}\right]. \tag{104}$$

Choose

$$\alpha_{T,F} := \min\left\{1,\ 2\sqrt{\frac{A_{T,F}L_F B}{\sigma^2 T}}\right\}, \qquad \mu_{T,F} := 1 - \alpha_{T,F}, \qquad \eta_{T,F} := \eta^\star_{T,F}(\alpha_{T,F}). \tag{105}$$

This is the leading-order optimal choice obtained by balancing the first and third terms in (104). With this choice,

$$\sqrt{\frac{2A_{T,F}L_F r}{T}\cdot\frac{4-3\alpha_{T,F}}{\alpha_{T,F}}} = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right),$$

and

$$\frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha_{T,F}}{2-\alpha_{T,F}}} = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right).$$

The remaining terms are lower order:

$$\frac{2\sqrt{r}\,L_F D_F\log(eT)}{T}\frac{2-\alpha_{T,F}}{\alpha_{T,F}} = O\left(\sqrt{\frac{\log T}{T}}\right), \qquad \frac{2\sqrt{r}\,\sigma}{\alpha_{T,F}T\sqrt{B}} = O\left(\frac{1}{\sqrt{T\log T}}\right).$$

Hence

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O\left(\left(\frac{A_{T,F}}{T}\right)^{1/4}\right) = O\left(\left(\frac{\log T}{T}\right)^{1/4}\right),$$

which proves part (i).

For part (ii), when $\sigma = 0$, the stochastic terms vanish from (102). The remaining bound is increasing in μ, hence minimized by

$$\mu_{T,F} = 0 \qquad \text{equivalently} \qquad \alpha_{T,F} = 1.$$

Then the exact optimizer in η is

$$\eta_{T,F} = \sqrt{\frac{2A_{T,F}}{L_F r\, T}}. \tag{106}$$

Substituting $\mu_{T,F} = 0$ and (106) into Theorem C.2 gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{1}{1-\beta}\left[\sqrt{\frac{2L_F r\, A_{T,F}}{T}} + \frac{2\sqrt{r}\, L_F D_F \log(eT)}{T}\right].$$

Since $A_{T,F} = O(\log T)$, this implies

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* = O\left(\sqrt{\frac{\log T}{T}}\right),$$

which proves part (ii).

For part (iii), suppose $\beta = 0$. Then $Y_t = Z_t$, so

$$S_t = 0 \qquad \text{and} \qquad Y_{t+1} - Y_t = -\eta P_t.$$

Thus the harmonic-drift term disappears from both the descent and tracking arguments. Repeating the proof of Theorem C.2 with $S_t \equiv 0$ gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r\,\eta}{2} + \frac{2\mu L_F r\,\eta}{1-\mu} + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{1-\mu}{1+\mu}} + \frac{2\sqrt{r}\,\sigma}{(1-\mu)T\sqrt{B}}. \tag{107}$$

Writing $\alpha := 1 - \mu$, this becomes

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r(4-3\alpha)}{2\alpha}\eta + \frac{2\sqrt{r}\,\sigma}{\sqrt{B}}\sqrt{\frac{\alpha}{2-\alpha}} + \frac{2\sqrt{r}\,\sigma}{\alpha T\sqrt{B}}.$$

For fixed α, the exact optimizer in η is

$$\eta^\star = \sqrt{\frac{2\Delta\,\alpha}{L_F r\, T(4-3\alpha)}}.$$

Substituting this back and balancing the first two leading terms gives

$$\alpha_T \asymp \min\left\{1,\ 2\sqrt{\frac{\Delta L_F B}{\sigma^2 T}}\right\}, \qquad \mu_T := 1 - \alpha_T, \qquad \eta_T := \sqrt{\frac{2\Delta\,\alpha_T}{L_F r\, T(4-3\alpha_T)}}.$$

With this choice,

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\|\nabla f(Y_t)\|_* = O(T^{-1/4}),$$

which proves (100).

