# Through the River: Understanding the Benefit of Schedule-Free Methods for Language Model Training

**Minhak Song**[*] MINHAKSONG@KAIST.AC.KR
*Department of Mathematical Sciences, KAIST, Daejeon, South Korea*

**Beomhan Baek**[*†] BHBAEK2001@SNU.AC.KR
*Department of Mathematical Sciences, Seoul National University, Seoul, South Korea*

**Kwangjun Ahn** KWANGJUNAHN@MICROSOFT.COM
*Microsoft Research, Cambridge, MA, United States*

**Chulhee Yun** CHULHEE.YUN@KAIST.AC.KR
*Kim Jaechul Graduate School of AI, KAIST, Seoul, South Korea*

## Abstract

As both model and dataset sizes continue to scale rapidly, conventional pretraining strategies with fixed compute budgets—such as cosine learning rate schedules—are increasingly inadequate for large-scale training. Recent alternatives, including warmup-stable-decay (WSD) schedules and weight averaging, offer greater flexibility. However, WSD relies on explicit decay phases to track progress, while weight averaging addresses this limitation at the cost of additional memory. In search of a more principled and scalable alternative, we revisit the Schedule-Free (SF) method (Defazio et al., 2024), which has shown strong empirical performance across diverse settings. We show that SF-AdamW effectively navigates the "river" structure of the loss landscape without decay phases or auxiliary averaging, making it particularly suitable for continuously scaling training workloads. To understand this behavior, we conduct a theoretical and empirical analysis of SF dynamics, revealing that it implicitly performs weight averaging without memory overhead. Guided by this analysis, we propose a refined variant of SF that improves robustness to momentum and performs better under large batch sizes, addressing key limitations of the original method. Together, these results establish SF as a practical, scalable, and theoretically grounded approach for language model training.

## 1. Introduction

As both model and dataset sizes continue to scale rapidly, conventional pretraining strategies with fixed training budgets—such as cosine learning rate schedules (Loshchilov and Hutter, 2017)—are becoming increasingly inadequate. These static approaches are ill-suited to the demands of large-scale, evolving datasets and open-ended training regimes. For example, DeepSeek-V3 (Liu et al., 2024, §4.2) employs a sophisticated multi-phase training procedure that falls outside the scope of traditional cosine scheduling.

To support prolonged and flexible training, practitioners have adopted more adaptive scheduling strategies. One widely used approach is the *warmup-stable-decay* (WSD) schedule (Hu et al., 2024), which avoids committing to a fixed compute budget by maintaining a main "branch" with a constant learning rate (LR) and periodically branching into decaying

[*] Equal contribution.
[†] Work done as an undergraduate intern at KAIST.

LR trajectories to produce intermediate checkpoints—enabling flexible and continued training. Despite its advantages, WSD has notable limitations. A key challenge lies in evaluating the quality of the current model without explicitly entering the decay phase. This lack of visibility complicates decisions around checkpointing and training continuation, leading to uncertainty in training management (see Appendix A.1).

One common workaround is to maintain a weight averaging, which improves generalization and provides more stable performance estimation. However, this comes at the cost of additional memory overhead and implementation complexity, especially in distributed training setups (see Appendix A.2).

These challenges motivate a key question:

*Is there an alternative with better flexibility, visibility, and minimal resource overhead?*

**Our main contributions.** In this work, we explore this question and identify the Schedule-Free (SF) method (Defazio et al., 2024) as a principled and scalable approach for language model pretraining. Our contributions are summarized as follows:

- We revisit the river-valley loss landscape and analyze two widely used strategies—WSD and weight averaging—through this lens, highlighting their respective strengths and limitations for scalable pretraining (Section 2).
- We then focus on the SF method and show that it effectively follows the "river" structure of the loss landscape without requiring a decay phase or auxiliary averaging. This makes it particularly well-suited for continuously scaling training workloads (Section 3).

## 2. Candidate Strategies for Scalable Pretraining

### 2.1. Backgrounds: Loss Landscape of Neural Networks

Despite the remarkable success of deep learning across numerous domains, classical optimization theory falls short of explaining the underlying dynamics of neural network training. In particular, the structure of the loss landscape has attracted growing attention as researchers seek to better understand why deep learning works and how to design more effective optimization algorithms.

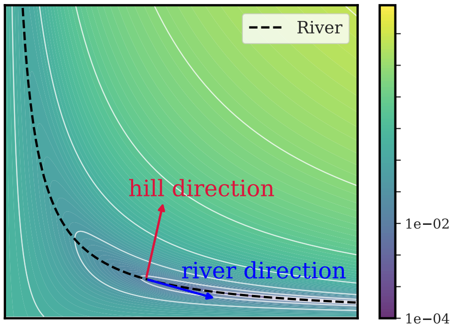

Figure 1: **River-valley structure in a toy loss landscape.** Contour plot of the objective defined in Appendix B.1, illustrating the flat river direction and steep hill direction characteristic of the river-valley geometry.

**River-Valley Landscape.** Recent studies—motivated by diverse goals—have converged on a common hypothesis regarding the geometry of neural network loss landscape. This hypothesis, which we refer to as the *river-valley loss landscape* (Wen et al., 2025), is closely related to concepts such as the *ill-conditioned valley* (Song et al., 2025), *basin in the loss landscape* (Hägele et al., 2024), and *ravine in the loss landscape* (Davis et al., 2024).

As its name suggests, a *river-valley* loss landscape resembles a winding ravine: steep "hill" walls flank a relatively flat "river" floor that snakes through parameter space. Wen et al. (2025) formalize this picture to interpret the warmup-stable-decay (WSD) LR schedule, arguing that large, noisy updates during the *stable* phase let SGD travel quickly *downstream* along the river, while the fast-decay phase pulls the iterate back to the bottom of valley. A concrete illustration is provided in Figure 1 (see also Appendix B.1). Complementary evidence from Song et al. (2025) show that "effective" optimizer updates happen along the river: projecting out the high-curvature hill directions does not harm progress of learning, indicating that motion in the hill directions is often dispensable once the iterate is near the river.

Motivated by these works, we explicitly decompose the loss into two orthogonal parts:

- the **river component**, which measures progress *along* the low-curvature valley floor, and
- the **hill component**, which penalizes deviations *away* from that floor.

In Appendix A, we revisit existing strategies—WSD and weight averaging—for scalable pretraining through the lens of the river-valley perspective.

### 2.2. Schedule-Free Methods

The Schedule-Free (SF) method (Defazio et al., 2024) provides a general framework that interpolates between two classical techniques: Polyak-Ruppert averaging, which returns the average of past iterates, and primal averaging, where gradients are evaluated at the averaged point. The abstract formulation of the SF method is given by:

$$\begin{aligned} \mathbf{x}_t &= (1-c_t)\,\mathbf{x}_{t-1} + c_t\,\mathbf{z}_t, \\ \mathbf{y}_t &= (1-\beta)\,\mathbf{z}_t + \beta\,\mathbf{x}_t, \\ \mathbf{z}_{t+1} &= \mathbf{z}_t - \gamma\Delta_t, \end{aligned} \tag{SF}$$

where $\gamma$ is the LR, $\beta$ is a momentum-like coefficient, $c_t = 1/t$, and the initialization satisfies $\mathbf{z}_1 = \mathbf{x}_1$. The update direction $\Delta_t$ is generic, making SF a flexible framework that can be combined with any baseline optimizer. For example, in Schedule-Free SGD, $\Delta_t$ corresponds to a stochastic gradient evaluated at the $\mathbf{y}_t$ iterate.

In this work, we focus on Schedule-Free AdamW (`SF-AdamW`), where $\Delta_t$ is computed using the RMSprop update along with a weight decay term. The full pseudocode is provided in Algorithm 1. Here, $\beta_1$ denotes the coefficient $\beta$ in (SF), and $\beta_2$ is the momentum parameter used in the second-moment of RMSprop.

`SF-AdamW` has demonstrated state-of-the-art performance across a range of deep learning tasks, including winning the Self-Tuning track in the 2024 AlgoPerf Challenge (Dahl et al., 2023; Kasimbeg et al., 2025). Importantly, it achieves this without requiring additional memory overhead compared to `AdamW`. However, its practical deployment reveals two key limitations: sensitivity to momentum hyperparameters (Hägele et al., 2024) and degraded performance under large batch sizes (Zhang et al., 2025; Morwani et al., 2025). We revisit both limitations in later sections and propose a refined variant of SF that addresses them.

## 3. Schedule-Free Optimizer as a Scalable Pretraining Method

In Section 2, we discussed the limitations of WSD and weight averaging as strategies for scalable pretraining. While WSD relies on a decay phase to achieve optimal performance, weight averaging avoids decay but incurs additional memory overhead. In this section, we empirically investigate the SF method as an alternative. We find it to be a strong candidate, as it requires neither decay nor extra memory.

**Experimental Setup.** We use a 124M parameter LLaMA (Touvron et al., 2023a,b) style decoder-only transformer, trained with `SF-AdamW` using a warmup phase followed by a constant LR. The batch size is 0.5M tokens, and training is conducted on a 6B-token subset of SlimPajama (Soboleva et al., 2023), with the compute budget determined by the Chinchilla scaling rule (Hoffmann et al., 2022). We report validation loss in terms of perplexity. Additional results using a 124M parameter GPT-2 (Radford et al., 2019) style decoder-only transformer trained on the OpenWebText2 dataset (Gao et al., 2020) are provided in Appendix F. Full training details are available in Appendix E.

**Vanishing Benefit of Learning Rate Decay and Weight Averaging.** As noted in Section 2, standard `AdamW` with a constant LR typically yields suboptimal performance, requiring a decay phase or weight averaging to reach better solutions. To test whether `SF-AdamW` exhibits similar behavior, we first perform a grid search to identify the best hyperparameters for both `AdamW` and `SF-AdamW` under constant LR (after warmup). In our setting, the optimal hyperparameters are $(\beta_1, \beta_2) = (0.9, 0.95)$ with LR 1e-3 for `AdamW`, and $(\beta_1, \beta_2) = (0.95, 0.99)$ with LR 2e-3 for `SF-AdamW`. Using these configurations, we train each model and periodically save checkpoints. At each checkpoint, we run a LR decay phase and evaluate the resulting loss. We also track the EWA of the $\mathbf{x}_t$ iterates throughout training. Results are shown in Figure 2.

Surprisingly, unlike `AdamW`, *neither the decay phase nor EWA provides additional benefit*: `SF-AdamW` with a constant LR consistently reaches near-optimal solutions on its own.

**Schedule-Free Optimizer Tracks the River.** We next examine how closely the `SF-AdamW` trajectory follows the river in the loss landscape, building on the observation by Wen et al. (2025) that `AdamW` with a decaying LR converges toward the river. Specifically, we run a short decay phase of `AdamW` (linear decay from 1e-4 to 0) at each checkpoint from the `SF-AdamW` run described in the previous experiment. This decay phase—starting from a small LR—is designed to make minimal progress along the river direction while substantially reducing the hill component, pulling the iterate toward the valley floor. As a baseline, we apply the same procedure to `AdamW`. Results are shown in Figure 2.

We observe that applying a decay phase of `AdamW` to the `SF-AdamW` trajectory results in minimal additional loss reduction, in contrast to the `AdamW` trajectory where the decay phase leads to a sharp drop in loss. This suggests that `SF-AdamW` already closely tracks the river throughout training, eliminating the need for LR decay or weight averaging.

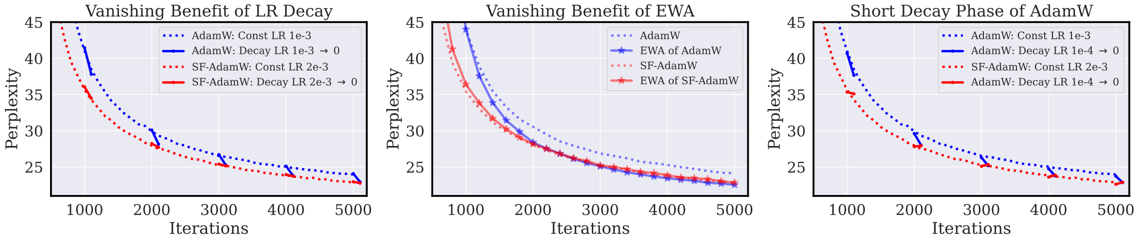

Figure 2: **SF-AdamW closely follows the river, unlike AdamW. Left, Middle:** While `AdamW` benefits from linear LR decay and EWA, `SF-AdamW` shows no improvement from either. **Right:** A short decay phase of `AdamW` (with linear LR decay from 1e-4 to 0) leads to a sharp loss drop for `AdamW`, but has no effect when applied to the `SF-AdamW` trajectory—suggesting that `SF-AdamW` already tracks the river throughout training (Observation 1).

