Training for the Model You Return: Improving Optimization for Iterate-Averaged Language Models

Kwok Chun Au
Department of Computer Science
Columbia University
ka3094@columbia.edu

Adam Block
Departments of Computer Science
and Electrical Engineering
Columbia University
adam.block@columbia.edu

Abstract

Many modern Language Model (LM) pipelines return an averaged model, such as an exponential moving average of the training iterates, rather than the final iterate itself. This raises a fundamental question: given that we will return an iterate average, how should we change training to improve the performance of this average? We study this question by formulating optimizer design for the iterate-average estimator as an optimal-control problem. In a continuous-time stochastic quadratic model, we solve for the control strategy that minimizes the error of the returned average subject to a penalty on the size of the intervention. A practical approximation to this controller yields PACE, a lightweight wrapper around AdamW that pulls the live weights toward their exponential moving average with a clipped, per-coordinate control strength. We prove that a stylized version of PACE converges at the standard stochastic convex optimization rate, up to a factor depending on the averaging rule, while in the quadratic setting it can strictly improve the limiting squared error of the iterate-average estimator and can do so by an arbitrarily large factor on some instances. Empirically, our results suggest that PACE improves over AdamW and EMA-evaluated AdamW in supervised fine-tuning of 1-2B parameter LMs and in GPT-2 pretraining on FineWeb for a wide range of learning rates, decay schedules, and other hyperparameters. [1]

1 Introduction

Inspired by the classical theory of stochastic convex optimization [31, 33], many Language Model (LM) pipelines use some form of model averaging, such as an exponential moving average (EMA) of training iterates, to stabilize optimization and improve the performance of the returned model [1, 3, 5, 7, 15, 17, 21, 28, 39]. A vast body of work has been devoted to understanding LM training, but the ubiquity of iterate averaging in practice raises a fundamental question: *given that an averaged LM will be returned, how should we change training to improve the performance of this average?*

While prior work has implicitly addressed this question by observing that iterate averaging can allow for more aggressive training through higher learning rates or more limited learning rate decay [3, 5, 9], in this work, we take a direct approach by asking whether we can design a simple modification to the base optimization algorithm that explicitly controls the training trajectory to improve the performance of the iterate-average estimator. Our starting point is to follow Block and Zhang [3] and observe that once we commit to returning an average of the training trajectory, the optimization algorithm is no longer just producing a final iterate, but rather a statistic of the entire trajectory: we may thus rephrase our key question by asking how we can best modify the training trajectory so as to make the returned model average maximally close to the optimum.

[1] All code is available at https://github.com/leonou2010/pace-optimizer.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

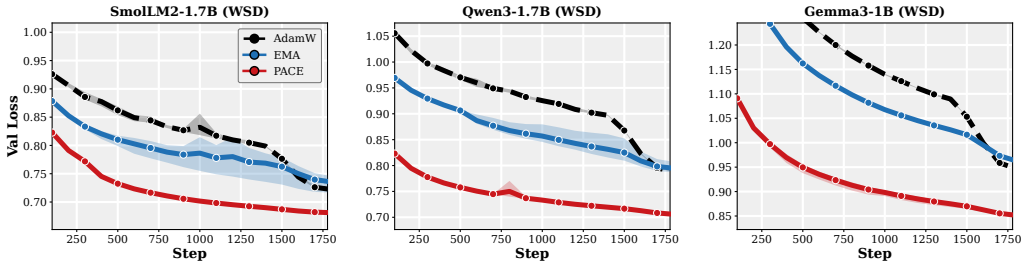

Figure 1: **Performance of PACE on fine-tuning.** Validation cross-entropy on `smol-smoltalk` for SmolLM2-1.7B (**left**), Qwen3-1.7B (**middle**), and Gemma3-1B (**right**). All three methods use WSD; curves show means over three seeds with min–max bands. PACE strictly improves on both baselines on all three models; see Figure 19 for the constant-learning-rate companion. A single-seed comparison with PACE at a constant learning rate and the baselines at their best decay schedules appears in Figure 16.

To answer this question, we take a principled approach to algorithm design by deriving a new optimization algorithm from an optimal-control formulation of a stochastic quadratic optimization problem. Indeed, while LM training is obviously not quadratic, many recent works have derived practically effective optimization interventions by designing algorithms in analytically tractable settings such as convex, quadratic, or linear problems [3, 10, 13, 37, 44]. In particular, our derivation begins in a deliberately simplified setting: continuous-time stochastic gradient descent on a quadratic objective with additive noise, where the 'modification' is realized as an additive *control*. In this model, the uncontrolled dynamics reduce to the familiar Ornstein–Uhlenbeck approximation of noisy gradient descent near a quadratic minimum [3, 19, 26]. We then add a control input to the dynamics and choose this input to minimize the squared error of the returned iterate average, subject to a quadratic penalty on the size of the intervention. This gives a linear-quadratic stochastic control problem [12, 20, 42]. Solving this problem yields an optimal controller with a natural interpretation: the algorithm should pull the live iterate toward an estimate of the optimum and toward consistency with the accumulated average of the trajectory.

The exact optimal controller is not directly implementable in modern LM training because it depends on quantities unknown to the learner. We therefore turn the controller into a practical algorithm through a sequence of heuristic approximations. We call the resulting method PACE (Algorithm 1), a lightweight wrapper around AdamW [24] that pays an additional memory cost of model weights in order to better stabilize the training process. Because PACE is derived in such a simplified setting, it is not a priori obvious whether the resulting algorithm has any concrete theoretical guarantees. We show that a stylized version of the algorithm, evaluated at the flat average of its iterates, has convergence guarantees for general convex losses with unbiased stochastic gradients:

Theorem 1 (Informal version of Theorem 3)**.** *For any convex loss with unbiased stochastic gradients, the iterate-average estimator returned by PACE converges to the minimum at the standard stochastic convex optimization rate, up to a factor depending on the strength of the EMA.*

In essence, Theorem 1 shows that our proposed method is never much worse than SGD for convex objectives, even when the quadratic assumption used in the derivation is dropped. We then show that, in the quadratic setting from which the algorithm was derived, the controlled dynamics can strictly improve the limiting squared error of the iterate-average estimator relative to the uncontrolled dynamics, and that this improvement can be arbitrarily large on certain quadratic instances:

Proposition 1 (Informal version of Propositions 2 and 3)**.** *In the quadratic setting from which PACE is derived, the optimal control strategy can strictly improve the limiting squared error of the iterate-average estimator relative to the uncontrolled dynamics, and this improvement can be arbitrarily large on certain quadratic instances.*

We complement our theory with experiments on LM fine-tuning and pretraining, results of the former of which can be seen in Figure 1. In particular, across three different models of size between 1B and 2B parameters, we find that PACE strictly improves over both AdamW and an EMA thereof. In Section 4, we present a more detailed analysis of the performance of PACE across a range of

hyperparameters as well as explore the efficacy of PACE on a small pretraining setup at the Chinchilla-optimal token budget [14]. We find that PACE is competitive with the Schedule-Free algorithm of Defazio et al. [9], which also uses iterate averaging to stabilize training and remove the need for learning rate decay and that PACE remains competitive with AdamW even when the latter is allowed to decay the learning rate to zero, suggesting that PACE can stabilize training in a regime that is not subject to learning rate decay. We now briefly summarize some related works, before moving on to the formal problem setup, algorithm derivation, and rigorous guarantees in Section 2.

Iterate averaging in deep learning. Originally introduced from the theory of stochastic convex optimization [31, 33], iterate averaging has been shown to be a powerful tool for stabilizing optimization and improving generalization in deep learning [3, 5, 7, 15, 17]. Many approaches have been proposed, including stochastic weight averaging [15], exponential moving averages, and even more complex schemes like model soups [39] and virtually all modern open-source LMs used some form of iterate averaging [1, 21, 28]. As in our work, there has been much literature on using continuous-time analysis to understand the behavior of iterate averaging in deep learning [5, 7, 25, 26], with Block and Zhang [3] being the most directly relevant to the present paper. Unlike that work, which uses continuous-time analysis in a noisy quadratic model to propose a new averaging scheme, here we commit to the standard exponential moving average and instead use continuous-time analysis to design a new optimization algorithm that explicitly controls the training trajectory to improve the performance of this standard averaging scheme.

Comparison to similar optimizers. Several authors have proposed analyzing optimization through the lens of control theory [8, 22, 40], although the former primarily investigates convergence bounds of existing optimizers and the latter two use the Kalman filter to derive an optimization intervention that, while interesting, does not appear to scale to LMs of modern size. With regard to the final algorithm, several prior optimizers have similar update rules. Pagliardini et al. [29], Zhang et al. [45] also consider averaging during training (instead of in an offline, post-hoc manner). The former applies averaging to gradients as opposed to iterates and can be thought of as a more sophisticated version of momentum, whereas the latter has a very different update rule from PACE and is intended to solve communication bottlenecks in distributed training. The authors of Zhang et al. [44] propose a method that takes gradient steps and periodically transports the live weights toward a slow-moving EMA; our approach recovers a similar update when the pullback value is large, although empirically we find that PACE improves performance with intermediate pullback values, improving on that prior work. Of greatest relevance is the schedule-free (SF) update proposed by Defazio et al. [9], which also uses iterate averaging to stabilize training and remove the need for learning rate decay, although that work proposes a different algorithm. Our method is competitive with SF empirically, although pays for improved theoretical convergence in the quadratic setting with a slower theoretical guarantee in the general convex setting. Moreover, our naïve implementation of PACE uses an additional memory cost of model weights, as we incorporate momentum.

Notation. We use lowercase letters to denote scalars and uppercase bold letters to denote matrices. We always denote the standard Brownian motion in $\mathbb{R}^d$ by W_t and $\|\cdot\|$ refers to the euclidean norm. We use $\mathrm{diag}(\alpha_1, \ldots, \alpha_d)$ to denote the diagonal matrix with entries $\alpha_1, \ldots, \alpha_d$ on the diagonal. For vector functions depending on time, we use a subscript to denote the time dependence (e.g. θ_t), unless we are concerned with a single coordinate, in which case we use parentheses (e.g. $\theta_i(t)$).