If in addition $\sigma = 0$, then (107) reduces to

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \frac{\Delta}{T\eta} + \frac{L_F r\,\eta}{2}.$$

Its exact optimizer is

$$\mu_T = 0, \qquad \eta_T = \sqrt{\frac{2\Delta}{L_F r\,T}}.$$

Substituting yields

$$\frac{1}{T}\sum_{t=0}^{T-1}\|\nabla f(Y_t)\|_* \leqslant \sqrt{\frac{2L_F r\,\Delta}{T}} = O(T^{-1/2}),$$

which proves (101). □

C.5 Steady-State Analysis for Weight Decay at Z

In this subsection, we prove the boundedness and steady-state characterization lemmas stated in Section 3. For convenience, we restate the lemmas here.

C.5.1 Proof of Lemma 3.1

Lemma 3.1 (Restated). *Consider the update $Z_{t+1} \leftarrow Z_t - \eta\lambda Z_t - \hat{\eta}_t \widehat{P}_t$ and assume $0 < \eta\lambda < 1$ with $\hat{\eta} = 0.2\eta\sqrt{mn}/\|\widehat{P}_t\|_F$. Then for all t, we have*

$$\|Z_t\|_F \leqslant \|Z_0\|_F(1-\eta\lambda)^t + \frac{0.2\sqrt{mn}}{\lambda} \implies \|Z_t\|_F \leqslant \frac{0.2\sqrt{mn}}{\lambda} \text{ as } t \to \infty.$$

Furthermore, since X_t and Y_t are convex combinations of $\{Z_s\}_{s\leqslant t}$, they satisfy the same bound.

Proof. Triangle inequality gives

$$\|Z_{t+1}\|_F \leqslant (1-\eta\lambda)\|Z_t\|_F + \hat{\eta}_t\|\widehat{P}_t\|_F = (1-\eta\lambda)\|Z_t\|_F + 0.2\eta\sqrt{mn}.$$

Unrolling and bounding the geometric sum:

$$\|Z_t\|_F \leqslant (1-\eta\lambda)^t\|Z_0\|_F + 0.2\eta\sqrt{mn}\sum_{k=0}^{t-1}(1-\eta\lambda)^k \leqslant (1-\eta\lambda)^t\|Z_0\|_F + \frac{0.2\sqrt{mn}}{\lambda}.$$

□

C.5.2 Proof of Lemma 3.2

Lemma 3.2 (Quasi-Steady-State of Z, Restated). *Under the same conditions as Lemma 3.1, for $\eta\lambda \ll 1$ the steady-state satisfies*

$$\lambda\rho_t = 0.1\Big[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\Big],$$

where $\rho_t := \|Z_t\|_F/\sqrt{mn}$ is the RMS norm and $\alpha_t := \langle \widehat{P}_t, Z_t\rangle/(\|\widehat{P}_t\|_F\|Z_t\|_F)$ is the alignment.

Proof. Squaring the update $Z_{t+1} = (1-\eta\lambda)Z_t - \hat{\eta}_t\widehat{P}_t$ gives

$$\|Z_{t+1}\|_F^2 = (1-\eta\lambda)^2\|Z_t\|_F^2 - 2(1-\eta\lambda)\hat{\eta}_t\langle Z_t, \widehat{P}_t\rangle + \hat{\eta}_t^2\|\widehat{P}_t\|_F^2. \tag{108}$$

The NorMuon learning rate $\hat{\eta}_t = 0.2\eta\sqrt{mn}/\|\widehat{P}_t\|_F$ ensures that $\hat{\eta}_t\|\widehat{P}_t\|_F = 0.2\eta\sqrt{mn}$ is constant. Substituting into (108):

$$\|Z_{t+1}\|_F^2 = (1-\eta\lambda)^2\|Z_t\|_F^2 - 0.4\eta(1-\eta\lambda)\alpha_t\sqrt{mn}\|Z_t\|_F + 0.04\eta^2 mn.$$

Dividing by mn:

$$\rho_{t+1}^2 = (1-\eta\lambda)^2\rho_t^2 - 0.4\eta(1-\eta\lambda)\alpha_t\rho_t + 0.04\eta^2. \tag{109}$$