***Observation 1:*** `SF-AdamW` can follow the river without LR decay or weight averaging.

**Sensitivity to Momentum Parameters.** Despite its strong empirical performance, SF method is highly sensitive to the choice of momentum parameters. For example, Hägele et al. (2024) report that `SF-AdamW` with $(\beta_1, \beta_2) = (0.9, 0.95)$ performs significantly worse in their pretraining setup, even exhibiting rising loss toward the end of training, whereas $(0.95, 0.99)$ leads to strong results. This sensitivity contrasts with the theoretical analysis of Defazio et al. (2024), which shows that the SF method is worst-case optimal for any momentum setting in convex Lipschitz problems.

To further investigate this gap, we repeat our experiment using `SF-AdamW` with suboptimal momentum $\beta_1 \in \{0.1, 0.5\}$. As before, we periodically save checkpoints and apply a short `AdamW` decay phase at each. Unlike the optimal case ($\beta_1 = 0.95$), we observe that the decay phase improves performance, suggesting that suboptimal momentum disrupts the optimizer's ability to follow the river. Results are shown in Figure 3.

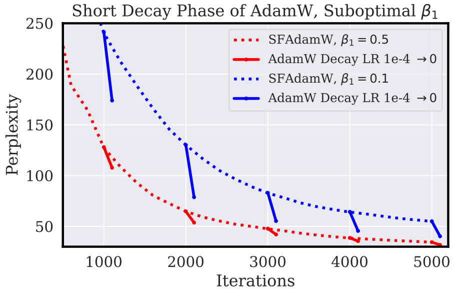

Figure 3: **SF-AdamW with suboptimal momentum fails to follow the river.** A short decay phase of `AdamW` applied to `SF-AdamW` checkpoints with $\beta_1 \in \{0.1, 0.5\}$ results in a sharp loss drop, unlike the case with $\beta_1 = 0.95$ (Observation 2).

***Observation 2:*** `SF-AdamW` is highly sensitive to momentum; poor choices can prevent it from reaching and following the river.

These findings lead to the following central question:

*Why does a well-tuned Schedule-Free method successfully follow the river, and what makes this behavior so sensitive to momentum?*

We address this question in Appendix B by analyzing the training dynamics of SF methods.

## 4. Summary of Additional Results

Due to space limit, we provide a summary of our additional results as below.

- We analyze the training dynamics of SF both theoretically and empirically. Our findings reveal that SF implicitly performs a form of weight averaging, without requiring additional memory. We also show that it operates at the Edge of Stability (Cohen et al., 2021) and derive its associated central flow (Cohen et al., 2025), providing a deeper understanding of its behavior (Appendix B).
- Based on these insights, we propose a refined version of the SF method that improves robustness to momentum parameters and scale better with large batch sizes—addressing key limitations of the original method (Appendix C).

## 5. Conclusion

We presented a principled view of Schedule-Free (SF) methods by studying its behavior through the geometry of the river-valley loss landscape. Our analysis shows that SF methods naturally follow the river without requiring explicit LR decay or weight averaging, making them a compelling approach for scalable pretraining. We further provided theoretical insights grounded in Edge of Stability and central flow dynamics. Building on this understanding, we proposed a refined variant that decouples momentum from averaging, improving both robustness and performance.

While our findings highlight the potential of SF methods, several open questions remain. Our theoretical analysis relies on simplifying assumptions, and validating the central flow approximation in deep learning is a natural next step. Furthermore, extending the river-valley framework to analyze other modern optimizers—and exploring their integration with SF methods—offers a promising direction for further investigation. Lastly, our experiments are limited to small-scale language models due to computational constraints. Scaling these findings to larger models and longer training durations remains an important direction for future work.

## Acknowledgements

This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00421203).

## References

Kwangjun Ahn and Ashok Cutkosky. Adam with model exponential moving average is effective for nonconvex optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=v416YLOQuU.

Kwangjun Ahn, Sebastien Bubeck, Sinho Chewi, Yin Tat Lee, Felipe Suarez, and Yi Zhang. Learning threshold neurons via edge of stability. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=9cQ6kToLnJ.

Kwangjun Ahn, Gagik Magakyan, and Ashok Cutkosky. General framework for online-to-nonconvex conversion: Schedule-free sgd is also effective for nonconvex optimization. *ICML*, 2025.

Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 948–1024. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/arora22a.html.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=jh-rTtvkGeM.

Jeremy Cohen, Alex Damian, Ameet Talwalkar, J Zico Kolter, and Jason D. Lee. Understanding optimization in deep learning with central flows. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sIE2rI3ZPs.

Jeremy M Cohen, Behrooz Ghorbani, Shankar Krishnan, Naman Agarwal, Sourabh Medapati, Michal Badura, Daniel Suo, David Cardoze, Zachary Nado, George E Dahl, et al. Adaptive gradient methods at the edge of stability. *arXiv preprint arXiv:2207.14484*, 2022.

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

Alex Damian, Eshaan Nichani, and Jason D. Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nhKHA59gXz.

Damek Davis, Dmitriy Drusvyatskiy, and Liwei Jiang. Gradient descent with adaptive stepsize converges (nearly) linearly under fourth-order growth, 2024. URL https://arxiv.org/abs/2409.19791.

Aaron Defazio, Xingyu Yang, Ahmed Khaled, Konstantin Mishchenko, Harsh Mehta, and Ashok Cutkosky. The road less scheduled. *Advances in Neural Information Processing Systems*, 37:9974–10007, 2024.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor

Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Y13gSfTjGr.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack William Rae, and Laurent Sifre. Training compute-optimal large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=iBBcRUlOAPR.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. MiniCPM: unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019. URL https://arxiv.org/abs/1803.05407.

Priya Kasimbeg, Frank Schneider, Runa Eschenhagen, Juhan Bae, Chandramouli Shama Sastry, Mark Saroufim, BOYUAN FENG, Less Wright, Edward Z. Yang, Zachary Nado, Sourabh Medapati, Philipp Hennig, Michael Rabbat, and George E. Dahl. 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.

Yunshui Li, Yiyuan Ma, Shen Yan, Chaoyi Zhang, Jing Liu, Jianqiao Lu, Ziwen Xu, Mengzhao Chen, Minrui Wang, Shiyi Zhan, et al. Model merging in pre-training of large language models. *arXiv preprint arXiv:2505.12082*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. DeepSeek-V3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024.

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.

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Maosong Sun, Zhiyuan Liu, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=KnoS9XxIlK.

Depen Morwani, Nikhil Vyas, Hanlin Zhang, and Sham Kakade. Connections between schedule-free optimizers, ademamix, and accelerated sgd variants. *arXiv preprint arXiv:2502.02431*, 2025.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

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

David Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

Mark Sandler, Andrey Zhmoginov, Max Vladymyrov, and Nolan Miller. Training trajectories, mini-batch losses and the curious role of the learning rate, 2023. URL https://arxiv.org/abs/2301.02312.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Minhak Song and Chulhee Yun. Trajectory alignment: Understanding the edge of stability phenomenon via bifurcation theory. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 71632–71682. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/e2a9256bd816ab9e082dfaa22f1f62a2-Paper-Conference.pdf.

Minhak Song, Kwangjun Ahn, and Chulhee Yun. Does SGD really happen in tiny subspaces? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=v6iLQBoIJw.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2023.127063. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Zixuan Wang, Zhouzi Li, and Jian Li. Analyzing sharpness along GD trajectory: Progressive sharpening and edge of stability. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=thgItcQrJ4y.

Kaiyue Wen, Zhiyuan Li, Jason S. Wang, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=m51BgoqvbP.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12104–12113, 2022.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham M. Kakade. How does critical batch size scale in pre-training? In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=JCiF03qnmi.

Xingyu Zhu, Zixuan Wang, Xiang Wang, Mo Zhou, and Rong Ge. Understanding edge-of-stability training dynamics with a minimalist example. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=p7EagBsMAEO.

# Appendix

# Appendix A. Candidate Strategies for Scalable Pretraining: WSD and Weight Averaging

## A.1. Warmup-Stable-Decay Schedule

To address the limitations of cosine schedules—particularly their reliance on a pre-specified training budget—the warmup-stable-decay (WSD) schedule has been proposed as a more flexible alternative (Zhai et al., 2022; Hu et al., 2024). WSD divides into three phases: warmup, stable, and decay, with the LR controlled separately in each. Unlike traditional schedules, WSD avoids committing to a fixed training horizon by maintaining a main branch with a constant LR and periodically branching off with decaying LRs to produce intermediate checkpoints. This structure enables flexible evaluation and checkpointing without the need to predefine the total number of training steps, making WSD a widely adopted strategy for scalable pretraining.

**Understanding WSD.** The WSD LR schedule has been widely adopted in large language model (LLM) pretraining due to its strong empirical performance. Motivated by this success, recent studies have sought to understand the mechanisms underlying its effectiveness. Hägele et al. (2024) systematically explore the impact of WSD hyperparameters and propose optimal choices for both the decay schedule and timing. In parallel, Luo et al. (2025) introduce a multi-power-law framework that predicts final pretraining loss as a function of the LR schedule. Interestingly, their learned optimal schedules closely resemble the WSD pattern, providing further evidence for its effectiveness.

Wen et al. (2025) provide a geometric interpretation of WSD through the lens of the river-valley loss landscape. Their key insights are:

1. During the stable phase, the high LR amplifies stochastic gradient noise, inducing oscillations along the high-curvature hill directions. Nevertheless, it enables rapid progress along the low-curvature river direction, which drives long-term improvement.
2. The decay phase plays a crucial role near convergence: it reduces oscillations in the hill directions and steers the iterate toward the valley floor, resulting in a sharp drop in loss that is not achievable during the stable phase alone.

**Limitations.** A key limitation of WSD is its reliance on manually initiating the decay phase. While the stable phase often yields a relatively flat loss curve, a sharp drop typically occurs only once decay begins, which makes it difficult to assess model quality or forecast final performance in advance. This raises a natural question: can we design optimizers that closely track optimal loss—by reaching the valley floor and following the river—*without* relying on explicit learning rate decay?

### A.2. Weight Averaging

Since its introduction in stochastic approximation (Ruppert, 1988; Polyak and Juditsky, 1992), parameter averaging has been widely explored for improving optimization stability and generalization in deep learning. By reducing gradient noise and smoothing the optimization trajectory, averaging schemes can often eliminate the need for explicit LR decay, making them an appealing candidate for scalable pretraining. Among them, two widely studied approaches stand out:

1. **Stochastic Weight Averaging (SWA).** SWA (Izmailov et al., 2019) enhances generalization by periodically averaging model weights. While the original method uses a cyclic LR schedule and averages every $c$ steps, many subsequent works simplify it by setting $c = 1$, performing a standard running averaging. Hägele et al. (2024) further refine SWA by applying fixed-length window averaging under constant LR, and demonstrated improved performance.
2. **Exponential Weight Averaging (EWA).** EWA maintains an exponential moving average of model weights, continuously smoothing the optimization trajectory. Recently, Zhang et al. (2025) show that combining EWA with a constant LR match the performance of cosine schedulers and WSD, particularly in large-batch settings. EWA has also been proven to be effective *theoretically* in nonconvex optimization (Ahn and Cutkosky, 2024; Ahn et al., 2025).

Interestingly, weight averaging is often regarded as functionally equivalent to LR decay. Sandler et al. (2023) analyze schemes like SWA and EWA, showing that their dynamics closely resemble those induced by decaying LRs. Motivated by this, several works advocate for weight averaging as a viable alternative to schedulers in scalable pretraining (e.g., Hägele et al., 2024, §4.1). Concurrent with our work, Li et al. (2025) report that training with a constant LR, when combined with weight averaging, matches the performance of models trained with decaying schedules at any point during training, without the need for LR decay.

From the river-valley perspective, weight averaging serves to cancel out oscillations along hill directions, enabling the optimization trajectory to align more closely with the river—*without* relying on explicit LR decay.

**Limitations.** Despite its benefits, weight averaging introduces a memory overhead, as it requires maintaining an additional copy of the model parameters. This becomes a bottleneck in large-scale LLM pretraining. For instance, storing a single 16-bit copy of LLaMA-8B requires over 16 GB of memory. This limits the practicality of weight averaging in memory-constrained or large-scale training environments.

## Appendix B. Understanding the Training Dynamics of Schedule-Free Optimizer

### B.1. Warmup: A Toy Model

To build intuition, we start by studying the following simple two-dimensional objective:

$$f(\mathbf{w}) = \frac{1}{2}(w^{(1)}w^{(2)} - 1)^2 + \log(1 + \exp(-w^{(1)}))\ ,$$

where $\mathbf{w} = (w^{(1)}, w^{(2)}) \in \mathbb{R}^2$. As shown in Figure 1, this objective exhibits a *river-valley* structure: the curve $w^{(1)}w^{(2)} = 1$ forms a river, along which the loss slowly decreases as $w^{(1)}$ increases. We use this toy model to visualize the training dynamics of `SF-AdamW` in a controlled setting.

We run `SF-AdamW` with a constant LR, initializing at $\mathbf{x}_1 = (2, 2)$, fixing $\beta_2 = 0.99$ and varying $\beta_1 \in \{0.1, 0.5, 0.9\}$. For each run, we track the iterates $(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)$. Results are shown in Figure 4.