2 Algorithm Derivation and Guarantees

2.1 Formal Problem Setup

Formally, we are interested in *stochastic optimization*, where for some loss function $F : \mathbb{R}^d \to \mathbb{R}$ with minimum $\mu^\star = \mathrm{argmin}_\theta F(\theta)$, we wish to return some $\widehat{\theta}$ that is close to $\mu^\star$. As is standard in the stochastic optimization literature [5, 9, 24, 31, 33], we will assume only that we have noisy gradient access to F, i.e., for any $\theta \in \mathbb{R}^d$, we can sample some g such that $\mathbb{E}[g] = \nabla F(\theta)$.

Our starting point is to recall the following empirical observation about the behavior of language model training: iterate averaging schemes like EMA and BEMA are extraordinarily effective at improving the performance of LMs, both through stabilizing the optimization and through improving generalization [15, 17, 28]. While there has been a large body of work dedicated to understanding

and improving such iterate averaging schemes, we instead focus on the question of how to improve training algorithms under the assumption that we will be returning an iterate average. In particular, we ask the following fundamental question:

Given that we will return an iterate average after training, how should we modify optimization algorithms to minimize the loss of the returned weights?

We will answer this question theoretically in a simplified setting, and empirically by training large language models with modern optimization algorithms. In theory, following a long line of work in optimization for deep learning, we will derive an answer to this question by considering a quadratic model [3, 5, 7, 10, 13, 44], and then prove convergence guarantees for our algorithm in a more general convex setting. In particular, for our derivation, we will make the following four assumptions. First, the loss function F is a quadratic function of the form $F(\theta) = {}^1\!/_2 \left(\theta - \mu^\star\right)^\top \mathbf{A} \left(\theta - \mu^\star\right)$ for a *diagonal* positive definite matrix $\mathbf{A}$ (for the general, non-diagonal case, see Section E). Second, in order to further simplify the analysis, we will follow prior work [3–5, 25, 26] and consider a continuous time limit of the optimization process, where the dynamics of the parameters θ_t are given by a stochastic differential equation (SDE). Third, we will assume that we are returning a simple time average of the iterates, as opposed to more sophisticated schemes like EMA and BEMA. Fourth, we will consider a Bayesian setting, where $\mu^\star$ is a random variable with a known prior distribution. Finally, we will focus on the case of vanilla SGD with stationary, homoscedastic noise, i.e. the noise covariance $\mathbf{\Sigma}$ is constant and does not depend on θ_t. We will model the modification to the base optimization algorithm as a *control* input u_t that can be added to the dynamics of θ_t, affecting the trajectory of the optimization process. Combining these assumptions, we thus return $\widehat{\theta}_{\mathsf{EMA},T}$:

$$\widehat{\theta}_{\mathsf{EMA},T} = \frac{1}{T}\int_0^T \theta_t dt \quad \text{where} \quad d\theta_t = \left[\mathbf{A}\left(\mu^\star - \theta_t\right) + u_t\right] dt + \mathbf{\Sigma}^{1/2} dW_t, \tag{1}$$

and W_t a standard Brownian motion in $\mathbb{R}^d$ [19, 42]. Critically, we wish $\widehat{\theta}_{\mathsf{EMA},T}$ to be a good estimator of $\mu^\star$ while exerting as little control effort as possible; one way to formalize this desideratum is to seek a control strategy u_t that minimizes the following objective:

$$J_T(u) = \mathbb{E}\left[\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\|^2 + \lambda \cdot \int_0^T \|u_t\|^2 \, dt\right], \tag{2}$$

where $\lambda > 0$ is a regularization parameter that controls the tradeoff between the quality of the returned estimator $\widehat{\theta}_{\mathsf{EMA},T}$ and how far from the base optimization algorithm the control strategy u_t is allowed to deviate. We will refer to the problem of minimizing $J_T(u)$ over all progressively measurable control strategies u_t as the *optimal control problem* for the iterate-average estimator. Note that in the case that $u_t = 0$ for all t, (1) reduces to the well-known Ornstein-Uhlenbeck process [19, 26], and we recover the setting considered in Block and Zhang [3].

2.2 Derivation of the Optimal Control Strategy

We now present the optimal control solution for the iterate-average estimator.

Theorem 2. *Suppose that $\mu^\star$ is drawn from a known prior distribution and θ_t evolves according to (1) for some control strategy u_t[2] and that $\mathbf{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_d)$ is a diagonal positive definite matrix. Let $\widehat{\mu}_t = \mathbb{E}\left[\mu^\star | \theta_s, s \le t\right]$. Then the optimal control strategy u_t that minimizes the objective $J_T(u)$ defined in (2) is given for each coordinate $i \in \{1, \ldots, d\}$ by*

$$\begin{aligned} u_i^\star(t) = &\frac{\left(1 - e^{-\alpha_i(T-t)}\right)^2}{\alpha_i^2 \lambda T^2 + T - t - {}^2\!/_{\alpha_i}\left(1 - e^{-\alpha_i(T-t)}\right) + \frac{1 - e^{-2\alpha_i(T-t)}}{2\alpha_i}} \left(\widehat{\mu}_i(t) - \theta_i(t)\right) \\ &+ \frac{\alpha_i\left(1 - e^{-\alpha_i(T-t)}\right)}{\alpha_i^2 \lambda T^2 + T - t - {}^2\!/_{\alpha_i}\left(1 - e^{-\alpha_i(T-t)}\right) + \frac{1 - e^{-2\alpha_i(T-t)}}{2\alpha_i}} \left(t \cdot \widehat{\mu}_i(t) - \int_0^t \theta_i(s) ds\right). \end{aligned} \tag{3}$$

We defer a complete proof of this result to Section E, as well as generalizing beyond the diagonal $\mathbf{A}$ setting. After some algebra and augmenting the state, the analysis rests on classical results from

[2] Formally, we assume u_t is *progressively measurable*, i.e. can only include information from observations up to time t.

Algorithm 1 PACE: **P**ullback **A**veraging **C**ontrol for **E**fficient Optimization

Require: Learning rate η, pullback strength c, EMA power κ, scale ε, update frequency uf.

$\widehat{\theta}_{\mathsf{EMA},0} \leftarrow \theta_0$
for $t = 1, 2, \ldots, T$ **do**
  $\theta_t \leftarrow \mathrm{AdamW.step}(\theta_{t-1};\, \eta)$ and $\hat{v}_t \leftarrow \mathrm{AdamW.preconditioner}(\theta'_t)$.
  if $t \mod \mathrm{uf} = 0$ **then**
    $\lambda_{t,i} \leftarrow \min\!\big(\eta\, c\, (1+t)^{-\kappa} / (\sqrt{\hat{v}_{t,i}} + \varepsilon),\ 1\big)$ ▷ Clipped per-coordinate pullback gain
    $\theta_t \leftarrow \theta_t + \lambda_t \odot (\theta_{t-1}^{\mathrm{EMA}} - \theta_{t-1})$
    $\widehat{\theta}_{\mathsf{EMA},t} \leftarrow (1-\beta_t)\, \widehat{\theta}_{\mathsf{EMA},t-1} + \beta_t\, \theta_t, \quad \beta_t = (1+t)^{-\kappa}$ ▷ EMA update
  end if
end for
return $\widehat{\theta}_{\mathsf{EMA},T}$

the theory of linear quadratic stochastic control [12, 20] including solving the Riccati equation and applying the principle of certainty equivalence. While (3) is attractive in that it is a closed-form expression for the optimal control strategy, it is not easily implementable in practice, as it depends on $\widehat{\mu}_t$, the posterior mean of $\mu^\star$ given the trajectory of θ_s up to time t. Even in the case of a Gaussian prior, which would imply that $\widehat{\mu}_t$ is given by a Kalman filter [12, 19, 42], the resulting control strategy would be difficult to implement in practice, especially at the scale of language models. Thus, we simplify (3) to obtain a more practical control strategy by making the following approximations: (i) we replace $\widehat{\mu}_t \approx \widehat{\theta}_{\mathsf{EMA},t}$ and (ii) we suppose that $t \ll T$. The first assumption is connected with the fundamental motivation of our work: we wish to return $\widehat{\theta}_{\mathsf{EMA},T}$ as an estimate of $\mu^\star$ at the end of training, so it is natural to use $\widehat{\theta}_{\mathsf{EMA},t}$ as an estimate of $\mu^\star$ at time t. This substitution is a practical approximation; the posterior mean also contains boundary and cumulative-control corrections (Section E.4). The second assumption is motivated primarily by a desire for simplicity and the observation that the optimal control strategy is most impactful in the early stages of training, when the optimization trajectory is far from the optimum and thus there is more room for improvement. Assuming $T \gg 1$, we approximate $T - t \approx T$ and $\exp(-\alpha_i(T-t)) \approx 0$. With these approximations, we obtain the following practical control:

$$u_t = T^{-1} \left(\mathbf{I} + \lambda \mathbf{A}^2 T\right)^{-1} \left(\widehat{\theta}_{\mathsf{EMA},t} - \theta_t\right). \tag{4}$$

While (4) is a valid approximation to the optimal control strategy, it is not yet ready for implementation, as it remains in continuous time. Furthermore, the modification to the base optimization algorithm is a function of $\mathbf{A}$, which is unknown in practice. Moreover, practical iterate averaging schemes in deep learning tend to use exponential instead of uniform weighting. We thus discretize (4) and replace $\mathbf{A}$ with a diagonal matrix $\mathbf{C}$ to arrive at the following update strategy: given current parameters θ_k at iteration k, stochastic gradient g_k, and learning rate η, parameters are updated according to

$$\theta_{k+1} = \theta_k - \eta \cdot g_k + \mathbf{C}\left(\widehat{\theta}_{\mathsf{EMA},k} - \theta_k\right), \quad \widehat{\theta}_{\mathsf{EMA},k+1} = (1-\beta) \cdot \widehat{\theta}_{\mathsf{EMA},k} + \beta \cdot \theta_{k+1}, \tag{5}$$

where β is the averaging parameter. Thus we are left with (5), which is a simple modification to the base optimization algorithm that can be implemented with minimal computational overhead. Indeed, $\mathbf{C}$ is a diagonal matrix and thus the matmul can be accomplished coordinate wise, at roughly the same cost as the preconditioning step in Adam or AdamW [18, 24]. While there still exist several additional tweaks to (5) required to make it work well in practice (Section 3), we have now derived a simple, practical modification to the base optimization algorithm that is motivated by the optimal control solution to the iterate-average estimator in a simplified setting.