In steady state, $\rho_{t+1} = \rho_t =: \rho_t$. Rearranging (109):

$$\rho_t^2\big[1-(1-\eta\lambda)^2\big] = -0.4\eta(1-\eta\lambda)\alpha_t\rho_t + 0.04\eta^2.$$

For $\eta\lambda \ll 1$, we have $1-(1-\eta\lambda)^2 \approx 2\eta\lambda$ and $1-\eta\lambda \approx 1$, giving the quadratic $\lambda\rho_t^2 + 0.2\alpha_t\rho_t - 0.02\eta = 0$. Taking the positive root:

$$\lambda\rho_t = 0.1\Big[-\alpha_t + \sqrt{\alpha_t^2 + 2\eta\lambda}\Big].$$

□

D Architecture and Hyperparameter Details

This appendix provides complete details of the model architectures and hyperparameter sweeps used in our experiments. All experiments were conducted on 8× NVIDIA A100-40GB GPUs [80] using PyTorch 2 [81].

Table 2: Small and large Model architecture details.

Hyperparameter	**Small Model (125M)**	**Large Model (772M)**
Vocabulary size	50,304	50,304
Number of layers	12	36
Hidden dimension (d_{model})	768	1,280
Number of attention heads	6	10
Head dimension	128	128
MLP hidden dimension	3,072	5,120
Context length	1,024	1,024
Total parameters	∼124M	∼772M
1D parameters (embedding/head)	38,633,472	64,389,120
2D parameters (attention/MLP)	84,934,656	707,788,800
Batch size (global)	512	512
Tokens per batch	524,288	524,288

Model Architecture We use a Llama-style GPT architecture with rotary positional embeddings (RoPE), RMSNorm, and QK-normalization in attention. The MLP uses a squared ReLU activation. Weight tying is applied between the token embedding and output head.

Training Regimes We follow Chinchilla-optimal training budgets [20], scaling the number of training steps proportionally with model size. For the small model, we evaluate four training horizons denoted 1×, 2×, 4×, and 8× Chinchilla-optimal. For the large model, we train at 1×, 2×, and 4× Chinchilla-optimal. Table 3 summarizes the training budgets.

Hyperparameter Search We conduct extensive hyperparameter sweeps for each optimizer using the 125M model. In particular, for AdamW we tuned the all hyper parameters except learning rate at 2× Chinchilla but the *learning rate was tuned for each horizon*, while for SF-AdamW we tuned at 2× Chinchilla and used the

Table 3: Training budgets in steps and tokens for each model and regime.

Model		1×	2×	4×	8×
Small (125M)	Steps	5,000	10,000	20,000	40,000
	Tokens	2.6B	5.2B	10.5B	21.0B
Large (772M)	Steps	30,000	60,000	120,000	—
	Tokens	15.7B	31.5B	62.9B	—

same configuration for runs up to 8× Chinchilla. Using the learning rate adjustment described in Section 2, SF-NorMuon was able to use an identical learning rate and weight decay as SF-AdamW. Table 4 summarizes the search space.

Table 4: Hyperparameter search space for each optimizer. Values in **bold** indicate the best configuration found. The learning rate in bold is for AdamW corresponds to the 2× training duration.

Optimizer	**Hyperparameter**	**Values Searched**
AdamW	Learning rate	{0.002, 0.004, 0.006, **0.008**, 0.01}
	β_1	{**0.9**, 0.95, 0.99, 0.995}
	β_2	{0.9, **0.95**, 0.99, 0.995}
	Weight decay λ	{0, 0.05, **0.1**}
	Warmup steps	{1000, 1500, 2000, **2500**}
	Scheduler	Cosine decay to 0
SF-AdamW	Learning rate	{0.005, **0.008**, 0.01, 0.012, 0.015}
	β_1	{0.9, **0.95**, 0.99, 0.995}
	β_2	{0.9, 0.95, **0.99**, 0.995}
	Weight decay λ	{0, **0.05**, 0.1}
	Warmup steps	{1500, **2000**, 2500}
	Decay location	{**y**, z}
SF-NorMuon	Learning rate	{0.005, **0.008**, 0.01, 0.012, 0.015}
	β_1	{**0.9**, 0.95, 0.99, 0.995}
	β_2	{0.9, **0.95**, 0.99, 0.995}
	Momentum μ	{**0.8**, 0.9, 0.95, 0.99}
	Weight decay λ	{0, **0.05**, 0.1}
	Warmup steps	{1500, **2000**, 2500}
	Decay location	{y, **z**}