We observe that the $\mathbf{y}_t$ iterates—where the gradient is evaluated—oscillate around the river across all momentum settings, with the center of oscillation remaining close to the river. In contrast, the $\mathbf{x}_t$ iterates fail to track the river for suboptimal values of $\beta_1$ (0.1 and 0.5), whereas $\beta_1 = 0.9$ results in a trajectory that remain closely aligned with the river.

This behavior aligns with Observation 2, where we observed that `SF-AdamW` fails to follow the river under suboptimal momentum configurations in language model training. Notably, even when $\beta_1$ is suboptimal, the $\mathbf{y}_t$ iterates continue to track the river on average, despite exhibiting oscillations (we revisit the nature of this oscillation in Appendix B.3). These observations suggest that the $\mathbf{y}_t$ sequence is more robust to $\beta_1$ and better aligned with the river geometry than the $\mathbf{x}_t$ iterates, making it a more reliable foundation for analyzing and guiding optimization in SF dynamics. We investigate this further in the context of language model training.

### B.2. Language Model Training

We evaluate the loss at the $\mathbf{y}_t$ iterates, as well as the EWA of $\mathbf{y}_t$, using the same experimental runs from Section 3 with $\beta_1 \in \{0.1, 0.5, 0.95\}$. Results are shown in Figure 5. Notably, for suboptimal $\beta_1$, we observe that the loss at $\mathbf{y}_t$ is consistently lower than that at $\mathbf{x}_t$, showing that $\mathbf{y}_t$ more faithfully follows the river geometry and remains robust to suboptimal momentum settings. This mirrors our findings in the toy model analysis (Appendix B.1), where $\mathbf{x}_t$ failed to follow the river under suboptimal momentum, while $\mathbf{y}_t$ continued to track

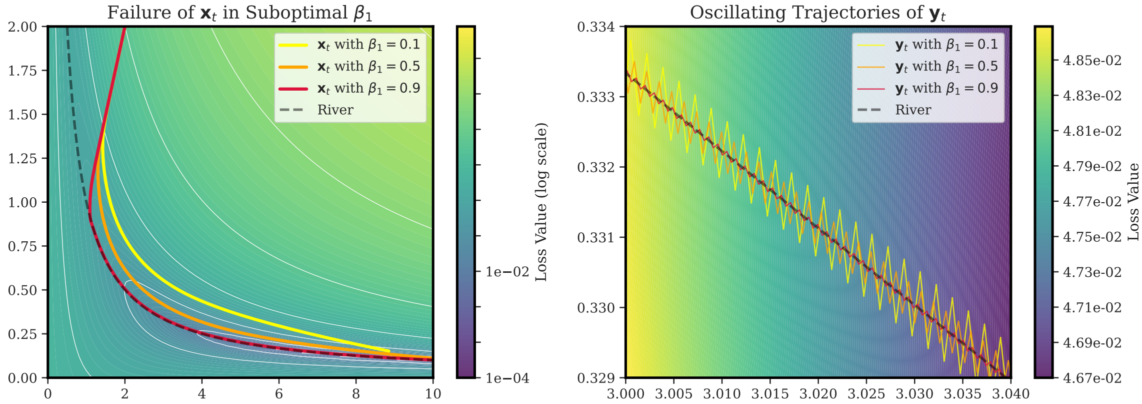

Figure 4: **SF-AdamW on Toy Model. Left:** The $\mathbf{x}_t$ iterates fail to follow the river for $\beta_1 \in \{0.1, 0.5\}$ (Observation 2). **Right:** The $\mathbf{y}_t$ iterates oscillate around the river but track it reliably on average, even for suboptimal values of $\beta_1$ (Observation 3). As $\beta_1$ increases, the oscillations shrink.

it. Moreover, the EWA of $\mathbf{y}_t$ consistently achieves lower loss than the raw $\mathbf{y}_t$ iterates across all momentum configurations—unlike the $\mathbf{x}_t$ iterates, where EWA offers no benefit for $\beta_1 = 0.95$. It illustrates that the EWA of $\mathbf{y}_t$ consistently remains closer to the river compared to the vanilla $\mathbf{y}_t$ iterates. This observation parallels the oscillatory behavior of $\mathbf{y}_t$ in the toy model, where the EWA would more closely align with the underlying river-like geometry.

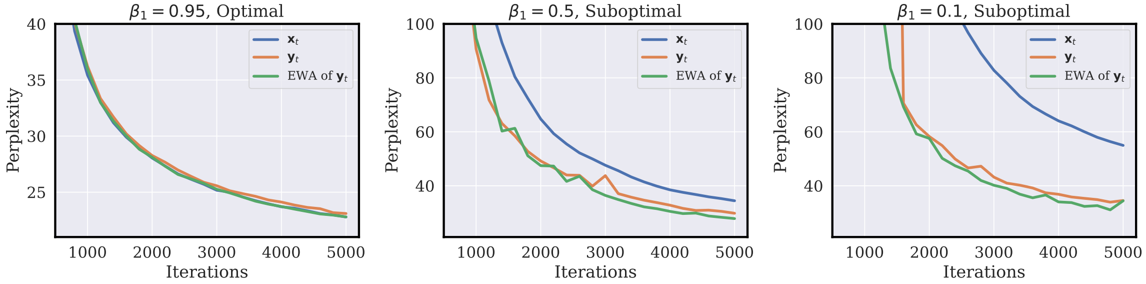

Figure 5: **Performance of $\mathbf{x}_t$, $\mathbf{y}_t$, and the EWA of $\mathbf{y}_t$ under varying $\beta_1$.** For suboptimal $\beta_1$, $\mathbf{y}_t$ outperforms $\mathbf{x}_t$, and across all momentum settings, the EWA of $\mathbf{y}_t$ achieves the lowest loss (Observation 3).

***Observation 3:*** In `SF-AdamW`, the $\mathbf{y}_t$ iterates remain well aligned with the river geometry of the loss landscape, even under suboptimal momentum settings, whereas $\mathbf{x}_t$ may deviate.

### B.3. Schedule-Free Methods at the Edge of Stability

We return to the toy model from Appendix B.1 to further analyze the optimization dynamics of the $\mathbf{y}_t$ iterates. Notably, the $\mathbf{y}_t$ sequence exhibits a period-two oscillation centered along the river (Figure 4). This behavior resembles the dynamics of full-batch GD at the *Edge of Stability* (EoS), where iterates oscillate along sharp directions while remaining globally stable. The EoS phenomenon was first identified empirically by Cohen et al. (2021) and has since been studied theoretically (Arora et al., 2022; Wang et al., 2022; Ahn et al., 2023; Damian et al., 2023; Song and Yun, 2023; Zhu et al., 2023). A key feature of this regime is that the sharpness—measured by the largest Hessian eigenvalue—stabilizes near the threshold $2/\gamma$.

We now make this connection precise by analyzing SF dynamics through the lens of EoS.

**Notation.** Let $f(\mathbf{w})$ denote the objective and $\mathbf{H}(\mathbf{w})$ its Hessian at $\mathbf{w}$. For brevity, write $\mathbf{H}_t := \mathbf{H}(\mathbf{w}_t)$. The largest eigenvalue $\lambda_1(\mathbf{H})$ is called the *sharpness*, and for a preconditioner $\mathbf{P}$, $\lambda_1(\mathbf{P}^{-1}\mathbf{H})$ is called the *preconditioned sharpness*.

We first the stability of Schedule-Free GD (`SF-GD`) on quadratic objectives, which serve as local Taylor approximations to neural network training losses. `SF-GD` is defined by (`SF`) with $\Delta_t \triangleq \nabla f(\mathbf{y}_t)$.

**Proposition 1 (Stability Threshold of `SF-GD`)** *Consider running* `SF-GD` *on a quadratic objective* $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{H}\mathbf{w} + \mathbf{g}^\top \mathbf{w} + c$. *If* $\lambda_1(\mathbf{H}) > \frac{2}{(1-\beta)\gamma}$, *then the iterates* $\{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$ *diverge.*

Notably, this stability threshold is scaled by a factor of $(1-\beta)^{-1}$ compared to the threshold for standard GD. This result is consistent with empirical observations from Defazio et al. (2024), which report that SF methods allow the use of larger LRs, particularly when $\beta$ is close to one.

Next, we extend the analysis to `SF-PrecondGDW`, defined by (`SF`) with $\Delta_t \triangleq \mathbf{P}^{-1}\nabla f(\mathbf{y}_t) + \lambda\mathbf{y}_t$, where $\mathbf{P}$ is a fixed, symmetric, and positive definite preconditioner, and $\lambda$ denotes the weight decay coefficient. This setting parallels the analysis of *Adaptive Edge of Stability* introduced by Cohen et al. (2022).

**Proposition 2 (Stability Threshold of `SF-PrecondGDW`)** *Consider running* `SF-PrecondGDW` *on* $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{H}\mathbf{w} + \mathbf{g}^\top \mathbf{w} + c$. *If* $\lambda_1(\mathbf{P}^{-1}\mathbf{H}) > \frac{2}{(1-\beta)\gamma} - \lambda$, *then the iterates* $\{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$ *diverge.*

Proofs of Propositions 1 and 2 are deferred to Section G.

`SF-AdamW` can be viewed as `SF-PrecondGDW` with with a slowly varying diagonal preconditioner $\mathbf{P}_t$. Hence, its stability is governed by the preconditioned sharpness $\lambda_1(\mathbf{P}_t^{-1}\mathbf{H}_t)$. As shown in Figure 6, the preconditioned sharpness at $\mathbf{y}_t$ iterates equilibrates near the stability threshold in both the toy model and CIFAR-10 experiments—exhibiting a typical EoS behavior.

***Observation 4:*** In full-batch settings, Schedule-Free methods operate at the Edge of Stability, with the (preconditioned) sharpness at $\mathbf{y}_t$ hovering around the stability threshold.

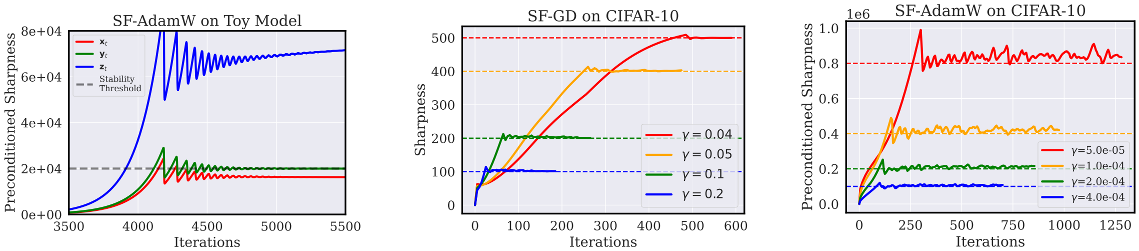

Figure 6: **The $\mathbf{y}_t$ iterates of Schedule-Free methods operate at the Edge of Stability.** Plots of (preconditioned) sharpness during full-batch training; dashed lines indicate stability thresholds. **Left:** Toy model trained using `SF-AdamW` with $(\beta_1, \beta_2) = (0.9, 0.99)$. **Middle, Right:** Fully connected network trained on a 5k subset of CIFAR-10 using `SF-GD` with $\beta = 0.9$ and `SF-AdamW` with $(\beta_1, \beta_2) = (0.95, 0.99)$; (preconditioned) sharpness is evaluated at the $\mathbf{y}_t$ iterates (Observation 4).

## B.4. A Reformulation of Schedule-Free Optimizer

Motivated by Observations 3 and 4, we now examine the dynamics of the $\mathbf{y}_t$ iterates. Following Defazio et al. (2024) and Morwani et al. (2025), define the momentum variable $\mathbf{m}_t := \frac{\mathbf{x}_t - \mathbf{z}_{t+1}}{\gamma}$. The Schedule-Free update in (SF) is then equivalent to

$$\begin{aligned} \mathbf{m}_t &= (1 - c_t)\,\mathbf{m}_{t-1} + \Delta_t, \\ \mathbf{y}_{t+1} &= \mathbf{y}_t - \gamma\big[\beta c_{t+1}\mathbf{m}_t + (1-\beta)\Delta_t\big], \end{aligned} \tag{$\texttt{SF}_{\mathbf{y}}$}$$

i.e. `SF` is simply a momentum-based optimizer update on $\mathbf{y}_t$.

The $\mathbf{x}_t$ iterate can then be expressed as

$$\mathbf{x}_t = \frac{(1-c_t)(1-\beta)\mathbf{x}_{t-1} + c_t\mathbf{y}_t}{(1-c_t)(1-\beta) + c_t}.$$

In other words, $\mathbf{x}_t$ is a weighted average of past $\mathbf{y}_t$'s. Hence, we arrive at the following conclusion.

***Observation 5:*** Schedule-Free implicitly performs weight averaging over momentum iterates *without* storing an extra model copy.