2.3 Convergence Guarantee in the Convex Setting

We derived (5) above as a practical approximation to the optimal control strategy for the iterate-average estimator in a simplified setting through heuristics: while we provided a rigorous derivation of the optimal control strategy in the quadratic setting, we then made use of several approximations and discretization in order to arrive at the update rule. In this section we provide formal guarantees for this update. We first show that for arbitrary convex loss functions, the uniform average of the

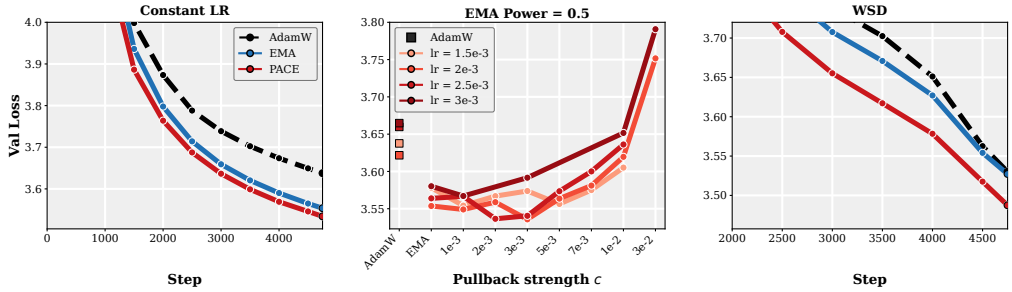

Figure 2: **Performance of PACE on pretraining** of GPT-2 (124M) on FineWeb at the Chinchilla-optimal token budget. **Left:** validation cross-entropy trajectories at a constant learning rate, with EMA and PACE at the same EMA power ($\kappa = 0.5$) so that they differ only in the pullback. **Middle:** effect of pullback strength c on validation cross-entropy at $\kappa = 0.5$. **Right:** the analogous comparison under WSD, using linear learning rate decay to zero over the last 20% of steps. PACE outperforms both the AdamW and EMA baselines under each schedule.

iterates generated by (5) converges at the same rate as SGD, up to a constant factor depending on $\mathbf{C}$ and β.

Theorem 3. *Suppose that $F : \mathbb{R}^d \to \mathbb{R}$ is a convex, G-Lipschitz function and suppose for each k we have access to a stochastic gradient g_k such that $\mathbb{E}[g_k] = \nabla F(\theta_k)$ and $\mathbb{E}[\|g_k - \nabla F(\theta_k)\|^2] \le \sigma^2$. Suppose that θ_k evolves according to (5) for some diagonal matrix $\mathbf{C}$ with entries in $[0, 1]$ and some $\beta \in (0, 1)$. Let $\overline{\theta}_T = T^{-1} \sum_{k=0}^{T-1} \theta_k$ be the flat average. Then it holds that for optimally tuned learning rate η, $\mathbb{E}\left[F(\overline{\theta}_T) - F(\mu^\star)\right] \le \sqrt{2-\beta}/\beta \cdot \|\theta_1 - \mu^\star\| \cdot \sqrt{(G^2+\sigma^2)/T}$.*

The proof of Theorem 3 is given in Section F and follows by a standard analysis of SGD coupled with a recursive analysis to ensure that the update does not drift too far away from SGD itself. In particular, Theorem 3 ensures that the flat average of the iterates generated by (5) enjoys the same $O(1/\sqrt{T})$ and is worse only by a multiplicative factor depending on the strength of the EMA update. Unfortunately, this issue is intrinsic to the analysis and likely cannot be improved.

While the previous result shows that (5) is never that much worse than SGD for convex functions, in the special case of quadratics, it can be quite a lot better. As stated above, locally approximating the loss function of a LM by a quadratic is a common heuristic in optimization for deep learning [3, 5, 43, 44], where the Hessian of the loss is used as a proxy for local curvature. Indeed, recent evidence shows that local quadratic models can predict optimization dynamics over windows lasting up to 10% of training in language models [27]. In this setting, we show that (5) enjoys improved convergence compared to SGD.

Proposition 2. *Suppose that $F(\theta) = 1/2(\theta - \mu^\star)^\top \mathbf{A}(\theta - \mu^\star)$ for some diagonal positive definite matrix $\mathbf{A}$ and that θ_k evolves according to (5) for some diagonal matrix $\mathbf{C}$ with entries in $[0, 1]$ and some $\beta \in (0, 1)$. Suppose further that the stochastic gradients are given by $g_k = \mathbf{A}(\theta_k - \mu^\star) + \xi_k$, where ξ_k is a zero-mean noise term with covariance $\sigma^2\mathbf{I}$. Then for any learning rate η such that $\|\eta\mathbf{A}\|_{\text{op}} < 1$ and choice of β, there is some $\mathbf{C}$ such that $\lim_{T\to\infty} \mathbb{E}\left[\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\|^2\right]$ is strictly smaller than the limiting squared error of the iterate-average estimator returned by SGD.*

The proof of Proposition 2 is given in Section F and demonstrates that (5) can rigorously improve the convergence of the iterate-average estimator in the quadratic setting. Finally, we show that the improvement over SGD can be arbitrarily large for certain choices of $\mathbf{C}$ and β for some problems.

Proposition 3. *For any $\varepsilon > 0$, there is some quadratic problem and some choice of $\mathbf{C}$ and β such that the limiting squared error of the iterate-average estimator returned by (5) is at least a factor of ε smaller than the limiting squared error of the iterate-average estimator returned by SGD.*

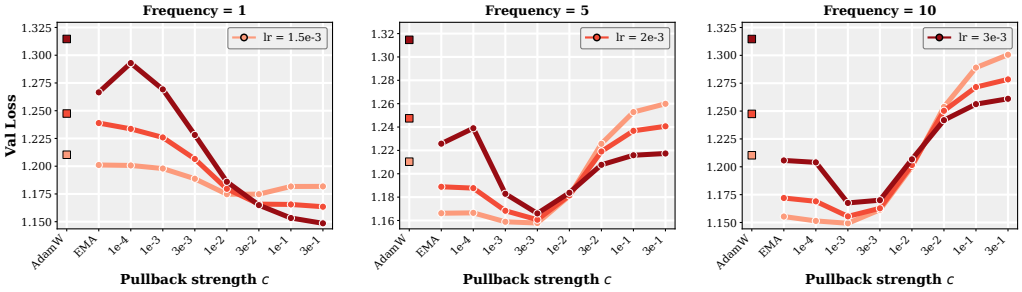

Figure 3: **Effect of update frequency and pullback strength.** Validation cross-entropy of SmolLM2-135M for different learning rates, pullback strengths, and update frequencies with κ=0.3. Optimal pullback strength remains relatively robust to learning rate and update frequency, with improvement over the EMA baseline across a wide range of settings.

3 Introducing PACE: Pullback Averaging Control for Efficient Optimization

While we derived a simple modification to the base optimization algorithm in (5), several additional tweaks make the proposed method work in practice. The first key modification is that we will replace the SGD update with an AdamW update [24], which is a standard optimizer choice for language models; in particular, we will use both momentum and preconditioning in the vanilla optimizer step. The second question is how to choose the matrix $\mathbf{C}$ that preconditions the pullback toward the iterate average. According to the theory in (4), the optimal choice of $\mathbf{C}$ scales inversely with $\mathbf{A}$ and T. While in practice language models are obviously not quadratic, locally they can be approximated thereby, with $\mathbf{A}$ corresponding to the Hessian [43, 44]; following the standard noisy-quadratic-model identification of Adam's second-moment estimate v_t as a Hessian-diagonal proxy, we will thus set $\mathbf{C}$ to be a diagonal matrix with entries given by the Adam preconditioner [18]. One point where our empirical implementation deviates from the theory is that the latter suggests preconditioning should be scale like $1/v_t$, but for stability reasons we find that the standard Adam preconditioner of $1/\sqrt{v_t}+\varepsilon$ works better in practice.

Following standard praxis in deep learning, in order to account for the nonstationarity of the optimization landscape, we replace a fixed β in the EMA with a decaying $\beta_t = (1+t)^{-\kappa}$, where $\kappa \in (0,1)$ is a hyperparameter [3, 5, 7, 17]; in effect this makes the iterate average more responsive to recent iterates, with $\kappa = 1$ recovering uniform averaging. For the same reason, we replace the T^{-1} scaling in (4) with a slower decaying factor that allows the pullback to remain significant even late in training. We incorporate clipping to ensure that our updates are always convex combinations of the iterate averaged point and the naïve optimizer update step. The resulting control becomes

$$u_t = \min\left((1+t)^{-\kappa} \cdot \eta c / (\sqrt{v_t}+\varepsilon), 1\right)\left(\widehat{\theta}_{\mathsf{EMA},t} - \theta_t\right), \tag{6}$$

where the operations are applied coordinate-wise, η is the learning rate, $c \geq 0$ is a single scalar setting the controller strength, and v_t is the Adam second-moment estimate. Note that in the limit as $c \uparrow \infty$, we recover the Lookahead optimizer with an inner loop of a single step [44], while at $c = 0$, we recover AdamW with EMA. Finally, we allow for the update to only occur at a specified frequency, uf, as is common in practical implementations of EMA. The result, PACE, is detailed in Algorithm 1.

Implementation Concerns. In practice, there are two concerns with implementing PACE at scale. First, we have introduced a new tuning parameter c that controls the strength of the pullback toward the iterate average, as well as the update frequency uf; on the other hand, the update frequency is already a parameter in standard implementations of EMA, and our experiments suggest that the optimal pullback strength is relatively robust and transfers across hyperparameters like learning rate (cf. Figure 3). Second, and perhaps more importantly, PACE requires maintaining an additional copy of model weights during training, which is a nontrivial memory overhead. One mitigating factor is that this additional pressure is less relevant at large scale due to the use of large batch sizes, but exploring further mitigations of the memory overhead of PACE is an important direction for future work.