Table 5 reports the best hyperparameter configuration for each optimizer on the 125M model at 8× Chinchilla-optimal training. These configurations were used for all experiments conducted for the large model.

Remarks. We make the following observations from our hyperparameter sweeps:

- **Momentum parameters:** For SF-AdamW, $(\beta_1, \beta_2) = (0.95, 0.99)$ consistently outperforms the SF-AdamW default of $(0.9, 0.95)$. For SF-NorMuon, we use separate beta values for 1D and 2D parameters, with the NorMuon-specific momentum $\mu = 0.8$ providing the best results.
- **Decay location:** For SF-AdamW, applying weight decay at the y iterate (gradient evaluation point) works best. For SF-NorMuon, applying decay at the z iterate (base sequence) achieves the lowest validation loss.

Learning Rate Sweep Results Figures 6–5 show the validation loss curves across different learning rates for each optimizer. For all experiments, other hyperparameters are fixed at their optimal values from Table 5.

Table 5: Best hyperparameter configurations for each optimizer on the small model (learning rate corresponds to 8× Chinchilla for AdamW).

Hyperparameter	**AdamW**	**SF-AdamW**	**SF-NorMuon**
Learning rate	0.004	0.008	0.008
(β_1, β_2) for 1D params	(0.9, 0.95)	(0.95, 0.99)	(0.95, 0.99)
(β_1, β_2) for 2D params	(0.9, 0.95)	(0.95, 0.99)	(0.9, 0.95)
Momentum μ	—	—	0.8
Weight decay λ	0.1	0.05	0.05
Warmup steps	2500	2000	2000
Decay location	—	y	z
Scheduler	Cosine	None	None

E Ablation of SF-NorMuon and Comparison with NorMuon

We conduct ablation experiments to validate the key design choices in SF-NorMuon, and compare schedule-free methods against NorMuon across multiple training horizons.

E.1 Ablation of SF-NorMuon Components

In Figure 3, we conduct ablation experiments to validate the two key modifications that distinguish SF-NorMuon from schedule-free spectral descent: the explicit momentum buffer and the row-wise adaptive normalization. All experiments use the 125M parameter model trained for 8× Chinchilla tokens (≈21B tokens) with decay at Z.

Explicit momentum. As discussed in §2, the polar update treats all singular directions uniformly, which can be more aggressive than coordinate-wise methods. We therefore introduce an explicit momentum buffer $M_t = \mu M_{t-1} + (1-\mu)G_t$ to smooth the gradient before computing its polar factor. The left panel of Figure 3 compares SF-NorMuon ($\mu = 0.8$) against the ablation with no explicit momentum ($\mu = 0$). Removing momentum increases the final validation loss from 3.14 to 3.28, a rather large degradation. This confirms that explicit momentum is essential for stable, effective training with spectral updates.

Row-wise normalization. The polar factor can produce updates with high variance across neurons, as different rows of a weight matrix may have vastly different typical magnitudes. Row-wise adaptive normalization addresses this by maintaining a running average of squared row norms and normalizing accordingly. The right panel of Figure 3 compares SF-NorMuon against the variant without row normalization on a tighter y-axis scale to reveal the small but consistent improvement. SF-NorMuon achieves a final loss of 3.136 compared to 3.142 for SF-Muon, and reaches the same loss approximately 12% faster in terms of tokens processed. This speedup is quite similar to the speedup achieved by NorMuon relative to Muon [36].