## B.5. Central Flow Analysis

Cohen et al. (2025) observe that, at the EoS, the time-averaged optimization trajectory follows a differential equation called the *central flow*, which characterizes the river that the dynamics trace during training. We adopt this framework to understand the magnitude of oscillation of $\mathbf{y}_t$ iterates of `SF-AdamW` along the river. In particular, we analyze the scalar surrogate `SF-ScalarAdam` with an adaptive preconditioner $\nu_t$ updated as $\nu_t = \beta_2\nu_{t-1} + (1-\beta_2)\|\nabla f(\mathbf{y}_t)\|^2$. Based on the reformulated update ($\texttt{SF}_{\mathbf{y}}$), and assuming that $c(t) = 1/t$ becomes negligible for sufficiently large $t$, the central flow equations are given by:

$$\frac{d\mathbf{y}}{dt} = -\frac{\gamma(1-\beta_1)}{\sqrt{\nu}}\Big[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2\nabla S(\mathbf{y})\Big], \qquad \frac{d\nu}{dt} = \frac{1-\beta_2}{\beta_2}\Big[\|\nabla f(\mathbf{y})\|^2 + \sigma^2 S(\mathbf{y})^2 - \nu\Big],$$

where $S(\mathbf{y}) = \lambda_1\big(\nabla^2 f(\mathbf{y})\big)$ and $\sigma^2$ is the steady-state variance of oscillations along the hill direction. Enforcing the stability condition $S(\mathbf{y})/\sqrt{\nu} = 2/[(1-\beta_1)\gamma]$ yields

$$\sigma^2(\mathbf{y}) = \frac{\langle \nabla S, -\nabla f\rangle + \frac{1-\beta_2}{\beta_2}\left[\frac{1}{4}S^2 - \frac{\|\nabla f\|^2}{(1-\beta_1)^2\gamma^2}\right]}{\frac{1}{2}\|\nabla S\|^2 + \frac{1-\beta_2}{(1-\beta_1)^2\beta_2\gamma^2}S^2}.$$

As $\beta_1$ increases, $\sigma^2$ decreases; thus, larger values of $\beta_1$ suppress oscillations along the hill directions, keeping the $\mathbf{y}_t$ iterates more closely aligned with the river—consistent with the empirical observation in Figure 4. A complete derivation, including the central flow of SF-GD, is provided in Section G.

## Appendix C. A Refined and Robust Schedule-Free Optimizer

While SF-AdamW achieves strong performance, it is highly sensitive to momentum hyperparameters and degrades under large batch sizes (Zhang et al., 2025; Morwani et al., 2025). Building on the insights from Appendix B, we revisit these issues and propose a refined variant that addresses them.

A key limitation in the vanilla (SF) setup ($c_t = 1/t$) is that $\beta$ simultaneously controls both (i) the momentum applied to $\mathbf{y}_t$ ($\text{SF}_{\mathbf{y}}$) and (ii) the implicit averaging window that defines $\mathbf{x}_t$:

$$\mathbf{x}_T = \sum_{t=1}^{T} \alpha_t \mathbf{y}_t, \quad \alpha_t := \frac{c_t}{(1-c_t)(1-\beta)+c_t} \prod_{s=t+1}^{T} \left[\frac{(1-c_s)(1-\beta)}{(1-c_s)(1-\beta)+c_s}\right].$$

When $\beta$ is small, the weights $\{\alpha_t\}$ become stretched, overemphasizing early iterates and preventing $\mathbf{x}_t$ from closely tracking the river (Figure 7). This also explains Observation 1, where applying EWA to $\mathbf{x}_t$ offers no benefit: since $\mathbf{x}_t$ is already a weighted average of $\mathbf{y}_t$, further averaging merely flattens the weights and weakens alignment with the river.

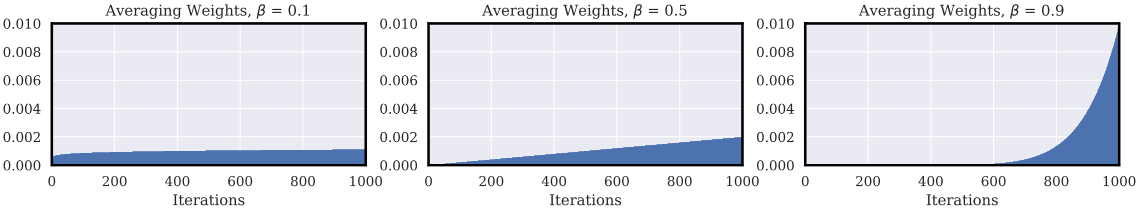

Figure 7: **Averaging Weights in SF methods.** Smaller values of $\beta$ flatten the averaging weights $\{\alpha_t\}$.

More fundamentally, $\beta$ plays a *dual role*: it controls both the momentum update of $\mathbf{y}_t$ and the width of the averaging window for $\mathbf{x}_t$. The optimal value for each may differ, and this mismatch can hinder performance. In large-batch training, a narrower window (i.e., larger $\beta$) is preferred, but this also slows $\mathbf{y}_t$ updates, as both $\beta c_{t+1}$ and $(1-\beta)$ in ($\text{SF}_{\mathbf{y}}$) become vanishingly small when $t$ is large.

**Our Refined SF Method.** We introduce an additional parameter $C$ and redefine $c_t = 1/t$ in (SF) as

$$c_t = \frac{(1-\beta)C}{t}, \quad \text{which then leads to} \quad \alpha_t \approx \frac{C}{T}\left(\frac{t}{T}\right)^{C-1}.$$

The full derivation is given in Appendix G.4, and pseudocode is provided in Algorithm 2. As shown in Figures 13 and 14, the weights $\{\alpha_t\}$ closely follow the theoretical approximation across different values of $\beta$, $C$, and $T$.

This modification makes the averaging weights $\{\alpha_t\}$ depend *solely* on $C$, allowing $\beta$ to independently control the momentum on $\mathbf{y}_t$. In other words, $C$ *decouples* the momentum and averaging behavior.

**Empirical gains.** We evaluate the performance of our refinement to `SF-AdamW` using the experimental setup as in Section 3. Keeping all other hyperparameters fixed from the tuned vanilla configuration and varying only $C$ (with $C = 1/(1-\beta_1)$ recovering the original method), we observe:

- **Momentum robustness.** For $\beta_1 \in \{0.1, 0.5\}$, the refined `SF-AdamW` allows $\mathbf{x}_t$ to match or outperform $\mathbf{y}_t$, in contrast to the underperformance of $\mathbf{x}_t$ reported in Observation 3 (Figure 8-left).
- **Improved best-case performance.** In the best vanilla setup ($\beta_1 = 0.95$), setting $C = 200$ leads to further reductions in validation loss; similar gains are observed with $\beta_1 = 0.9$ (Figure 8-middle).
- **Large-batch setting.** With 2M-token batches, vanilla `SF-AdamW` ($\beta_1 = 0.98$) lags behind `AdamW` with cosine schedule, whereas setting $C = 200$ matches its final performance (Figure 8-right).

These results show that decoupling momentum and averaging via introducing $C$ eliminates the tradeoff inherent in vanilla SF, yielding a more robust and scalable optimizer.

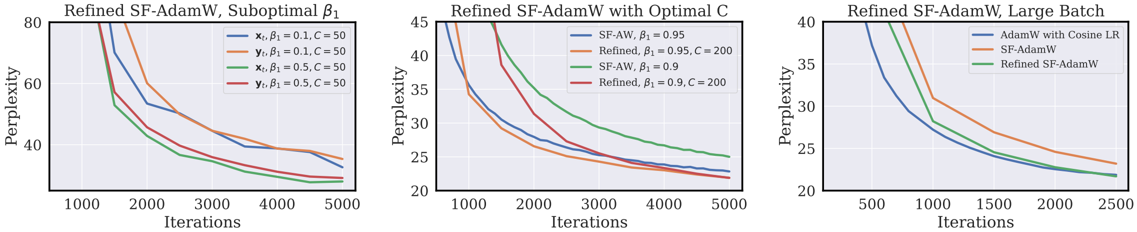

Figure 8: **Refined SF-AdamW. Left:** Performance of $\mathbf{x}_t$ and $\mathbf{y}_t$ iterates using the refined `SF-AdamW` with $\beta_1 \in \{0.1, 0.5\}$ and $C = 50$. **Middle:** Refined `SF-AdamW` with $\beta_1 \in \{0.95, 0.9\}$ and $C = 200$ achieves improved performance over the best vanilla `SF-AdamW` run. **Right:** Under a large batch size (2M tokens), vanilla `SF-AdamW` with $\beta_1 = 0.98$ underperforms compared to `AdamW` with a cosine schedule, while the refined `SF-AdamW`—with only a sweep over $C = 200$—matches its final performance.

## Appendix D. Pseudocode for Schedule-Free AdamW

For completeness, we present the pseudocode for the Schedule-Free AdamW (`SF-AdamW`) algorithm in algorithm 1, along with our proposed refinement in algorithm 2. The refinement introduces a decoupling parameter $C$ to independently control the averaging window, addressing the coupling issue discussed in Appendix C.

**Algorithm 1** `SF-AdamW` (Defazio et al., 2024)

```
Input: $x_1$, learning rate $\gamma$, decay $\lambda$, warmup steps $T_{\text{warmup}}, \beta_1, \beta_2, \epsilon$
$z_1 = x_1$
$v_0 = 0$
for $t = 1$ to $T$ do
    $y_t = (1 - \beta_1) z_t + \beta_1 x_t$
    $g_t \in \partial f(y_t, \zeta_t)$
    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{v}_t = v_t / (1 - \beta_2^t)$
    $\gamma_t = \gamma \min(1, t/T_{\text{warmup}})$
    $z_{t+1} = z_t - \gamma_t g_t / (\sqrt{\hat{v}_t} + \epsilon) - \gamma_t \lambda y_t$
    $c_{t+1} = \frac{\gamma_t^2}{\sum_{i=1}^{t} \gamma_i^2}$
    $x_{t+1} = (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1}$
Return $x_{T+1}$
```

**Algorithm 2** Refined `SF-AdamW` (with decoupling parameter $C$)

```
Input: $x_1$, learning rate $\gamma$, decay $\lambda$, warmup steps $T_{\text{warmup}}, \beta_1, \beta_2, \epsilon$, decoupling parameter $C$
$z_1 = x_1$
$v_0 = 0$
for $t = 1$ to $T$ do
    $y_t = (1 - \beta_1) z_t + \beta_1 x_t$
    $g_t \in \partial f(y_t, \zeta_t)$
    $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{v}_t = v_t / (1 - \beta_2^t)$
    $\gamma_t = \gamma \min(1, t/T_{\text{warmup}})$
    $z_{t+1} = z_t - \gamma_t g_t / (\sqrt{\hat{v}_t} + \epsilon) - \gamma_t \lambda y_t$
    $c_{t+1} = \min \left\{ \frac{\gamma_t^2}{\sum_{i=1}^{t} \gamma_i^2} \cdot (1 - \beta_1) C,\ 1 \right\}$
    $x_{t+1} = (1 - c_{t+1})\, x_t + c_{t+1} z_{t+1}$
Return $x_{T+1}$
```

## Appendix E. Experimental Details

### E.1. Language Model Experiments

**Codebase.** All language model experiments are implemented using the public `llm-baselines` codebase: https://github.com/epfml/llm-baselines. Our only modification is the addition of custom optimizer implementations to support the Schedule-Free and refined Schedule-Free methods. All other components (model, data pipeline, logging) remain unchanged.

**Architectures.** Our main experiments use a 124M-parameter LLaMA-style decoder-only transformer with SwiGLU activations (Shazeer, 2020), RoPE embeddings (Su et al., 2024), RMSNorm (Zhang and Sennrich, 2019), and alternating attention/MLP blocks (12 layers, 12 attention heads, hidden dimension 768). Additional results in Section F verify our findings with a 124M-parameter GPT-2-style transformer. Both architectures are implemented in `llm-baselines` with standard design choices.

**Datasets.** Our main experiments use the 6B-token subset of the SlimPajama dataset (Soboleva et al., 2023), available on Hugging Face.[1] We tokenize with the GPT-2 tokenizer (Radford et al., 2019), which has a vocabulary size of 50,304. In Section F, we also validate our main findings on the OpenWebText2 dataset (Gao et al., 2020),[2] using the same setup.

**Training details.** We train models using `AdamW` and `SF-AdamW`, with a short warmup phase comprising 5% of total steps. For large-batch runs with cosine decay, the learning rate is annealed to 10% of its peak. Main experiments use a batch size of 1,024 sequences of context length 512 tokens (0.5M tokens total), trained for 5,000 steps, amounting to roughly 2.5B tokens ($\sim 1\times$ Chinchilla scale). Large-batch experiments use a 2M-token batch size and 2,500 steps ($\sim$5B tokens, or $\sim 2\times$ Chinchilla scale) to evaluate efficiency in the overtrained regime. Validation is performed during training using 3,200 sequences of context length 512 tokens ($\sim$1.6M tokens) to compute validation loss (perplexity) curves. For computing EWA, we use a decay factor of 0.99 for all experiments.