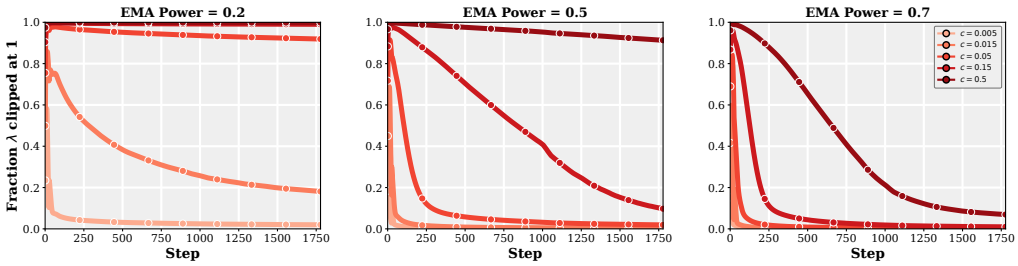

Figure 4: **Fraction of clipped updates** throughout fine-tuning of SmolLM2-1.7B at a fixed learning rate of $\eta = 3\times10^{-4}$ for different pullback strengths c and EMA powers κ. For large c and small κ, a substantial fraction of coordinates are clipped at 1 throughout training, meaning that the pullback fully transports those coordinates to the EMA point, especially early in training.

4 Empirical Investigation of PACE

Our proposed algorithm, PACE, was derived in a simplified quadratic model and it is thus not *a priori* clear that it will be effective in practice on the non-convex, non-quadratic problems that arise in language model training. In this section, we empirically evaluate PACE on two standard language-modeling regimes: supervised fine-tuning of openweight 1–2B-parameter language models, and from-scratch pretraining of a 124M-parameter model at the Chinchilla-optimal token budget.

Our experiments are primarily organized around three questions: (i) does PACE improve on vanilla AdamW and an EMA thereof across model families and learning rates? (ii) is the improvement robust to the optimizer's three new hyperparameters (pullback strength c, EMA power κ, and update frequency uf)? and (iii) how does PACE compare to alternative baselines like learning rate decay and Schedule-Free [9]? Further ablations are deferred to Section A.

4.1 Empirical Setup

For *post-training*, we finetune three pre-trained language models of similar scale, **SmolLM2-1.7B** [1], **Qwen3-1.7B** [41], and **Gemma3-1B** [11], on the `smol-smoltalk` dataset [2] for one epoch on the training split, plus a small-model dense ablation on **SmolLM2-135M**. For *pretraining*, we train **GPT-2 (124M)** [32] on `FineWeb` [30] at the Chinchilla-optimal token budget of 2.5B tokens [14]. Fine-tuning runs use a single NVIDIA RTX PRO 6000 GPU; the GPT-2 pretraining sweep runs on a Google Cloud TPU v6e (Trillium) pod, with training hyperparameters and resource costs summarized in and around Table 1 of Section A. We report final validation cross-entropy on a held-out `smol-smoltalk` split for fine-tuning and on a fixed FineWeb evaluation shard for pretraining.

4.2 Main Results

In Figure 1, we show the training curves of PACE, EMA, and vanilla AdamW on the fine-tuning task outlined above. We see that PACE substantially improves on EMA at the same per-model learning rate. The curves report three-seed means under WSD for all three methods; a single-seed comparison with the baselines at their best decay schedules appears in Fig. 16. In Figure 2, we show the pretraining regime, where we again see a clear improvement of PACE over EMA. In Figure 10 (deferred to the appendix), we repeat these experiments for Qwen on the Tulu dataset [21], with similar results. In all cases, we compare the best configurations of EMA and vanilla training to PACE, where we swept across learning rate, decay schedule, EMA power κ, and pullback strength c to find the best configuration for each baseline. Thus, these results suggest that PACE can significantly improve on the performance of the iterate-average estimator in practice, and that this improvement is robust across model families and datasets.

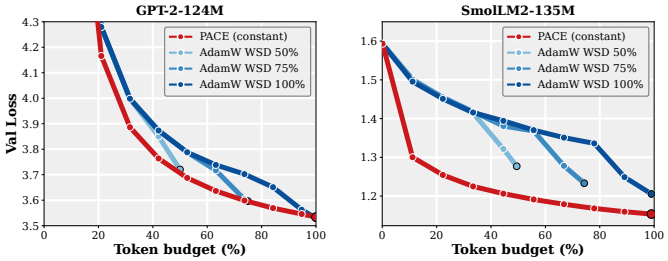

Figure 5: **Token-budget comparisons on GPT-2-124M pretraining (a) and SmolLM2-135M fine-tuning (b).** Validation cross-entropy of PACE at a constant learning rate against AdamW with a WSD decay launched at 50%, 75%, or 100% of the budget. In pretraining each decay branch ends on the PACE curve, so the decay recovers only what PACE reaches without a schedule; in fine-tuning every branch ends above it.

4.3 Further Empirical Results

In addition to our main results, we also conduct a number of ablations and further experiments to better understand the behavior of PACE and its sensitivity to its hyperparameters. While many results are deferred to Section A in the interest of space, we briefly summarize some of the key findings.

Effect of pullback strength c, EMA power κ, and update frequency. In each of our settings, we conduct a dense sweep over the pullback strength c and EMA power κ hyperparameters of PACE for several learning rates and update frequencies in order to understand the sensitivity of PACE to these hyperparameters and to verify that the improvement over EMA-evaluated AdamW is robust. An example from our fine-tuning experiments is shown in Figure 3, which is broadly replicated across models and settings (Figures 6 and 8). In particular, we see that relatively small values of c (e.g. $c = 3 \times 10^{-3}$) can substantially improve on EMA-evaluated AdamW, and that the improvement is robust across a range of learning rates and update frequencies.

Effect of clipping λ. In (6) and Algorithm 1, we clip the pullback gain λ to ensure that the update is always a convex combination of the iterate average and the naïve optimizer step. When the clipping is instantiated, PACE transports the relevant coordinate of the current iterate θ_k to exactly equal that of the EMA point $\widehat{\theta}_{\mathsf{EMA},k}$, recovering a special case of Lookahead [44]. In Fig. 4, we show how the fraction of clipped coordinates varies with different hyperparameters throughout training on SmolLM2-1.7B at $\eta = 3{\times}10^{-4}$, observing that in extreme cases, essentially all coordinates are clipped (especially early in training), but generally clipping is not enforced for most coordinates.

Effect of LR Decay. While our theory assumed constant learning rate, decay schedules are a standard part of practical training pipelines. We experiment with three decay schedules: (i) constant learning rate, (ii) linear decay to zero over the last 20% of steps, and (iii) cosine decay to zero. In Figure 2 (right) we compare PACE to AdamW with linear decay to zero over the last 20% of steps in pretraining, finding that PACE strictly improves even in this setting. Exhaustive results for all three schedules can be found in Sections A and B.1; in summary, PACE provides improvement over the baseline across all schedules, and the improvement is robust to the choice of schedule.

Comparing to the schedule-free baseline. We also compare PACE to the most prominent schedule-free optimizer (SF) [9], which also uses iterate averaging to stabilize training and remove the need for learning rate decay. In Figure 15, we compare PACE to Schedule-Free and observe competitive performance. While Schedule-Free requires one fewer copy of model weights than our naïve PACE implementation, it must switch between 'fast' and 'slow' moving iterate averages during training, leading to slower training (12 vs 10 hours on a v6e TPU for Qwen3-1.7B in our experiments). Finally, in Figure 5b, we show that PACE applied with a constant learning rate outperforms AdamW with WSD at every token budget, suggesting that PACE can be an effective alternative to learning rate decay. Similar results hold for pretraining, as shown in Figure 5a, providing further evidence that

PACE can be an effective alternative to learning rate decay, although we emphasize that improved performance is typically achieved by combining PACE with a decay schedule.

5 Discussion

In this work, we introduced PACE, a simple modification to existing LM optimization algorithms, formally derived in an optimal control formulation, that explicitly controls the training trajectory to improve the performance of the iterate-average estimator. We then showed empirically that PACE can improve training performance in both pre- and post-training settings.

Limitations. A theoretical limitation is the restrictiveness of the assumptions under which PACE was derived. This is partly mitigated by the convergence guarantee in the convex setting, but rigorously incorporating momentum and adaptive preconditioning into the analysis of the optimizer is an important direction for future work. There are several empirical limitations of the work as well. First, due to resource constraints, we were unable to verify the efficacy of PACE at model scales larger than 2B parameters for post-training and 124M parameters for pretraining. Second, PACE requires maintaining an additional copy of model weights relative to plain AdamW, but not relative to AdamW with EMA; a more detailed exploration of the memory overhead and mitigations thereof will be important for practical implications.

Future Work. In addition to the directions for future work mentioned above, there are at least two additional questions raised by our work. First, because our analysis is on SGD, our convergence guarantees do not require momentum and it is natural to explore the extent to which the pullback term alone can compensate for the lack of momentum; this would be a significant practical benefit of the method, as it would reduce the memory overhead of PACE by one copy of model weights. Second, throughout our empirical study, we used AdamW as a base optimizer, but modern LM pretraining pipelines increasingly use second-order methods such as SOAP [37] and MuON [16]; exploring the extent to which PACE can be adapted to these optimizers and whether it provides similar benefits in that setting is an important direction for future work. Moreover, it is natural to ask if these second-order preconditioners could be applied to the pullback term itself, which would be a more direct implementation of the optimal-control solution in (4).

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the Google TPU Research Cloud for providing some of the compute resources used in this work. We gratefully acknowledge use of the research computing resources of the Empire AI Consortium, Inc, with support from Empire State Development of the State of New York, the Simons Foundation, and the Secunda Family Foundation. AB thanks Capital One for support through the Capital One Center for AI and Responsible Financial Innovation at Columbia.

References

[1] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. SmolLM2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

[2] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. SmolTalk: A synthetic instruction-tuning dataset accompanying SmolLM2. https://huggingface.co/datasets/HuggingFaceTB/smoltalk, 2025. Released alongside SmolLM2 [1].

[3] Adam Block and Cyril Zhang. Ema without the lag: Bias-corrected iterate averaging schemes. *arXiv preprint arXiv:2508.00180*, 2025.

[4] Adam Block, Youssef Mroueh, and Alexander Rakhlin. Generative modeling with denoising auto-encoders and langevin sampling. *arXiv preprint arXiv:2002.00107*, 2020.

[5] Adam Block, Dylan J Foster, Akshay Krishnamurthy, Max Simchowitz, and Cyril Zhang. Butterfly effects of sgd noise: Error amplification in behavior cloning and autoregression. In *The Twelfth International Conference on Learning Representations*, 2024.

[6] Peter J Brockwell and Richard A Davis. *Introduction to time series and forecasting*. Springer, 2002.