E.2 Comparison with NorMuon

We compare schedule-free methods (SF-NorMuon, SF-AdamW) against their scheduled counterparts (NorMuon, AdamW) across multiple training horizons (1×, 2×, 4×, 8× Chinchilla). All scheduled runs use a cosine learning rate schedule tuned for each horizon. Table 6 reports the final validation loss for each method at each horizon, and Figure 7 visualizes the training curves.

NorMuon vs AdamW. Across all horizons, NorMuon consistently outperforms AdamW. This confirms the benefits of spectral-norm geometry for matrix-valued parameters as discussed in §2, because the polar

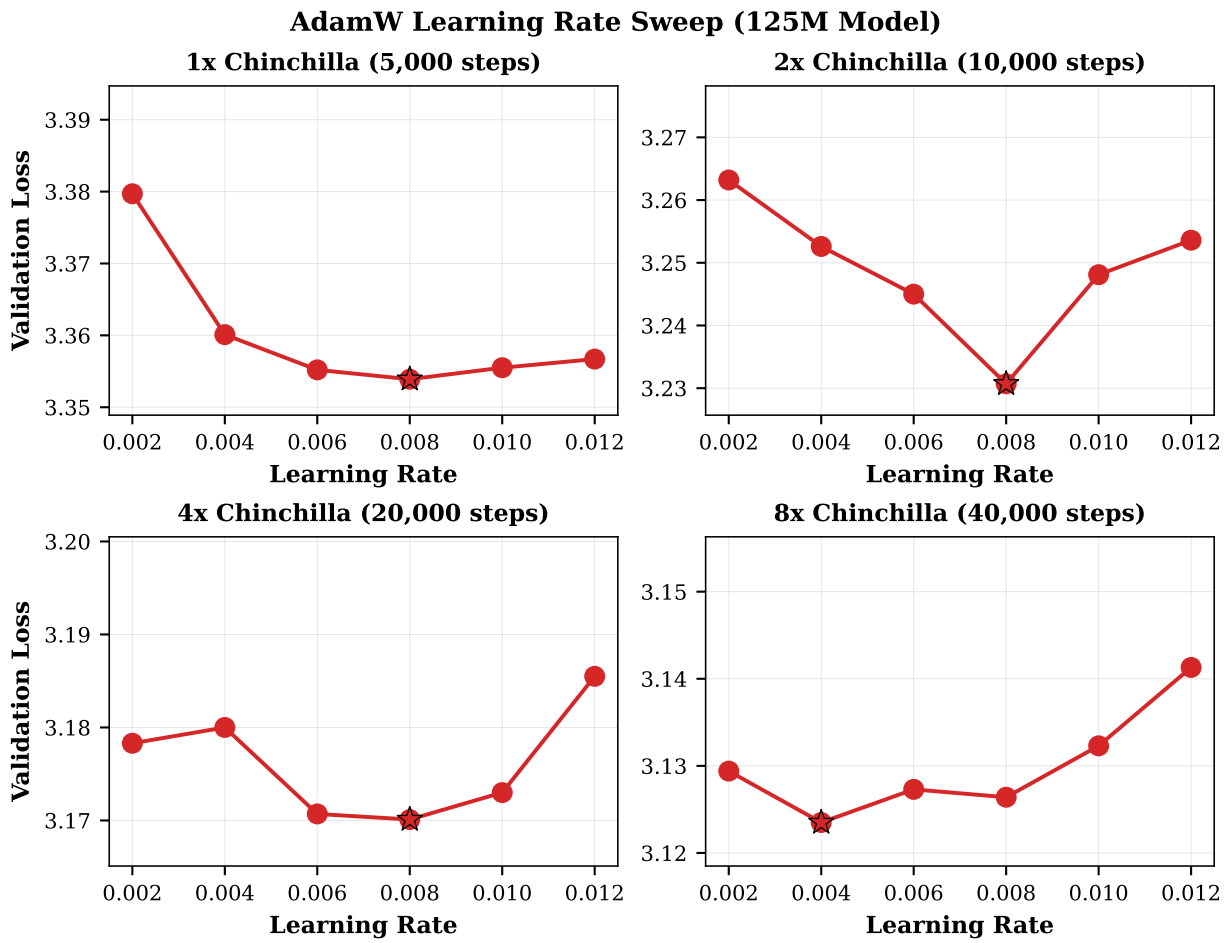

Figure 6: Learning rate sweep for AdamW with cosine scheduling on the 125M model across four training horizons: 1× (5K steps), 2× (10K steps), 4× (20K steps), and 8× (40K steps) Chinchilla-optimal. The optimal learning rate is $\eta = 0.008$ for 1×– 4× Chinchilla, and $\eta = 0.004$ for 8× Chinchilla, indicating that longer training benefits from smaller learning rates.