**Hyperparameters.** We fix the weight decay to 0.1 in all experiments. Gradient clipping is set to 1.0 for `AdamW` and disabled (0.0) for `SF-AdamW`. We perform sweeps over the learning rate, momentum parameters, and the decoupling parameter $C$ (for refined `SF-AdamW`). Full configurations are provided in Tables 1 to 4.

**Compute Resources.** All experiments are conducted on a single node with 8 NVIDIA A6000 GPUs (48GB VRAM each) using data-parallel training. A typical full 5,000-step run with a 0.5M-token batch size takes approximately 3 hours.

### E.2. Edge of Stability Experiments on CIFAR-10

We provide the experimental setup for Figure 6 in Appendix B.3, where we study whether Schedule-Free methods operate at the Edge of Stability in a deep learning setting.

Our experiments build on the public `edge-of-stability` codebase,[3] modifying only the optimizer to incorporate Schedule-Free methods. The model is a 3-layer MLP with hidden

[1] https://huggingface.co/datasets/DKYoon/SlimPajama-6B

[2] https://huggingface.co/datasets/segyges/OpenWebText2

[3] https://github.com/locuslab/edge-of-stability

| Optimizer | Learning Rate | $(\beta_1, \beta_2)$ | $C$ (Refined SF) |
|---:|:---:|:---:|:---:|
| AdamW | {5e-4, **1e-3**, 2e-3, 5e-3} | {**(0.9, 0.95)**, (0.95, 0.99)} | – |
| SF-AdamW | {1e-3, **2e-3**, 5e-3} | {(0.9, 0.99), **(0.95, 0.99)**, (0.98, 0.99)} | – |
| Refined SF-AdamW | 2e-3 | (0.95, 0.99) | {20, 50, 100, **200**, 500} |

Table 1: **Hyperparameter sweep: Main experiments.** Grid of hyperparameters used in our main experiments (SlimPajama-6B, 0.5M-token batch size), including learning rates, momentum pairs $(\beta_1, \beta_2)$, and the decoupling parameter $C$ (for Refined SF-AdamW). Bold entries indicate the best-performing configuration for each optimizer.

| Optimizer | Learning Rate | $(\beta_1, \beta_2)$ | $C$ (Refined SF) |
|---:|:---:|:---:|:---:|
| AdamW (Cosine LR) | {5e-4, **1e-3**, 2e-3, 5e-3} | {**(0.9, 0.95)**, (0.95, 0.99)} | – |
| SF-AdamW | {1e-3, **2e-3**, 5e-3} | {(0.9, 0.99), (0.95, 0.99), **(0.98, 0.99)**} | – |
| Refined SF-AdamW | 2e-3 | (0.98, 0.99) | {20, 50, 100, **200**, 500} |

Table 2: **Hyperparameter sweep: Large-batch experiments.** Grid of hyperparameters used in our large-batch experiments (SlimPajama-6B, 2M-token batch size), including learning rates, momentum pairs $(\beta_1, \beta_2)$, and the decoupling parameter $C$ (for Refined SF-AdamW). Bold entries indicate the best-performing configuration.

| Optimizer | Learning Rate | $(\beta_1, \beta_2)$ | $C$ (Refined SF) |
|---:|:---:|:---:|:---:|
| AdamW | {5e-4, **1e-3**, 2e-3} | {**(0.9, 0.95)**, (0.95, 0.99)} | – |
| SF-AdamW | {1e-3, **2e-3**, 5e-3} | {(0.9, 0.99), **(0.95, 0.99)**, (0.98, 0.99)} | – |
| Refined SF-AdamW | 2e-3 | (0.95, 0.99) | {50, 100, **200**, 500} |

Table 3: **Hyperparameter sweep: OpenWebText2 experiments.** Grid of hyperparameters used in our additional experiments on OpenWebText2 (0.5M-token batch size), including learning rates, momentum pairs $(\beta_1, \beta_2)$, and the decoupling parameter $C$ (for Refined SF-AdamW). Bold entries indicate the best-performing configuration.

| Optimizer | Learning Rate | $(\beta_1, \beta_2)$ | $C$ (Refined SF) |
|---:|:---:|:---:|:---:|
| AdamW (Cosine LR) | {5e-4, 1e-3, 2e-3, **5e-3**} | {**(0.9, 0.95)**, (0.95, 0.99)} | – |
| SF-AdamW | {5e-4, 1e-3, **2e-3**, 5e-3} | {(0.95, 0.99), **(0.98, 0.99)**} | – |
| Refined SF-AdamW | {2e-3, **5e-3**, 1e-2} | (0.98, 0.99) | {20, 50, 100, 200, **500**, 1000} |

Table 4: **Hyperparameter sweep: OpenWebText2 large-batch experiments.** Grid of hyperparameters used in our additional large-batch experiments on OpenWebText2 (2M-token batch size), including learning rates, momentum pairs $(\beta_1, \beta_2)$, and the decoupling parameter $C$ (for Refined SF-AdamW). Bold entries indicate the best-performing configuration.

width 200 and tanh activations, trained on the first 5,000 samples of CIFAR-10 using mean squared error (MSE) loss.

For SF-GD, we fix momentum $\beta = 0.9$ and vary the learning rate, training each run until the loss reaches 0.02. For SF-AdamW, we fix $\beta_1 = 0.95$, $\beta_2 = 0.99$, and weight decay at 0.1, and vary the learning rate, training until the loss reaches 0.05.

## Appendix F. Additional Results

In this section, we present additional experiments on the OpenWebText2 dataset using a 124M-parameter GPT-2 style decoder-only transformer, replicating the setups from the main text. As summarized below, the results confirm that our main findings (Observations 1, 2 and 3) generalize across both datasets and architectures.

- **Observation 1:** Following the same procedure as in the main text, we perform a grid search and apply a LR decay phase at each checkpoint. Throughout training, we track the EWA of the iterates $\mathbf{x}_t$, and at each checkpoint, we also execute a short LR decay phase using the AdamW optimizer. In OpenWebText2 setup, the optimal hyperparameters are $(\beta_1, \beta_2) = (0.9, 0.95)$ with LR 1e-3 for AdamW, and $(\beta_1, \beta_2) = (0.95, 0.99)$ with LR 2e-3 for SF-AdamW. We observe that our main findings in Observation 1 are successfully reproduced (see Figure 9).
- **Observation 2:** We also repeat our experiment using SF-AdamW with suboptimal $\beta_1 \in \{0.1, 0.5\}$. We similarly observe that the decay phase improves performance, implying that suboptimal momentum disrupts the optimizer's ability to follow the river (see Figure 10).
- **Observation 3.** Using the same experimental runs with $\beta_1 \in \{0.1, 0.5, 0.95\}$, we similarly evaluate the loss at the $\mathbf{y}_t$ iterates and its EWA. We observe that the loss at $\mathbf{y}_t$ is consistently lower than that at $\mathbf{x}_t$ for suboptimal $\beta_1$, which is analogous to Observation 3. Moreover, the EWA of $\mathbf{y}_t$ achieves lower loss than $\mathbf{y}_t$ iterates (see Figure 11).
- **Results on Refined SF-AdamW:** We evaluate the performance of our refined SF-AdamW following the same procedure as in the main text. For a lower momentum setting ($\beta_1 = 0.5$), the iterate $\mathbf{x}_t$ consistently outperforms $\mathbf{y}_t$. In the best-case of SF-AdamW ($\beta_1 = 0.95$ with LR 2e-3), setting $C = 200$ yields further performance gains. Also, even with a suboptimal momentum value ($\beta_1 = 0.9$), comparable performance can be recovered by using $C = 50$.

In the large-batch experiments, `AdamW` with cosine LR schedule ($(\beta_1, \beta_2) = (0.9, 0.95)$ with LR 5e-3) outperforms `SF-AdamW` with a constant LR ($(\beta_1, \beta_2) = (0.98, 0.99)$ with LR 2e-3). Our refined `SF-AdamW` ($(\beta_1, \beta_2) = (0.98, 0.99), C = 500$ with LR 5e-3), however, matches this performance (see Figure 12).

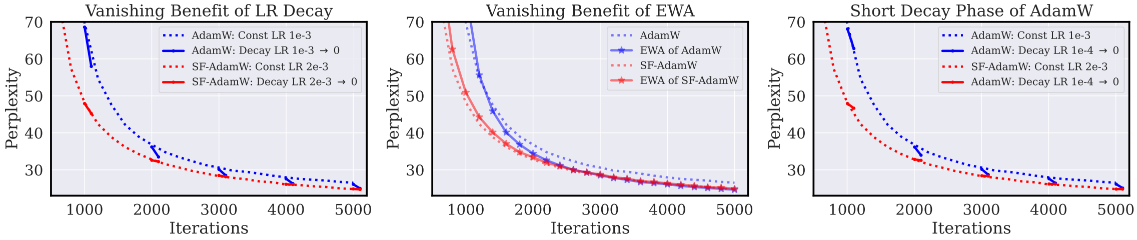

Figure 9: **OpenWebText2 Experiment: SF-AdamW closely follows the river, unlike AdamW. Left, Middle:** While `AdamW` benefits from linear LR decay and EWA, `SF-AdamW` shows no improvement from either. **Right:** A short decay phase of `AdamW` (with linear LR decay from 1e-4 to 0) leads to a sharp loss drop for `AdamW`, but has no effect when applied to the `SF-AdamW` trajectory—suggesting that `SF-AdamW` already tracks the river throughout training (Observation 1).

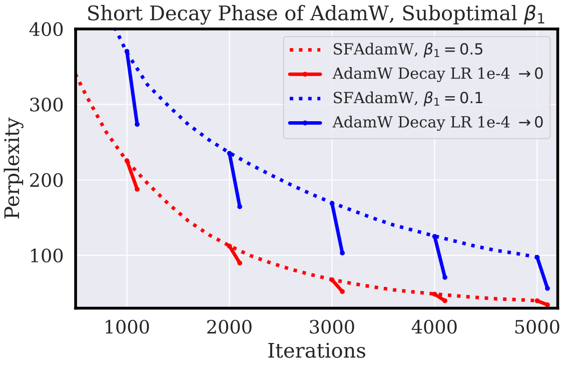

Figure 10: **OpenWebText2 Experiment: SF-AdamW with suboptimal momentum fails to follow the river.** A short decay phase of `AdamW` applied to `SF-AdamW` checkpoints with $\beta_1 \in \{0.1, 0.5\}$ results in a sharp loss drop, unlike the case with $\beta_1 = 0.95$ (Observation 2).

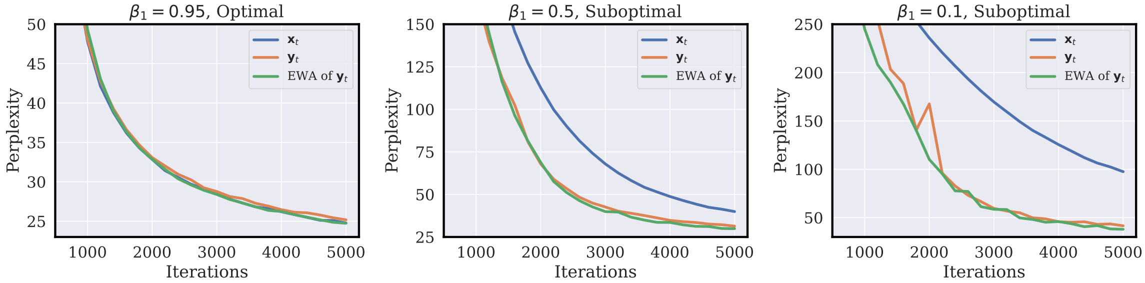

Figure 11: **OpenWebText2 Experiment: Performance of $\mathbf{x}_t$, $\mathbf{y}_t$, and the EWA of $\mathbf{y}_t$ under varying $\beta_1$.** For suboptimal $\beta_1$, $\mathbf{y}_t$ outperforms $\mathbf{x}_t$, and across all momentum settings, the EWA of $\mathbf{y}_t$ achieves the lowest loss (Observation 3).

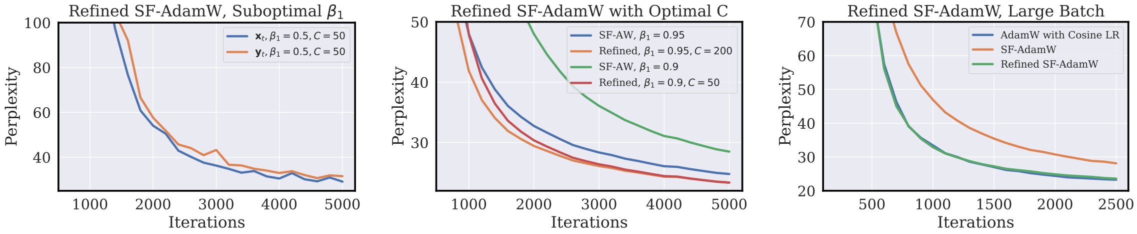

Figure 12: **OpenWebText2 Experiment: Refined SF-AdamW. Left:** Performance of $\mathbf{x}_t$ and $\mathbf{y}_t$ iterates using the refined `SF-AdamW` with $\beta_1 = 0.5$ and $C = 50$. **Middle:** Refined `SF-AdamW` with $(\beta_1, C) \in \{(0.95, 200), (0.9, 50)\}$ achieves improved performance over the best vanilla `SF-AdamW` run. **Right:** Under a large batch size (2M tokens), vanilla `SF-AdamW` with $\beta_1 = 0.98$ underperforms compared to `AdamW` with a cosine schedule, while the refined `SF-AdamW` with $C = 500$ matches its final performance.