[7] Dan Busbridge, Jason Ramapuram, Pierre Ablin, Tatiana Likhomanenko, Eeshan Gunesh Dhekane, Xavier Suau Cuadros, and Russell Webb. How to scale your ema. *Advances in Neural Information Processing Systems*, 36:73122–73174, 2023.

[8] Aram Davtyan, Sepehr Sameni, Llukman Cerkezi, Givi Meishvili, Adam Bielski, and Paolo Favaro. Koala: A kalman optimization algorithm with loss adaptivity. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6471–6479, 2022.

[9] Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[10] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[11] Gemma Team, Google DeepMind. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[12] Tryphon T Georgiou and Anders Lindquist. The separation principle in stochastic control, redux. *IEEE Transactions on Automatic Control*, 58(10):2481–2494, 2013.

[13] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pages 1842–1850. PMLR, 2018.

[14] 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, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

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

[17] Jean Kaddour. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Yury A Kutoyants, Yury A Kutoyants, and Y Kutoyants. Statistical inference for ergodic diffusion processes. 2004.

[20] Huibert Kwakernaak and Raphael Sivan. *Linear optimal control systems*, volume 1. Wiley-interscience New York, 1972.

[21] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Christopher Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. TÜLU 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[22] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[23] Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pretraining. *arXiv preprint arXiv:2305.14342*, 2023.

[24] Ilya Loshchilov, Frank Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5):5, 2017.

[25] Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35: 7697–7711, 2022.

[26] Stephan Mandt, Matthew D Hoffman, David M Blei, et al. Continuous-time limit of stochastic gradient descent revisited. *NIPS-2015*, 2015.

[27] Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, Sham Kakade, and Alex Damian. A defense of the quadratic model. *arXiv preprint arXiv:2607.21716*, 2026.

[28] Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

[29] Matteo Pagliardini, Pierre Ablin, and David Grangier. The AdEMAMix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.

[30] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2024.

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

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

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

[34] Robert H Shumway and David S Stoffer. *Time series analysis and its applications: with R examples*. Springer, 2006.

[35] Minhak Song, Beomhan Baek, Kwangjun Ahn, and Chulhee Yun. Through the river: Understanding the benefit of schedule-free methods for language model training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. arXiv:2507.09846.

[36] Henk Van de Water and Jan Willems. The certainty equivalence property in stochastic control theory. *IEEE Transactions on Automatic Control*, 26(5):1080–1087, 1981.

[37] Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

[38] Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

[39] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022.

[40] Zixuan Xia, Aram Davtyan, and Paolo Favaro. Koala++: Efficient kalman-based optimization with gradient-covariance products. *arXiv preprint arXiv:2506.04432*, 2025.

[41] An Yang, Anwen Anwar, Baosong Bao, Beichen Bi, Bo Cai, Changjian Chen, Chao Chen, Daohai Chen, Daniel Chen, Daoguang Cheng, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[42] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls: Hamiltonian systems and HJB equations*, volume 43. Springer Science & Business Media, 1999.

[43] Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

[44] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.

[45] Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. *Advances in neural information processing systems*, 28, 2015.

Contents

1 Introduction **1**

2 Algorithm Derivation and Guarantees **3**

2.1 Formal Problem Setup 3

2.2 Derivation of the Optimal Control Strategy 4

2.3 Convergence Guarantee in the Convex Setting 5

3 Introducing PACE: Pullback Averaging Control for Efficient Optimization **7**

4 Empirical Investigation of PACE **8**

4.1 Empirical Setup 8

4.2 Main Results 8

4.3 Further Empirical Results 9

5 Discussion **10**

A Further Details on Empirical Setup **15**

A.1 Supplementary figures 15

B Additional Results **22**

B.1 Fine-tuning 22

B.2 Pretraining 22

C Token-Budget Comparisons with WSD **23**

D Technical Preliminaries **26**

D.1 Linear Quadratic Gaussian (LQG) Problems 26

D.2 Autoregressive Processes and the Yule-Walker Equations 27

E Mathematical Derivation of PACE **28**

E.1 Optimal Control for the Relaxed Objective 29

E.2 Optimal Algorithm for the Original Objective 30

E.3 Generalizing to the Case Where $\mu^\star$ is Unknown 32

E.4 Estimating the optimum under controlled dynamics 34

E.5 Deriving a Practical Algorithm through Approximation 34

F Proofs of Convergence Results **35**

F.1 Proof of Theorem 3 35

F.2 Proof of Proposition 2 38

F.3 Proof of Proposition 3 40

F.4 Stability and relative improvement in the quadratic model 40

Table 1: Per-run training budgets. Fine-tuning runs are one epoch unless noted; pretraining uses 2.5B-tokens.

Model	Params	Tokenizer	Seq. len	Eff. batch / steps
SmolLM2-135M	135M	SmolLM2	1280	128 / 889
SmolLM2-1.7B	1.7B	SmolLM2	1280	256 / 1,778
Qwen3-1.7B	1.7B	Qwen BPE	1280	256 / 1,779
Gemma3-1B	1.0B	Gemma3	1280	256 / 1,779
GPT-2-124M	124M	GPT-2 BPE	1024	512 / 4,750

The SmolLM2-135M method-comparison pool used in Fig. 15 uses a long-budget rerun with 3,557 steps.

A Further Details on Empirical Setup

In this appendix we collect the additional information needed to reproduce the experiments in Section 4: training data, models, optimizer hyperparameters, per-experiment grids, hardware and compute, evaluation protocol, and the supplementary figures referenced from the body.

Training data. We use `smol-smoltalk` [2] as our default fine-tuning corpus, using the fixed train/validation split prepared for these experiments and packing examples to sequence length 1280. Some supplementary sweeps use `tulu-3` [21] with the same supervised fine-tuning pipeline. For pretraining, we train GPT-2-124M on `FineWeb` [30].

Models. Table 1 summarises the five model families we consider. The fine-tuning models are initialized from the official Hugging Face checkpoints. SmolLM2 and Gemma3 use their native tokenizers.

Training. Unless stated otherwise for the Schedule-Free and WSD baselines, AdamW-family runs use the standard defaults $(\beta_1, \beta_2) = (0.9, 0.999)$, $\varepsilon = 10^{-8}$, weight decay $\omega = 10^{-2}$, gradient clipping at 1.0, and a 50-step linear warmup followed by a constant learning rate (no decay). Training uses `bfloat16` mixed precision. When needed, we use gradient accumulation to reach the reported effective batch size. The learning rate η varies per experiment and is reported in Table 2.

Optimizer hyperparameters. PACE adds three tunable hyperparameters on top of standard AdamW: the pullback strength c, the EMA power κ, and the update frequency uf (per-experiment values in Table 2). We use $\gamma = 1.0$, $\rho = 0$, and clamp the per-parameter (per-coordinate) pullback gain $\lambda_{t,i}$ at 1, matching the implementation defaults. For Schedule-Free [9] we use the authors' reference implementation under their published NanoGPT recipe (β_1=0.98, β_2=0.95, weight decay 0.05, no gradient clipping). AdamW with EMA is the matched $c = 0$ comparison.

Hardware and compute. Fine-tuning runs used NVIDIA RTX PRO 6000 Blackwell (96 GB) cards; the GPT-2 pretraining sweep used Google Cloud TPU v6e (Trillium) chips. Per-run budgets are listed in Table 1.

Evaluation. For supervised fine-tuning we report held-out cross-entropy on the fixed validation split for each dataset. For pretraining we report cross-entropy on a fixed FineWeb validation shard. EMA-family methods are evaluated using θ_t^{EMA} rather than the live iterate (we swap weights at evaluation time and restore them afterwards); plain AdamW reports its live-weight value; Schedule-Free reports its native vanilla evaluation since SF does not maintain a separate estimator.

Reproducibility. The code, configurations, and reproduction instructions are available at the GitHub repository linked in the abstract.

A.1 Supplementary figures

This subsection collects the full hyperparameter-sweep and figures referenced from Section 4 and the result appendices; takeaways are stated in the captions and discussed where each figure is cited.

Table 2: Sweep space used by the body and appendix figures.

Figures	Cell	Sweep space
1, 6, 11	1B fine-tuning	SmolLM2-1.7B: $\eta \in \{3, 7, 10\}\times10^{-4}$ Qwen3-1.7B: $\eta \in \{3, 5\}\times10^{-4}$ Gemma3-1B: $\eta \in \{10^{-3}, 5\times10^{-3}\}$ Plotted c bins $\in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}, 3\times10^{-2}, 10^{-1}, 3\times10^{-1}\}$ $\kappa \in \{0.2, 0.5, 0.7\}$, uf$=1$
10	Qwen3-1.7B Tulu	$\eta \in \{3, 5\}\times10^{-4}$, $\kappa \in \{0.2, 0.5, 0.7\}$ $c \in \{5\times10^{-2}, 10^{-1}, 1.5\times10^{-1}, 2.5\times10^{-1}\}$
4, 12, 13	SmolLM2-1.7B clipping	Same SmolLM2-1.7B fine-tuning runs as above; clipping telemetry is reported for $\eta \in \{3\times10^{-4}, 10^{-3}\}$, $\kappa \in \{0.2, 0.5, 0.7\}$, and $c \in \{5\times10^{-3}, 1.5\times10^{-2}, 5\times10^{-2}, 1.5\times10^{-1}, 5\times10^{-1}\}$.
3, 8	SmolLM2-135M	Dense sweep: $\eta \in \{1,\ 1.5,\ 2,\ 3,\ 5\}\times10^{-3}$ $c \in \{0,\ 10^{-4},\ 10^{-3},\ 3\times10^{-3},\ 10^{-2}, 3\times10^{-2},\ 10^{-1},\ 3\times10^{-1}\}$ $\kappa \in \{0.3,\ 0.5,\ 0.7\}$, uf $\in \{1,\ 5,\ 10\}$
15, 7, 14, 17	Method and WSD comparisons	SmolLM2-135M: AdamW/EMA/PACE at $\eta \in \{10^{-3}, 5\times10^{-3}\}$; Schedule-Free: $\eta \in \{10^{-3},\ 3\times10^{-3},\ 5\times10^{-3},\ 10^{-2}\}$ WSD baselines use $\eta \in \{1, 1.5, 2, 3, 5\}\times10^{-3}$ Qwen3-1.7B: AdamW/EMA/PACE at $\eta \in \{3, 5\}\times10^{-4}$; Schedule-Free at $\eta \in \{3\times10^{-4}, 10^{-3}\}$.
2, 9, 20	GPT-2-124M	$\eta \in \{3\times10^{-4},\ 10^{-3},\ 1.5\times10^{-3},\ 2\times10^{-3}, 2.5\times10^{-3},\ 3\times10^{-3},\ 10^{-2}\}$ c: coarse grid 10^{-4}–3×10^{-1} plus local grid 1×10^{-3}–10^{-2} $\kappa \in \{0.2,\ 0.5,\ 0.7\}$, uf$=1$
18, 19	1B schedules and multi-seed runs	Schedules $\in$ {constant, cosine, WSD} for AdamW/EMA at $\eta = 10^{-3}$ for SmolLM2-1.7B and Gemma3-1B, and $\eta = 5\times10^{-4}$ for Qwen3-1.7B; PACE at a constant η. The multi-seed figure uses constant and WSD schedules, with three seeds per method.
15	SmolLM2-135M budgets	AdamW WSD at token budgets $\{444, 667, 889\}$ steps, $\eta \in \{3, 5\}\times10^{-3}$; PACE at a constant learning rate over the full budget (c=10^{-1}, κ=0.3).
2, 20, 21	GPT-2-124M schedules	Shared grid per schedule (constant, cosine, WSD) at η=2×10^{-3}: $c \in \{3\times10^{-4}, 10^{-3}, 2\times10^{-3}, 3\times10^{-3}, 5\times10^{-3}, 7\times10^{-3}, 10^{-2}\}$ $\kappa \in \{0.1, 0.2, 0.3, 0.5\}$, plus AdamW and EMA, with three seeds per configuration.
5a	GPT-2-124M budgets	WSD at token budgets $\{2375, 3563, 4750\}$ steps (η=2×10^{-3}, κ=0.5, c=3×10^{-3}); the token-budget overlay uses PACE at a constant learning rate (c=2×10^{-3}, κ=0.5).