update makes uniform progress across all singular directions of the gradient rather than being dominated by large coordinate-wise entries.

SF-NorMuon is competitive with scheduled NorMuon. Despite being an anytime optimizer that does not know the training horizon in advance, SF-NorMuon closely tracks the performance of scheduled NorMuon. At each horizon, the gap between SF-NorMuon and NorMuon is only $\sim 0.02 - 0.03$ nats. This is notable because scheduled methods benefit from the learning rate cool down at the end of training horizon [82], yet SF-NorMuon remains competitive through implicit learning rate decay via online averaging. In contrast, SF-AdamW under performs both AdamW and also SF-NorMuon substantially.

Table 6: Final validation loss at each training horizon (125M model). Scheduled methods (NorMuon, AdamW) use cosine decay tuned for each horizon; schedule-free methods (SF-NorMuon, SF-AdamW) use the same hyperparameters throughout and can be evaluated at any checkpoint.

Horizon	SF-NorMuon	SF-AdamW	NorMuon	AdamW
1× (2.6B tokens)	3.318	3.399	3.292	3.354
2× (5.2B tokens)	3.229	3.282	3.200	3.231
4× (10.5B tokens)	3.170	3.205	3.139	3.170
8× (21.0B tokens)	3.136	3.141	3.103	3.124

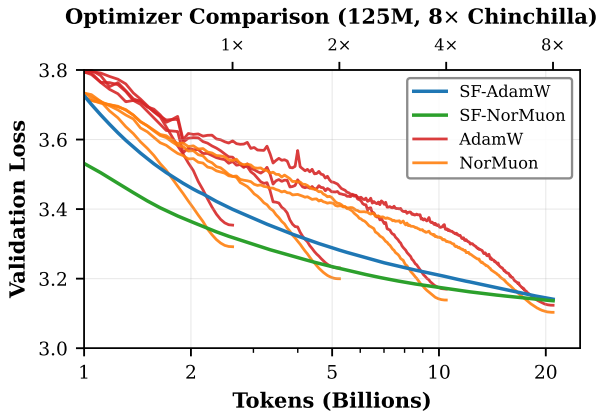

Figure 7: Optimizer comparison on the 125M model. Scheduled methods (NorMuon, AdamW) are shown with separate curves for each horizon (1×, 2×, 4×, 8× Chinchilla), while schedule-free methods (SF-NorMuon, SF-AdamW) use a single run evaluated at any point. SF-NorMuon closely tracks scheduled NorMuon throughout training, demonstrating that the benefits of spectral-norm optimization transfer effectively to the schedule-free setting.

F Reference PyTorch Implementation

Below we provide a self-contained PyTorch implementation of SF-NorMuon, closely following Algorithm 1. Our implementation is based on the code from [17, 34].

Implementation notes.