# Appendix G. Omitted Derivations and Proofs

## G.1. Proof of Stability Threshold of Schedule-Free Optimizer

### G.1.1. Proof of Proposition 1

**Proposition 1 (Stability Threshold of SF-GD)** *Consider running* SF-GD *on a quadratic objective* $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{H}\mathbf{w} + \mathbf{g}^\top \mathbf{w} + c$. *If* $\lambda_1(\mathbf{H}) > \frac{2}{(1-\beta)\gamma}$, *then the iterates* $\{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$ *diverge.*

**Proof** To begin with, we revisit the update rule of SF-GD, given by

$$\begin{aligned} \mathbf{x}_t &= (1-c_t)\,\mathbf{x}_{t-1} + c_t\,\mathbf{z}_t, \\ \mathbf{y}_t &= (1-\beta)\,\mathbf{z}_t + \beta\,\mathbf{x}_t, \\ \mathbf{z}_{t+1} &= \mathbf{z}_t - \gamma\nabla f(\mathbf{y}_t). \end{aligned} \tag{SF-GD}$$

On a quadratic objective, we get $\nabla f(\mathbf{w}) = \mathbf{H}\mathbf{w} + \mathbf{g}$. By substituting this and combining the last two relations, we get

$$\begin{aligned} \mathbf{z}_{t+1} &= \mathbf{z}_t - \gamma(\mathbf{H}\mathbf{y}_t + \mathbf{g}) \\ &= \mathbf{z}_t - \gamma((1-\beta)\mathbf{H}\mathbf{z}_t + \beta\mathbf{H}\mathbf{x}_t + \mathbf{g}) \\ &= (\mathbf{I} - \gamma(1-\beta)\mathbf{H})\mathbf{z}_t - \beta\gamma\mathbf{H}\mathbf{x}_t - \gamma\mathbf{g}. \end{aligned}$$

By substituting this to the first relation, we get a recurrence relation governing $\mathbf{x}_t$ as follows:

$$\begin{aligned} \mathbf{x}_{t+1} =& (1-c_{t+1})\mathbf{x}_t + c_{t+1}\mathbf{z}_{t+1} \\ =& (1-c_{t+1})\mathbf{x}_t + c_{t+1}((\mathbf{I} - \gamma(1-\beta)\mathbf{H})\mathbf{z}_t - \beta\gamma\mathbf{H}\mathbf{x}_t - \gamma\mathbf{g}) \\ =& ((1-c_{t+1})\mathbf{I} - \beta\gamma c_{t+1}\mathbf{H})\mathbf{x}_t + c_{t+1}(\mathbf{I} - \gamma(1-\beta)\mathbf{H})\mathbf{z}_t - \gamma c_{t+1}\mathbf{g} \\ =& ((1-c_{t+1})\mathbf{I} - \beta\gamma c_{t+1}\mathbf{H})\mathbf{x}_t + c_{t+1}(\mathbf{I} - \gamma(1-\beta)\mathbf{H})\left(\frac{1}{c_t}\mathbf{x}_t - \left(\frac{1}{c_t} - 1\right)\mathbf{x}_{t-1}\right) - \gamma c_{t+1}\mathbf{g} \\ =& \left(\left(1 + \frac{c_{t+1}}{c_t} - c_{t+1}\right)\mathbf{I} - \gamma c_{t+1}\left(\frac{1-\beta}{c_t} + \beta\right)\mathbf{H}\right)\mathbf{x}_t \\ &+ \left(\left(c_{t+1} - \frac{c_{t+1}}{c_t}\right)\mathbf{I} - \gamma\left(c_{t+1} - \frac{c_{t+1}}{c_t}\right)(1-\beta)\mathbf{H}\right)\mathbf{x}_{t-1} - \gamma c_{t+1}\mathbf{g}. \end{aligned} \tag{1}$$

Define $(\mathbf{q}, a) := (\mathbf{q}, \lambda_1(\mathbf{H}))$ to be the largest eigenvector/eigenvalue pair of $\mathbf{H}$ and $\tilde{x}_t = \mathbf{q}^\top\mathbf{x}_t + \frac{1}{a}\mathbf{q}^\top\mathbf{g}$. Then the sequence $\{\mathbf{q}^\top\mathbf{x}_t\}$ diverges if and only if the sequence $\{\tilde{x}_t\}$ diverges. By multiplying $\mathbf{q}^\top$ on both sides of Equation (1), we get

$$\begin{aligned} \mathbf{q}^\top\mathbf{x}_{t+1} = & \left(1 + \frac{c_{t+1}}{c_t} - c_{t+1} - \gamma c_{t+1}\left(\frac{1-\beta}{c_t} + \beta\right)a\right)\mathbf{q}^\top\mathbf{x}_t \\ &+ \left(c_{t+1} - \frac{c_{t+1}}{c_t}\right)(1 - \gamma(1-\beta)a)\mathbf{q}^\top\mathbf{x}_{t-1} - \gamma c_{t+1}\mathbf{q}^\top\mathbf{g}, \end{aligned}$$

from $\mathbf{q}^\top\mathbf{H} = a\mathbf{q}^\top$. From the definition of $\tilde{x}_t$, we get

$$\tilde{x}_{t+1} = \left(1 + \frac{c_{t+1}}{c_t} - c_{t+1} - \gamma c_{t+1}\left(\frac{1-\beta}{c_t} + \beta\right)a\right)\tilde{x}_t + \left(c_{t+1} - \frac{c_{t+1}}{c_t}\right)(1 - \gamma(1-\beta)a)\tilde{x}_{t-1},$$

which is a linear time-varying second order difference equation governing $\tilde{x}_t$.

Its asymptotic behavior is governed by the limiting recurrence relation:

$$\bar{x}_{t+1} = (2 - a(1-\beta)\gamma)\bar{x}_t + (a(1-\beta)\gamma - 1)\bar{x}_{t-1}.$$

Two roots of this recurrence relation are given by

$$\begin{aligned}\lambda_1 &= \frac{2 - a(1-\beta)\gamma + \sqrt{(2-a(1-\beta)\gamma)^2 - 4(a(1-\beta)\gamma - 1)}}{2}\\ \lambda_2 &= \frac{2 - a(1-\beta)\gamma - \sqrt{(2-a(1-\beta)\gamma)^2 - 4(a(1-\beta)\gamma - 1)}}{2}.\end{aligned}$$

If $2 < a(1-\beta)\gamma < 4 + 2\sqrt{2}$, then

$$\begin{aligned}|\lambda_1|^2 = |\lambda_2|^2 &= \left(\frac{2 - a(1-\beta)\gamma}{2}\right)^2 - \frac{(2-a(1-\beta)\gamma)^2 - 4(a(1-\beta)\gamma - 1)}{4}\\ &= a(1-\beta)\gamma - 1 > 1,\end{aligned}$$

which implies $\bar{x}_t$ diverges.

If $a(1-\beta)\gamma \geq 4 + 2\sqrt{2}$, then $\lambda_2$ can be regarded as a real valued function with respect to $a(1-\beta)\gamma$. Since $\lambda_2$ is decreasing and $\lambda_2 < -1$ when $a(1-\beta)\gamma = 4 + 2\sqrt{2}$, we get $\lambda_2 < -1$ when $a(1-\beta)\gamma \geq 4 + 2\sqrt{2}$, which implies that $\bar{x}_t$ also diverges.

Since we take $\gamma > 0$ and $0 \leq \beta < 1$, the condition $a > \frac{2}{(1-\beta)\gamma}$ implies diverging $\bar{x}_t$ as well as $\tilde{x}_t$. ■

### G.1.2. Proof of Proposition 2

To show Proposition 2, we first prove the following reparameterization lemma.

**Lemma 1** *Define* `SF-PrecondGD` *by* (`SF`) *with* $\Delta_t \triangleq \mathbf{P}^{-1}\nabla f(\mathbf{y}_t)$. *Let* $\{\mathbf{x}_t\}$ *denotes the iterates of* `SF-PrecondGD` *on the objective* $f(\mathbf{w})$, *and let* $\{\tilde{\mathbf{x}}_t\}$ *denote the iterates of* `SF-GD` *on the reparameterized objective* $\tilde{f}(\mathbf{w}) = f(\mathbf{P}^{-1/2}\mathbf{w})$ *with initialization* $\tilde{\mathbf{x}}_1 = \mathbf{P}^{1/2}\mathbf{x}_1$. *Then, we have* $\tilde{\mathbf{x}}_t = \mathbf{P}^{1/2}\mathbf{x}_t$ *for all steps* $t$.

**Proof** We claim that the equivalence $(\tilde{\mathbf{x}}_t, \tilde{\mathbf{y}}_t, \tilde{\mathbf{z}}_t) = (\mathbf{P}^{1/2}\mathbf{x}_t, \mathbf{P}^{1/2}\mathbf{y}_t, \mathbf{P}^{1/2}\mathbf{z}_t)$ holds for all $t$, where $(\tilde{\mathbf{x}}_t, \tilde{\mathbf{y}}_t, \tilde{\mathbf{z}}_t)$ denotes the iterates on the reparametrized objective.

For $t = 1$, it holds from the definition. Assume that the equivalence holds at $t$. Then, the update for $\tilde{\mathbf{z}}_{t+1}$ is given by

$$\begin{aligned}\tilde{\mathbf{z}}_{t+1} &= \tilde{\mathbf{z}}_t - \gamma\nabla\tilde{f}(\tilde{\mathbf{y}}_t)\\ &= \tilde{\mathbf{z}}_t - \gamma\mathbf{P}^{-1/2}\nabla f(\mathbf{P}^{-1/2}\tilde{\mathbf{y}}_t)\\ &= \mathbf{P}^{1/2}\mathbf{z}_t - \gamma\mathbf{P}^{-1/2}\nabla f(\mathbf{y}_t) \quad \text{(inductive hypothesis)}\\ &= \mathbf{P}^{1/2}(\mathbf{z}_t - \gamma\mathbf{P}^{-1}\nabla f(\mathbf{y}_t))\\ &= \mathbf{P}^{1/2}\mathbf{z}_{t+1}.\end{aligned}$$

Meanwhile,

$$\begin{aligned}\tilde{\mathbf{x}}_{t+1} &= (1-c_{t+1})\tilde{\mathbf{x}}_t + c_t\tilde{\mathbf{z}}_{t+1}\\ &= \mathbf{P}^{1/2}((1-c_{t+1})\mathbf{x}_t + c_t\mathbf{z}_{t+1})\\ &= \mathbf{P}^{1/2}\mathbf{x}_{t+1}\\ \tilde{\mathbf{y}}_{t+1} &= (1-\beta)\tilde{\mathbf{z}}_{t+1} + \beta\tilde{\mathbf{x}}_{t+1}\\ &= \mathbf{P}^{1/2}((1-\beta)\mathbf{z}_{t+1} + \beta\mathbf{x}_{t+1})\\ &= \mathbf{P}^{1/2}\mathbf{y}_{t+1},\end{aligned}$$

which proves the claim. ■

**Proposition 2 (Stability Threshold of `SF-PrecondGDW`)** *Consider running* `SF-PrecondGDW` *on* $f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top\mathbf{H}\mathbf{w} + \mathbf{g}^\top\mathbf{w} + c$. *If* $\lambda_1(\mathbf{P}^{-1}\mathbf{H}) > \frac{2}{(1-\beta)\gamma} - \lambda$, *then the iterates* $\{(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t)\}$ *diverge.*

**Proof** Recall that `SF-PrecondGDW` is defined by (SF) with $\Delta_t \triangleq \mathbf{P}^{-1}\nabla f(\mathbf{y}_t) + \lambda\mathbf{y}_t$, which is identical to

$$\mathbf{P}^{-1}\nabla f(\mathbf{y}_t) + \lambda\mathbf{y}_t = \mathbf{P}^{-1}\nabla g(\mathbf{y}_t),$$

where $g(\mathbf{w}) = f(\mathbf{w}) + \frac{1}{2}\lambda\|\mathbf{P}^{1/2}\mathbf{w}\|^2$. Therefore, `SF-PrecondGDW` is identical to `SF-PrecondGD` on the objective $g(\mathbf{w})$.