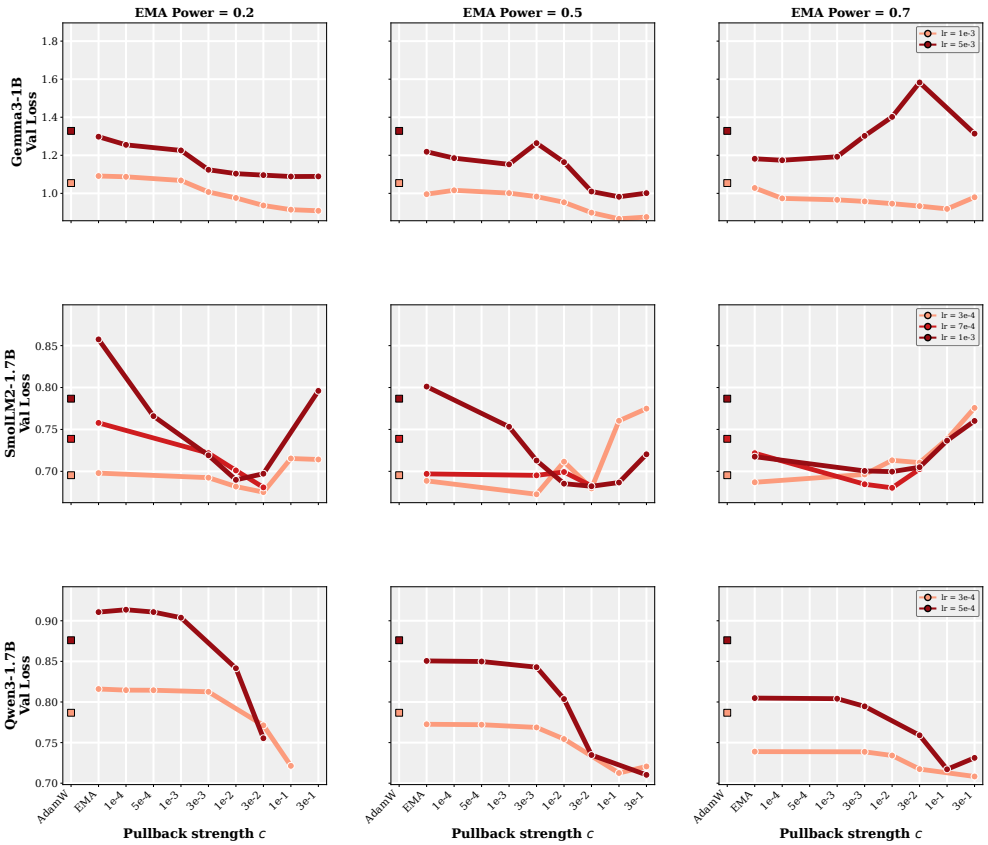

Figure 6: **Effect of EMA power and pullback strength on SmolLM2-1.7B, Qwen3-1.7B, and Gemma3-1B.** Validation cross-entropy across learning rates, pullback strengths, and EMA powers. Optimal pullback strength remains relatively robust to learning rate and EMA power, with improvement over the EMA baseline across a wide range of settings and models.

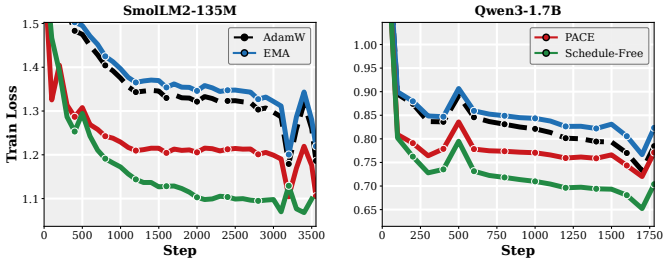

Figure 7: **Training-loss comparison of PACE to Schedule-Free [9], AdamW, and EMA on fine-tuning.** Smoothed training loss on the same model selections as Fig. 15. Although PACE has higher training loss than Schedule-Free in this view, its validation loss is competitive in Fig. 15.

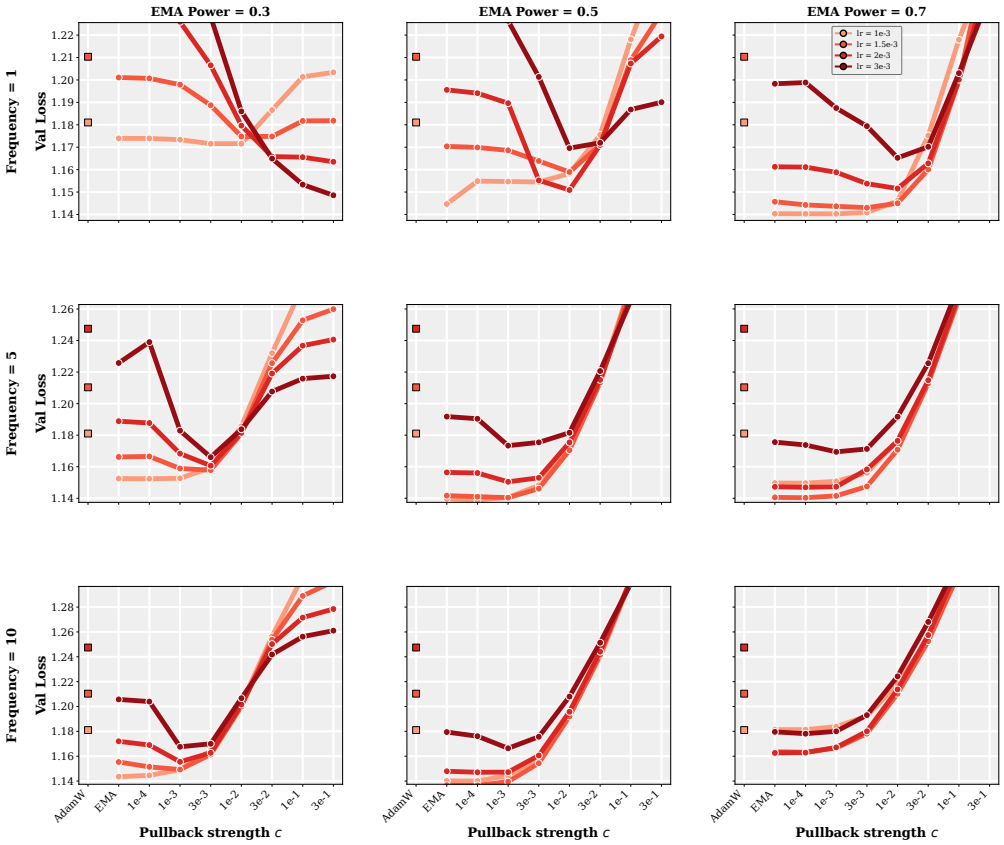

Figure 8: **Effect of update frequency, EMA power, and pullback strength.** Validation cross-entropy of SmolLM2-135M for different learning rates, pullback strengths, EMA powers, and update frequencies. Optimal pullback strength remains relatively robust to learning rate and update frequency, with improvement over the EMA baseline across a wide range of settings.

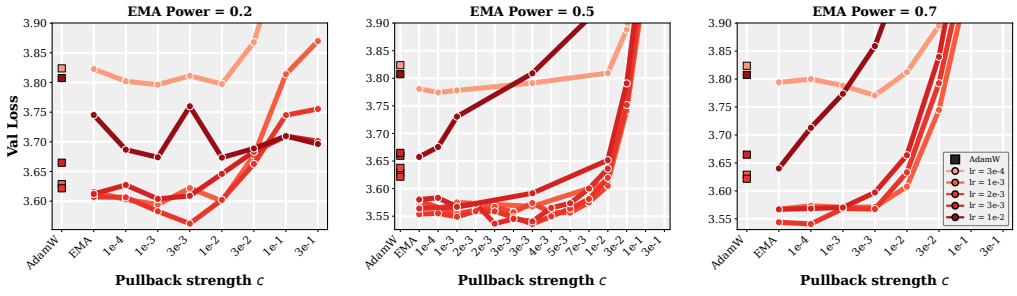

Figure 9: **Effect of EMA power and pullback strength on pretraining.** Validation cross-entropy on GPT-2-124M trained on FineWeb for different learning rates, pullback strengths, and EMA powers. Optimal pullback strength remains robust, with improvement over the EMA baseline across a wide range of settings.