- The `zeropower_via_newtonschulz5` function computes an approximation to the polar factor $P = \text{polar}(M)$ using 5 iterations of the Newton-Schulz method with coefficients $(a, b, c) = (3.4445, -4.7750, 2.0315)$. The function handles both tall and wide matrices by transposing when necessary. There are other alternatives methods to accomplish the polar transformation which either use randomized methods [37, 38] or differ in the choice of coefficients [41].
- In addition to the live weights `p` (the gradient evaluation sequence Y_t), the optimizer maintains three state variables per parameter: `z` (the fast iterate Z_t), `v` (the row-wise second moment estimate v_t), and `mom` (the momentum buffer M_t).
- The `eval()` method computes X_t on the fly using Z_t and Y_t. The `train()` method restores `p` to the Y_t iterate for further training.
- Weight decay is applied to the fast iterate Z_t, consistent with our analysis in Section 3, and the learning rate scaling factor `eta_scale` (default 0.2) normalizes the effective step size to be comparable to Adam's RMS-normalized updates.

```python
import math, torch

@torch.compile
def zeropower_via_newtonschulz5(G, steps=5, eps=1e-7):
    assert len(G.shape) == 2
    a, b, c = (3.4445, -4.7750, 2.0315)
    X = G.bfloat16()
    X /= (X.norm() + eps)
    if G.size(0) > G.size(1):
        X = X.T
        for _ in range(steps):
            A = X @ X.T
            B = A @ X
            X = a * X + b * B + c * A @ B
        return X.T
    for _ in range(steps):
        A = X @ X.T
        B = A @ X
        X = a * X + b * B + c * A @ B
    return X

class NorMuonScheduleFree(torch.optim.Optimizer):
    def __init__(self, params, lr=0.005, betas=(0.9, 0.95), momentum=0.8,
                 eps=1e-8, weight_decay=0.1, warmup_steps=2000, eta_scale=0.2):
        defaults = dict(lr=lr, betas=betas, momentum=momentum, eps=eps,
                        weight_decay=weight_decay, warmup_steps=warmup_steps,
                        eta_scale=eta_scale, k=0, train_mode=False, weight_sum=0.0)
        super().__init__(params, defaults)

    @torch.no_grad()
    def eval(self):
        for group in self.param_groups:
            if group["train_mode"]:
                beta = group["betas"][0]
                for p in group["params"]:
                    state = self.state[p]
                    if "z" in state: p.lerp_(end=state["z"], weight=1.0 - 1.0 / beta)
                group["train_mode"] = False

    @torch.no_grad()
    def train(self):
        for group in self.param_groups:
            if not group["train_mode"]:
                beta = group["betas"][0]
                for p in group["params"]:
                    state = self.state[p]
                    if "z" in state: p.lerp_(end=state["z"], weight=1.0 - beta)
                group["train_mode"] = True

    @torch.no_grad()
    def step(self, closure=None):
        loss = closure() if closure else None
        for group in self.param_groups:
            beta, beta2 = group["betas"]
            mu, eps = group["momentum"], group["eps"]
            eta_scale, decay = group["eta_scale"], group["weight_decay"]
            k, warmup_steps = group["k"], group["warmup_steps"]
            sched = (k + 1) / warmup_steps if k < warmup_steps else 1.0
            lr = group["lr"] * sched
            weight = lr * lr
            weight_sum = group["weight_sum"] = group["weight_sum"] + weight
            ckp1 = weight / weight_sum
            for p in group["params"]:
                if p.grad is None: continue
                grad, state = p.grad, self.state[p]
                if "z" not in state:
                    state["z"] = p.clone()
                    state["v"] = torch.zeros(p.shape[0], device=p.device, dtype=torch.float32)
                    state["mom"] = torch.zeros_like(p)
                z, v, mom = state["z"], state["v"], state["mom"]
                mom.mul_(mu).add_(grad, alpha=1.0 - mu)
                P = zeropower_via_newtonschulz5(mom).to(p.dtype)
                row_ms = (P * P).mean(dim=1).float()
                v.mul_(beta2).add_(row_ms, alpha=1.0 - beta2)
                Phat = P / (v.sqrt() + eps).to(P.dtype).unsqueeze(1)
                m, n = p.shape
                eta_hat = eta_scale * lr * math.sqrt(m * n) / max(1e-12, Phat.float().norm())
                x_t = (p - (1.0 - beta) * z) / beta  # recover X_t before z update
                if decay != 0.0: z.sub_(z, alpha=lr * decay)
                z.sub_(Phat, alpha=eta_hat)  # z is now Z_{t+1}
                x_tp1 = (1.0 - ckp1) * x_t + ckp1 * z
                p.copy_((1.0 - beta) * z + beta * x_tp1)
            group["k"] = k + 1
        return loss
```