Let $\{\tilde{\mathbf{x}}_t\}$ be the iterates of `SF-GD` on the reparameterized objective $\tilde{g}(\mathbf{w}) = g(\mathbf{P}^{-1/2}\mathbf{w})$ with initialization $\tilde{\mathbf{x}}_1 = \mathbf{P}^{1/2}\mathbf{x}_1$. From Theorem 1, $\{\mathbf{x}_t\}$, the iterates of `SF-PrecondGDW`, satisfy $\tilde{\mathbf{x}}_t = \mathbf{P}^{1/2}\mathbf{x}_t$, which implies that if $\tilde{\mathbf{x}}_t$ diverges then $\mathbf{x}_t$ also diverges.

From Proposition 1, if $\lambda_1(\mathbf{P}^{-1}\mathbf{H} + \lambda\mathbf{I}) > \frac{2}{(1-\beta)\gamma}$, then $\tilde{\mathbf{x}}_t$ diverges. This proves the claim. ■

## G.2. Deriving the Reformulation of Schedule-Free Optimizer

Define a momentum variable by

$$\mathbf{m}_t \triangleq \frac{\mathbf{x}_t - \mathbf{z}_{t+1}}{\gamma},$$

Then, it follows that

$$\mathbf{m}_t = \frac{\mathbf{x}_t - \mathbf{z}_{t+1}}{\gamma} = \frac{\mathbf{x}_t - \mathbf{z}_t}{\gamma} + \Delta_t = (1-c_t)\frac{\mathbf{x}_{t-1} - \mathbf{z}_t}{\gamma} + \Delta_t = (1-c_t)\mathbf{m}_{t-1} + \Delta_t$$

We can also write the update for $\mathbf{y}_t$:

$$\begin{aligned}\mathbf{y}_{t+1} &= (1-\beta)\,\mathbf{z}_{t+1} + \beta\,\mathbf{x}_{t+1}\\ &= (1-\beta)\,(\mathbf{z}_t - \gamma\Delta_t) + \beta\,((1-c_{t+1})\mathbf{x}_t + c_{t+1}\mathbf{z}_{t+1})\\ &= (1-\beta)\mathbf{z}_t + \beta\mathbf{x}_t - (1-\beta)\gamma\Delta_t + \beta c_{t+1}(\mathbf{z}_{t+1} - \mathbf{x}_t)\\ &= \mathbf{y}_t - \gamma\,[\beta c_{t+1}\mathbf{m}_t + (1-\beta)\Delta_t]\end{aligned}$$

Hence, the equivalent way of writing the schedule-free update rule is the following:

$$\mathbf{m}_t = (1 - c_t)\mathbf{m}_{t-1} + \Delta_t.$$
$$\mathbf{y}_{t+1} = \mathbf{y}_t - \gamma\beta c_{t+1}\mathbf{m}_t - \gamma(1-\beta)\Delta_t.$$

Given this, we can rewrite $\mathbf{x}_t$ as a weighted average of $\mathbf{y}_t$:

$$\begin{aligned}
\mathbf{x}_{t+1} &= (1 - c_{t+1})\mathbf{x}_t + c_{t+1}\mathbf{z}_{t+1}, \\
\mathbf{x}_{t+1} &= (1 - c_{t+1})\mathbf{x}_t + c_{t+1}\left(\frac{\mathbf{y}_{t+1} - \beta\mathbf{x}_{t+1}}{1-\beta}\right), \\
\mathbf{x}_{t+1} &= \frac{(1 - c_{t+1})(1-\beta)\mathbf{x}_t + c_{t+1}\mathbf{y}_{t+1}}{(1 - c_{t+1})(1-\beta) + c_{t+1}}.
\end{aligned}$$

## G.3. Deriving the Central Flow of Schedule-Free Optimizer

### G.3.1. Deriving the Central Flow of Schedule-Free GD

We begin with the reformulated update rule for SF-GD, as derived from ($\mathrm{SF}_{\mathbf{y}}$):

$$\begin{aligned}
\mathbf{m}_t &= (1 - c_t)\mathbf{m}_{t-1} + \nabla f(\mathbf{y}_t), \\
\mathbf{y}_{t+1} &= \mathbf{y}_t - \gamma\beta c_{t+1}\mathbf{m}_t - \gamma(1-\beta)\nabla f(\mathbf{y}_t).
\end{aligned}$$

As in gradient descent, stable training dynamics are often well-approximated by their continuous-time analogs. We can therefore define a corresponding *stable flow* for SF-GD:

$$\begin{aligned}
\frac{d\mathbf{y}}{dt} &= -\gamma(1-\beta)\nabla f(\mathbf{y}) - \gamma\beta c(t+1)\mathbf{m}, \\
\frac{d\mathbf{m}}{dt} &= \frac{1}{1-c(t)}\left[\nabla f(\mathbf{y}) - c(t)\mathbf{m}\right].
\end{aligned}$$

However, at the Edge of Stability, the optimization trajectory deviates from this stable flow. We now derive a *central flow* to characterize the time-averaged behavior of SF-GD under this regime, particularly when a single top eigenvalue remains at the stability threshold. This derivation is not rigorous but follows the ansatz approach used by Cohen et al. (2025).

We model the trajectory as $\mathbf{y}_t = \bar{\mathbf{y}}_t + \rho_t\mathbf{u}_t$, where $\bar{\mathbf{y}}_t$ is the time-averaged iterate, $\mathbf{u}_t$ is the top Hessian eigenvector at $\bar{\mathbf{y}}_t$, and $\rho_t$ is the scalar displacement along $\mathbf{u}_t$. By construction, $\mathbb{E}[\rho_t] = 0$. Using a Taylor expansion of $\nabla f(\mathbf{y})$ around the reference point $\bar{\mathbf{y}}$, we obtain:

$$\nabla f(\mathbf{y}) = \nabla f(\bar{\mathbf{y}}) + \rho S(\bar{\mathbf{y}})\mathbf{u} + \tfrac{1}{2}\rho^2\nabla S(\bar{\mathbf{y}}) + \mathcal{O}(\rho^3),$$

where $S(\mathbf{y}) := \lambda_1(\nabla^2 f(\mathbf{y}))$ denotes the sharpness at $\mathbf{y}$. Taking expectations, the time-averaged gradient norm becomes:

$$\mathbb{E}[\nabla f(\mathbf{y}_t)] \approx \nabla f(\bar{\mathbf{y}}) + \mathbb{E}[\rho_t]S(\bar{\mathbf{y}})\mathbf{u} + \tfrac{1}{2}\mathbb{E}[\rho_t^2]\nabla S(\bar{\mathbf{y}}) = \nabla f(\bar{\mathbf{y}}) + \tfrac{1}{2}\mathbb{E}[\rho_t^2]\nabla S(\bar{\mathbf{y}}).$$

Based on these approximations, we can derive the following central flow dynamics of $\bar{\mathbf{y}}_t$:

$$\begin{aligned}
\frac{d\mathbf{y}}{dt} &= -\gamma(1-\beta)\left[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2(t)\nabla S(\mathbf{y})\right] - \gamma\beta c(t+1)\mathbf{m}, \\
\frac{d\mathbf{m}}{dt} &= \frac{1}{1-c(t)}\left[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2(t)\nabla S(\mathbf{y}) - c(t)\mathbf{m}\right],
\end{aligned}$$

where $\sigma^2(t)$ models $\mathbb{E}[\rho_t^2]$, the instantaneous variance of the oscillations around the central flow trajectory (i.e., the river trajectory).

Recall that at the Edge of Stability, the sharpness equilibrates near the stability threshold. We therefore assume that it remains constant along the central flow trajectory, satisfying

$$S(\mathbf{y}) = \frac{2}{(1-\beta)\gamma}, \quad \frac{d}{dt}\left(S(\mathbf{y})\right) = 0.$$

There exists a unique value of $\sigma^2(t)$ that ensures this condition holds, particularly in the regime where $t$ is large, where the coefficient $c(t) = 1/t$ becomes negligible. To compute this value of $\sigma^2$, we apply the chain rule and substitute the central flow dynamics:

$$\frac{dS(\mathbf{y})}{dt} = \left\langle \nabla S(\mathbf{y}), \frac{d\mathbf{y}}{dt} \right\rangle \approx \left\langle \nabla S(\mathbf{y}), -\gamma(1-\beta)\left[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2(t)\nabla S(\mathbf{y})\right] \right\rangle,$$

where we approximate $c(t) \approx 0$. Setting $\frac{dS(\mathbf{y})}{dt} = 0$ and rearranging gives:

$$\sigma^2(\mathbf{y}) \approx \frac{2\left\langle \nabla S(\mathbf{y}), -\nabla f(\mathbf{y}) \right\rangle}{\|\nabla S(\mathbf{y})\|^2}.$$

### G.3.2. Deriving the Central Flow of Schedule-Free Scalar Adam

We begin with the reformulated update rule for `SF-ScalarAdam`, as derived from ($\mathtt{SF}_{\mathbf{y}}$):

$$\begin{aligned}
\nu_t &= \beta_2 \nu_{t-1} + (1-\beta_2)\|\nabla f(\mathbf{y}_t)\|^2, \\
\mathbf{m}_t &= (1-c_t)\mathbf{m}_{t-1} + \frac{\nabla f(\mathbf{y}_t)}{\sqrt{\nu_t}}, \\
\mathbf{y}_{t+1} &= \mathbf{y}_t - \gamma\beta_1 c_{t+1}\mathbf{m}_t - \gamma(1-\beta_1)\frac{\nabla f(\mathbf{y}_t)}{\sqrt{\nu_t}}.
\end{aligned}$$

As in gradient descent, stable training dynamics are often well-approximated by their continuous-time analogs. We can therefore define a corresponding *stable flow* for `SF-ScalarAdam`:

$$\begin{aligned}
\frac{d\mathbf{y}}{dt} &= -\frac{\gamma(1-\beta_1)}{\sqrt{\nu}}\nabla f(\mathbf{y}) - \gamma\beta_1 c(t+1)\mathbf{m}, \\
\frac{d\mathbf{m}}{dt} &= \frac{1}{1-c(t)}\left[\frac{1}{\sqrt{\nu}}\nabla f(\mathbf{y}) - c(t)\mathbf{m}\right], \\
\frac{d\nu}{dt} &= \frac{1-\beta_2}{\beta_2}\left[\|\nabla f(\mathbf{y})\|^2 - \nu\right],
\end{aligned}$$

However, at the Edge of Stability, the optimization trajectory deviates from this stable flow. We now derive a *central flow* to characterize the time-averaged behavior of `SF-ScalarAdam` under this regime, particularly when a single top eigenvalue remains at the stability threshold. This derivation is not rigorous but follows the ansatz approach used by Cohen et al. (2025).

We model the trajectory as $\mathbf{y}_t = \bar{\mathbf{y}}_t + \rho_t\mathbf{u}_t$, where $\bar{\mathbf{y}}_t$ is the time-averaged iterate, $\mathbf{u}_t$ is the top Hessian eigenvector at $\bar{\mathbf{y}}_t$, and $\rho_t$ is the scalar displacement along $\mathbf{u}_t$. By construction, $\mathbb{E}[\rho_t] = 0$. Using a Taylor expansion of $\nabla f(\mathbf{y})$ around the reference point $\bar{\mathbf{y}}$, we obtain:

$$\nabla f(\mathbf{y}) = \nabla f(\bar{\mathbf{y}}) + \rho S(\bar{\mathbf{y}})\mathbf{u} + \tfrac{1}{2}\rho^2 \nabla S(\bar{\mathbf{y}}) + \mathcal{O}(\rho^3),$$

where $S(\mathbf{y}) := \lambda_1(\nabla^2 f(\mathbf{y}))$ denotes the sharpness at $\mathbf{y}$. Taking expectations, the time-averaged gradient and squared gradient norm become:

$$\begin{aligned}\mathbb{E}[\nabla f(\mathbf{y}_t)] &\approx \nabla f(\bar{\mathbf{y}}) + \mathbb{E}[\rho_t] S(\bar{\mathbf{y}})\mathbf{u} + \tfrac{1}{2}\mathbb{E}[\rho_t^2]\nabla S(\bar{\mathbf{y}}) = \nabla f(\bar{\mathbf{y}}) + \tfrac{1}{2}\mathbb{E}[\rho_t^2]\nabla S(\bar{\mathbf{y}}),\\ \mathbb{E}[\|\nabla f(\mathbf{y}_t)\|^2] &\approx \|\nabla f(\bar{\mathbf{y}})\|^2 + 2\mathbb{E}[\rho_t] S(\bar{\mathbf{y}})\left\langle \nabla f(\bar{\mathbf{y}}), \mathbf{u}\right\rangle + \mathbb{E}[\rho_t^2] S(\bar{\mathbf{y}})^2 = \|\nabla f(\bar{\mathbf{y}})\|^2 + \mathbb{E}[\rho_t^2] S(\bar{\mathbf{y}})^2.\end{aligned}$$

Based on these approximations, we can derive the following central flow dynamics of $\bar{\mathbf{y}}_t$:

$$\begin{aligned}\frac{d\mathbf{y}}{dt} &= -\frac{\gamma(1-\beta_1)}{\sqrt{\nu}}\left[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2(t)\nabla S(\mathbf{y})\right] - \gamma\beta_1 c(t+1)\mathbf{m},\\ \frac{d\mathbf{m}}{dt} &= \frac{1}{1-c(t)}\left[\frac{1}{\sqrt{\nu}}(\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2(t)\nabla S(\mathbf{y})) - c(t)\mathbf{m}\right],\\ \frac{d\nu}{dt} &= \frac{1-\beta_2}{\beta_2}\left[\|\nabla f(\mathbf{y})\|^2 + \sigma^2(t) S(\mathbf{y})^2 - \nu\right],\end{aligned}$$

where $\sigma^2(t)$ models $\mathbb{E}[\rho_t^2]$, the instantaneous variance of the oscillations around the central flow trajectory (i.e., the river trajectory).