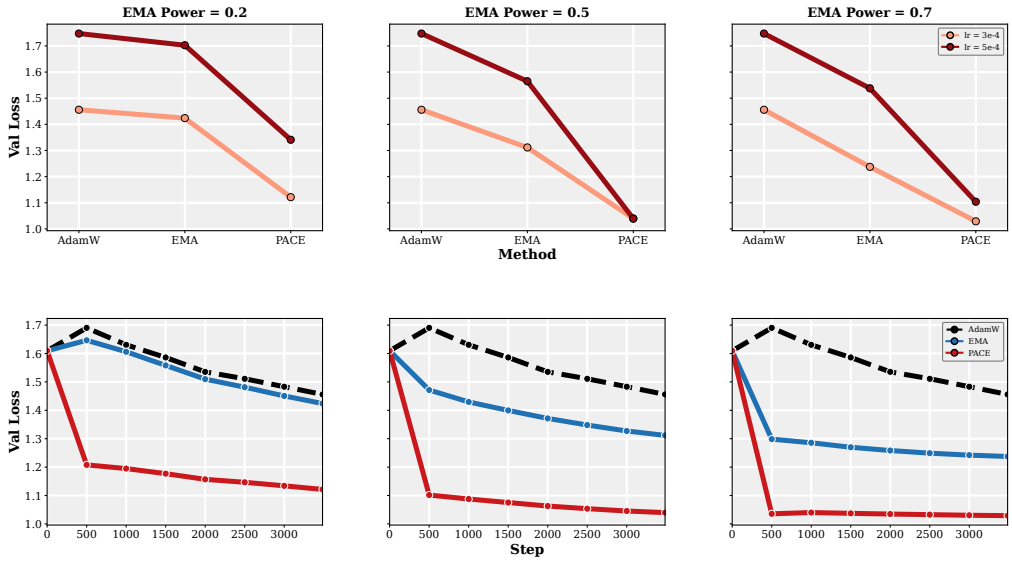

Figure 10: **Effect of EMA power and pullback strength on Qwen3-1.7B Tulu-3 fine-tuning.** Validation cross-entropy across learning rates and EMA powers. **Top:** final validation loss for AdamW, EMA, and PACE. **Bottom:** validation trajectories at $\eta = 3\times10^{-4}$. PACE improves over the matched AdamW and EMA baselines across the reported settings.

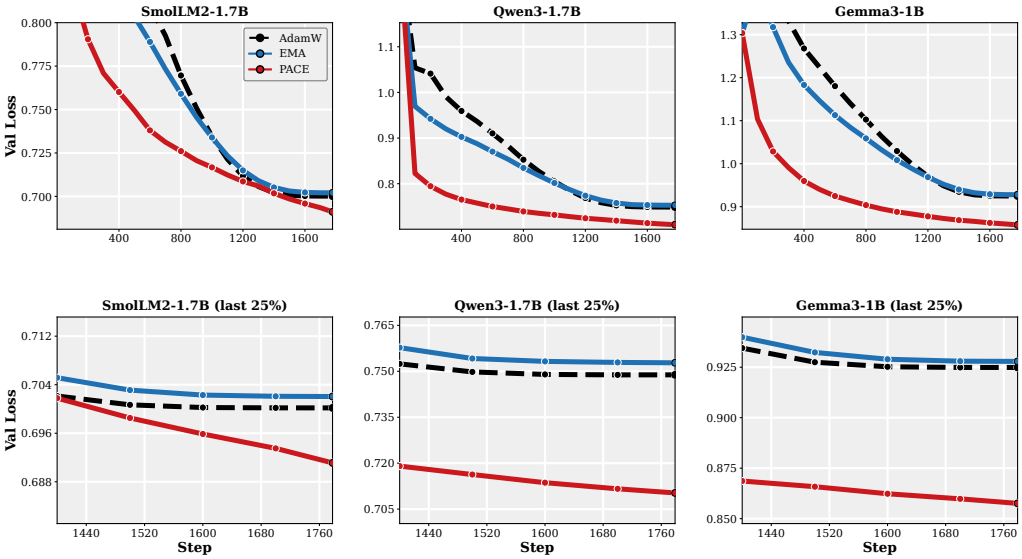

Figure 11: **Performance of PACE on fine-tuning.** Validation cross-entropy on `smol-smoltalk` for SmolLM2-1.7B (**left**), Qwen3-1.7B (**middle**), and Gemma3-1B (**right**). For each model, PACE uses a constant learning rate, while the AdamW and EMA baselines use their best learning-rate-decay schedule (cosine or WSD). The top row shows the full validation trajectories, and the bottom row zooms into the final quarter of training. PACE strictly improves on both baselines on all three models; see Figure 19 for a three-seed robustness study.

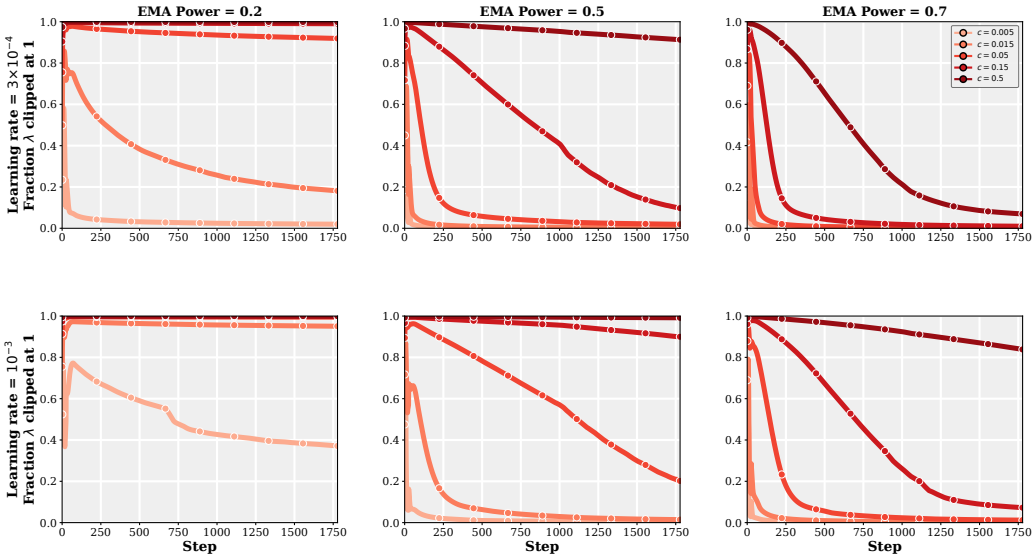

Figure 12: **Fraction of clipped updates over training.** Fraction of parameters with saturated pullback gain $\lambda_{t,i} = 1$ on SmolLM2-1.7B, across learning rate, EMA power, and pullback strength. Saturation increases with pullback strength and is most pronounced for smaller EMA power.

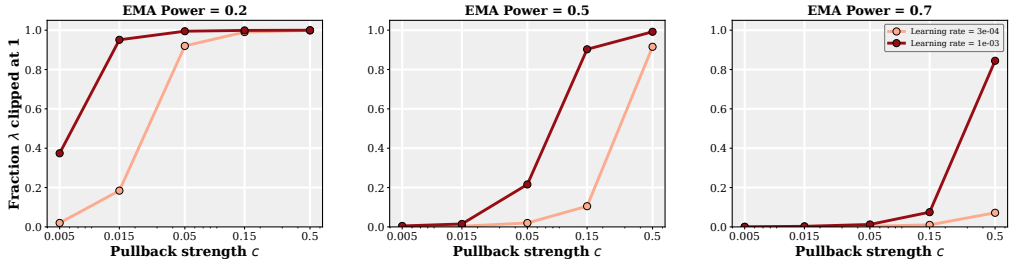

Figure 13: **End-of-training fraction of clipped updates.** Fraction of parameters with saturated pullback gain $\lambda_{t,i} = 1$ on SmolLM2-1.7B as a function of pullback strength and learning rate. Smaller EMA power reaches saturation at smaller pullback strengths.

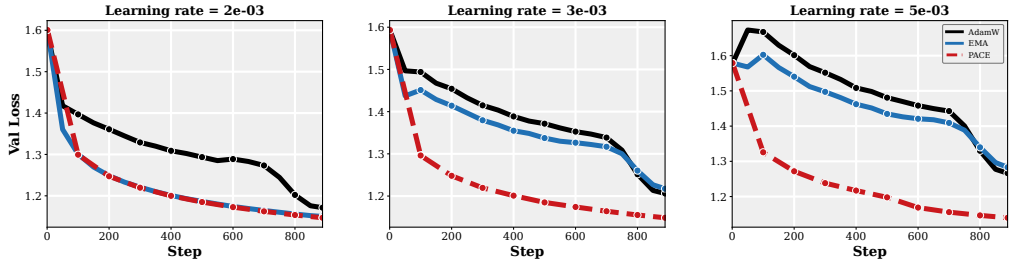

Figure 14: **PACE against AdamW and EMA under fine-tuning.** Validation cross-entropy on SmolLM2-135M at three learning rates, with the baselines given their best decay schedule and PACE held at a constant learning rate. PACE ends below both baselines at every learning rate, and its margin over them grows sharply as the learning rate increases.

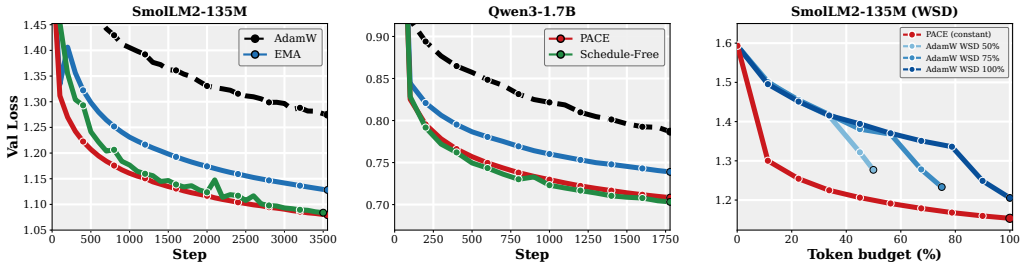

Figure 15: **Comparison of PACE to Schedule-Free [9], AdamW, and EMA on fine-tuning.** Validation cross-entropy on `smol-smoltalk` for SmolLM2-135M (**left**), Qwen3-1.7B (**middle**), and a token-budget comparison on SmolLM2-135M (**right**): AdamW with WSD decays the learning rate to zero over the last 20% of token budgets of $50\%/75\%/100\%$ of the run, while PACE trains at a constant learning rate over the full budget. PACE improves on AdamW and EMA-only and is competitive with Schedule-Free; in the right panel, every WSD branch ends above the PACE trajectory at every budget.