Recall that at the Edge of Stability, the preconditioned sharpness equilibrates near the stability threshold. We therefore assume that it remains constant along the central flow trajectory, satisfying

$$\frac{S(\mathbf{y})}{\sqrt{\nu}} = \frac{2}{(1-\beta_1)\gamma}, \quad \frac{d}{dt}\left(\frac{S(\mathbf{y})}{\sqrt{\nu}}\right) = 0.$$

There exists a unique value of $\sigma^2(t)$ that ensures this condition holds, particularly in the regime where $t$ is large, where the coefficient $c(t) = 1/t$ becomes negligible. To compute this value of $\sigma^2$, we apply the chain rule and substitute the central flow dynamics:

$$\begin{aligned}\frac{d}{dt}\left(\frac{S(\mathbf{y})}{\sqrt{\nu}}\right) =& \frac{1}{\sqrt{\nu}}\left\langle \nabla S(\mathbf{y}), \frac{d\mathbf{y}}{dt}\right\rangle - \frac{S(\mathbf{y})}{2\nu^{3/2}}\cdot\frac{d\nu}{dt}\\ \approx& \frac{1}{\sqrt{\nu}}\left\langle \nabla S(\mathbf{y}), -\frac{\gamma(1-\beta_1)}{\sqrt{\nu}}\left[\nabla f(\mathbf{y}) + \tfrac{1}{2}\sigma^2\nabla S(\mathbf{y})\right]\right\rangle\\ &- \frac{S(\mathbf{y})}{2\nu^{3/2}}\cdot\frac{1-\beta_2}{\beta_2}\left[\|\nabla f(\mathbf{y})\|^2 + \sigma^2 S(\mathbf{y})^2 - \nu\right],\end{aligned}$$

where we approximate $c(t) \approx 0$. Setting $\frac{d}{dt}\left(\frac{S(\mathbf{y})}{\sqrt{\nu}}\right) = 0$ and rearranging gives:

$$\sigma^2 \approx \frac{\langle \nabla S(\mathbf{y}), -\nabla f(\mathbf{y})\rangle + \frac{1-\beta_2}{2(1-\beta_1)\beta_2\gamma}\left[S(\mathbf{y})\sqrt{\nu} - \frac{1}{\sqrt{\nu}}S(\mathbf{y})\|\nabla f(\mathbf{y})\|^2\right]}{\frac{1}{2}\|\nabla S(\mathbf{y})\|^2 + \frac{1-\beta_2}{2(1-\beta_1)\beta_2\gamma}\cdot\frac{S(\mathbf{y})^3}{\sqrt{\nu}}}.$$

Using the condition $\frac{S(\mathbf{y})}{\sqrt{\nu}} = \frac{2}{(1-\beta_1)\gamma}$, we substitute $\sqrt{\nu} = \frac{1}{2}(1-\beta_1)\gamma S(\mathbf{y})$ into the expression and obtain:

$$\sigma^2(\mathbf{y}; \beta_1, \beta_2, \gamma) \approx \frac{\langle \nabla S(\mathbf{y}), -\nabla f(\mathbf{y})\rangle + \frac{1-\beta_2}{\beta_2}\left[\frac{1}{4}S(\mathbf{y})^2 - \frac{1}{(1-\beta_1)^2\gamma^2}\|\nabla f(\mathbf{y})\|^2\right]}{\frac{1}{2}\|\nabla S(\mathbf{y})\|^2 + \frac{1-\beta_2}{(1-\beta_1)^2\beta_2\gamma^2}S(\mathbf{y})^2}.$$

Notably, $\sigma^2$ depends only on the current iterate $\mathbf{y}$ and the hyperparameters $\beta_1$, $\beta_2$, and $\gamma$. Moreover, unlike `SF-GD`, $\sigma^2$ *does* depend on momentum parameters.

### G.4. Omitted Calculations in Appendix C

We derive the closed-form approximation of the averaging weights $\alpha_t$ under the modified SF method, where the update coefficient is set to

$$c_t = \frac{(1-\beta)C}{t}.$$

Under this setting, we show that the induced averaging weights satisfy the approximation

$$\alpha_t \approx \frac{C}{T}\left(\frac{t}{T}\right)^{C-1}.$$

Recall that for general $\{c_t\}$, the iterates $\mathbf{x}_T$ can be written as a weighted average of past $\mathbf{y}_t$:

$$\mathbf{x}_T = \sum_{t=1}^{T} \alpha_t \, \mathbf{y}_t, \quad \alpha_t := \frac{c_t}{(1-c_t)(1-\beta)+c_t} \prod_{s=t+1}^{T} \left[\frac{(1-c_s)(1-\beta)}{(1-c_s)(1-\beta)+c_s}\right].$$

Now, substitute $c_t = \frac{(1-\beta)C}{t}$. For large $t$, we approximate:

$$\begin{aligned}
\alpha_t &= \frac{\frac{C}{t}}{1-\frac{(1-\beta)C}{t}+\frac{C}{t}} \left[\prod_{s=t+1}^{T} \frac{1-\frac{(1-\beta)C}{s}}{1+\frac{\beta C}{s}}\right] \\
&\approx \frac{C}{t}\left[\prod_{s=t+1}^{T} \frac{1-\frac{(1-\beta)C}{s}}{1+\frac{\beta C}{s}}\right] \\
&\approx \frac{C}{t}\left[\prod_{s=t+1}^{T} \frac{\exp\left(-\frac{(1-\beta)C}{s}\right)}{\exp\left(\frac{\beta C}{s}\right)}\right] \\
&= \frac{C}{t}\left[\prod_{s=t+1}^{T} \exp\left(-\frac{C}{s}\right)\right] \\
&= \frac{C}{t}\exp\left(-\sum_{s=t+1}^{T}\frac{C}{s}\right).
\end{aligned}$$

Using the integral approximation for the harmonic sum:

$$\sum_{s=t+1}^{T}\frac{1}{s} \approx \int_{s=t}^{T}\frac{1}{s},$$

we obtain

$$\begin{aligned}
\alpha_t &\approx \frac{C}{t}\exp\left(-\sum_{s=t+1}^{T}\frac{C}{s}\right) \\
&\approx \frac{C}{t}\exp\left(-\int_{s=t}^{T}\frac{C}{s}\right) \\
&= \frac{C}{t}\exp\left(-C\log T + C\log t\right) \\
&= \frac{Ct^{C-1}}{T^{C}}.
\end{aligned}$$

Thus, we conclude that

$$\alpha_t \;\approx\; \frac{C}{T}\Big(\frac{t}{T}\Big)^{C-1}.$$

Figure 13 and Figure 14 show that the averaging weights $\{\alpha_t\}$ in our refined SF method closely follow the approximation $\alpha_t \approx \frac{C}{T}(\frac{t}{T})^{C-1}$, across different values of $\beta$, $C$, and $T$.

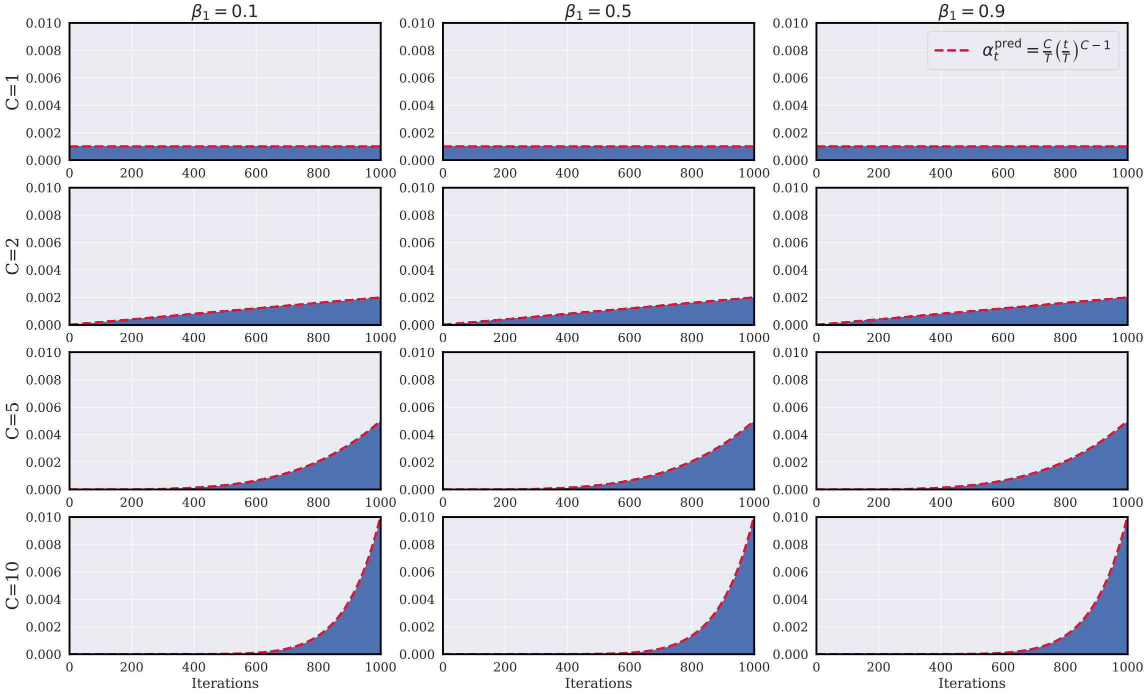

Figure 13: **Averaging weights in the refined SF method.** Histogram of $\{\alpha_t\}_{t=1}^T$ over $T = 1000$ iterations for varying values of $\beta$ and $C$.

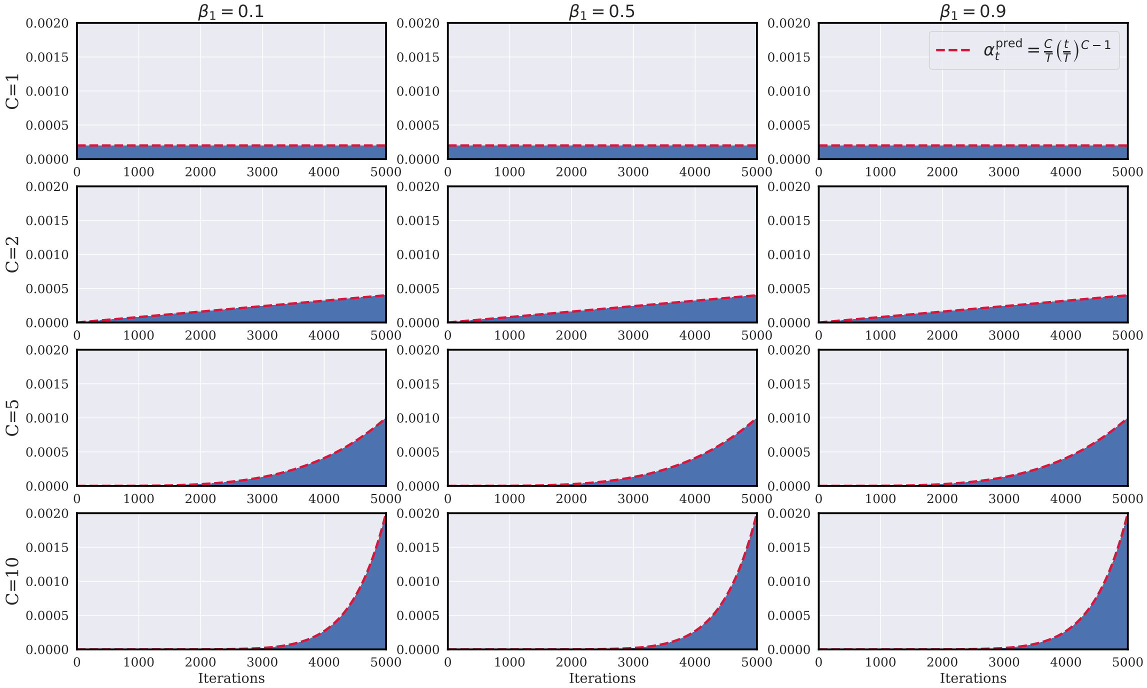

Figure 14: **Averaging weights in the refined SF method.** Histogram of $\{\alpha_t\}_{t=1}^{T}$ over $T = 5000$ iterations for varying values of $\beta$ and $C$.