B Additional Results

B.1 Fine-tuning

The advantage of PACE over AdamW and EMA is robust: it holds whatever learning-rate schedule the baselines are given (Figure 18) and across three random seeds (Figure 19).

Baselines across all learning-rate schedules. Even when AdamW and EMA are given their best schedule—constant, cosine, or WSD—PACE at a constant learning rate stays below both, on all three models (Figure 18).

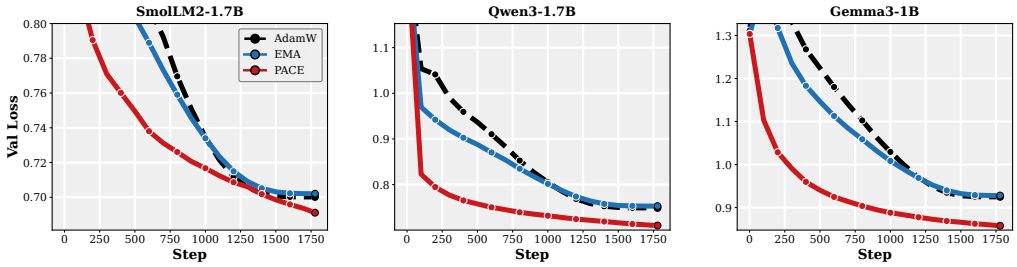

Figure 16: **Performance of PACE on fine-tuning.** Validation cross-entropy on `smol-smoltalk` for SmolLM2-1.7B (**left**), Qwen3-1.7B (**middle**), and Gemma3-1B (**right**). For each model, PACE uses a constant learning rate, while the AdamW and EMA baselines use their best learning-rate-decay schedule (cosine or WSD). PACE strictly improves on both baselines on all three models; see Figure 19 for a three-seed robustness study.

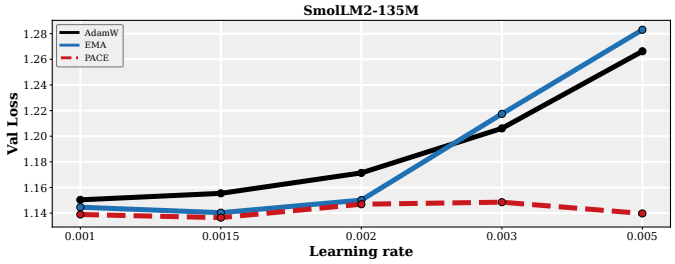

Figure 17: **Fine-tuning across learning rates.** Best validation cross-entropy on SmolLM2-135M for AdamW, EMA, and PACE, with the baselines given their best decay schedule and PACE held at a constant learning rate. PACE attains the lowest loss at every learning rate, by a margin that widens as the learning rate increases and the baselines degrade.

Multi-seed robustness. Across three seeds, the entire PACE band stays below those of AdamW and EMA at every step—under both a constant learning rate and WSD, on all three models—so the improvement exceeds the seed-to-seed spread (Figure 19).

B.2 Pretraining

In pretraining, PACE beats both the AdamW and EMA baselines under every learning-rate schedule, and the best pullback strength is the same under all of them. The figures show the full pullback-strength sweep on GPT-2-124M (trained from scratch on `FineWeb` at the Chinchilla-optimal budget), as the mean over three seeds.

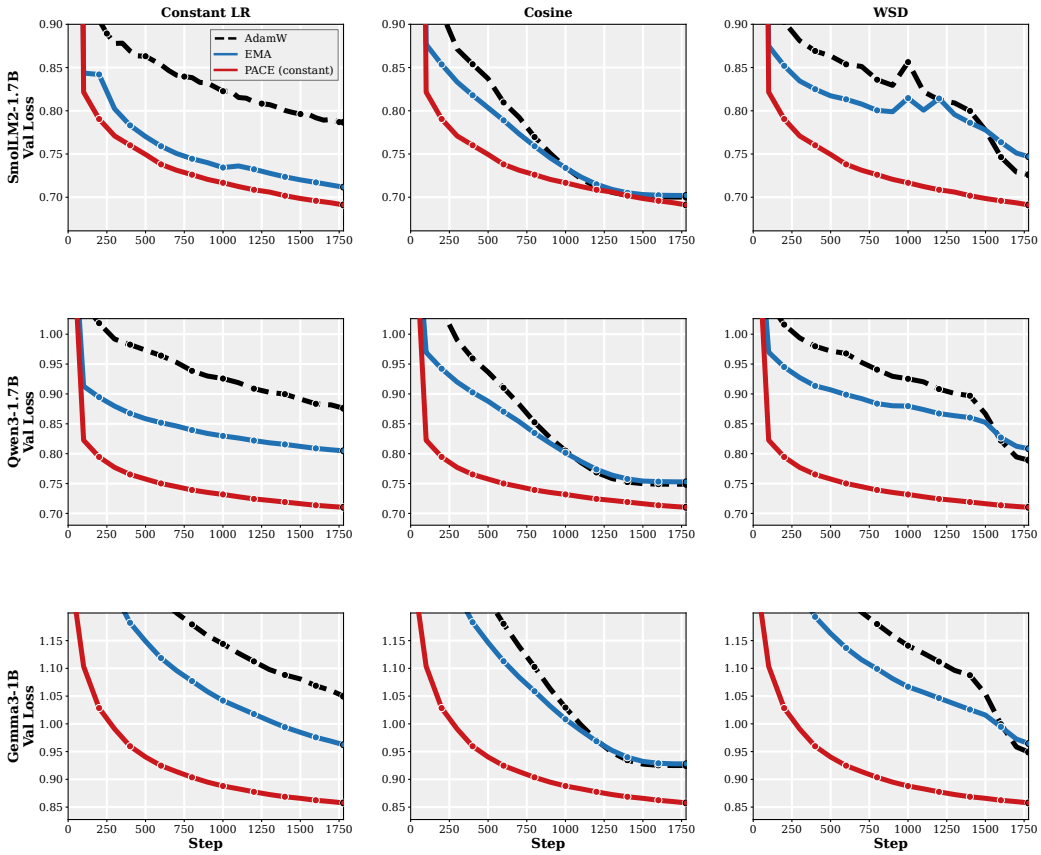

Figure 18: **PACE against the baselines under every learning-rate schedule.** Validation cross-entropy on `smol-smoltalk`; rows are models, columns are the baselines' learning-rate schedule (constant, cosine, WSD), with the PACE constant-learning-rate run as the reference in each panel. PACE improves on both baselines under every schedule on all three models.

Multi-seed pullback-strength sweep across schedules. Figure 20 sweeps the pullback strength c under all three schedules (the AdamW and best-κ EMA baselines are the two leftmost ticks). Two things stand out. First, at its optimum PACE beats both baselines under every schedule, and the improvement holds across a band of pullback strengths ($10^{-3} \le c \le 3{\times}10^{-3}$)—wider still under WSD and cosine. Second, the optimum is the same ($c \approx 3{\times}10^{-3}$) under all three schedules, so the choice of c transfers. WSD with PACE is best overall (3.507 ± 0.006, below the constant-LR optimum of 3.537 ± 0.004 by several times the seed spread), and PACE at a constant learning rate comes within 0.006 of AdamW under WSD (3.537 vs. 3.531) with no schedule at all. Figure 21 resolves the same grid by the EMA power κ (at seed 42) and shows a basin around the optimal c at every κ.

C Token-Budget Comparisons with WSD

In Figures 5a and 5b we compared PACE at a constant learning rate to AdamW with linear decay to zero. In this appendix we make that comparison systematic for pretraining and interpret the token-budget comparisons in both regimes. The analysis is inspired by the token-budget comparisons used to study Schedule-Free methods [9, 35], and we read it through the river-valley description of the loss landscape [35, 38]: at a constant learning rate, the iterate oscillates along the steep "hill" directions of a valley while making steady progress along its flat "river" floor, and the decay phase of WSD acts by suppressing this oscillation and bringing the iterate down to the floor. This perspective makes clear both the appeal of WSD and its cost: although the decay phase produces the characteristic sharp

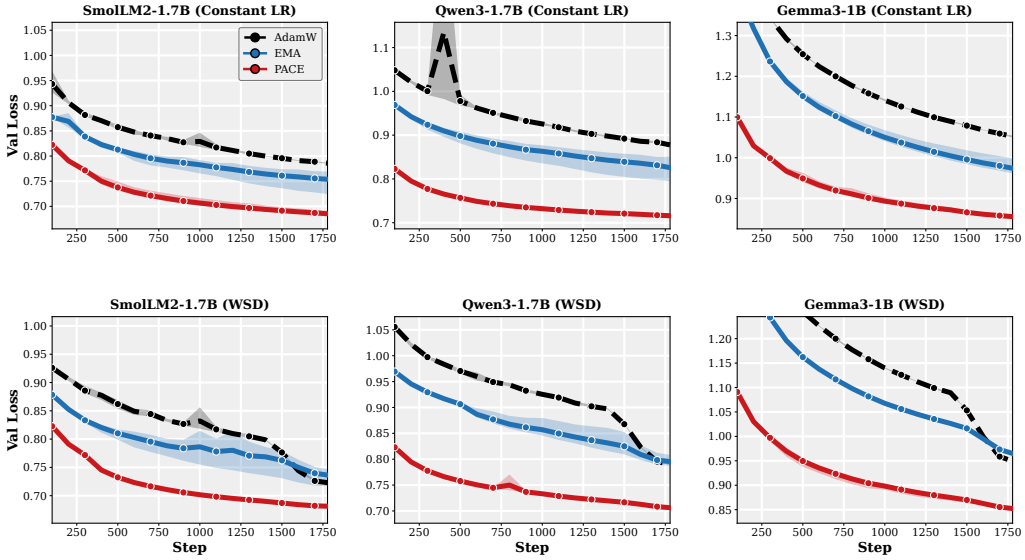

Figure 19: **Multi-seed robustness of fine-tuning.** Across three seeds, PACE stays below both AdamW and EMA on all three models, under both a constant learning rate and WSD, by more than the seed-to-seed variation at every step.

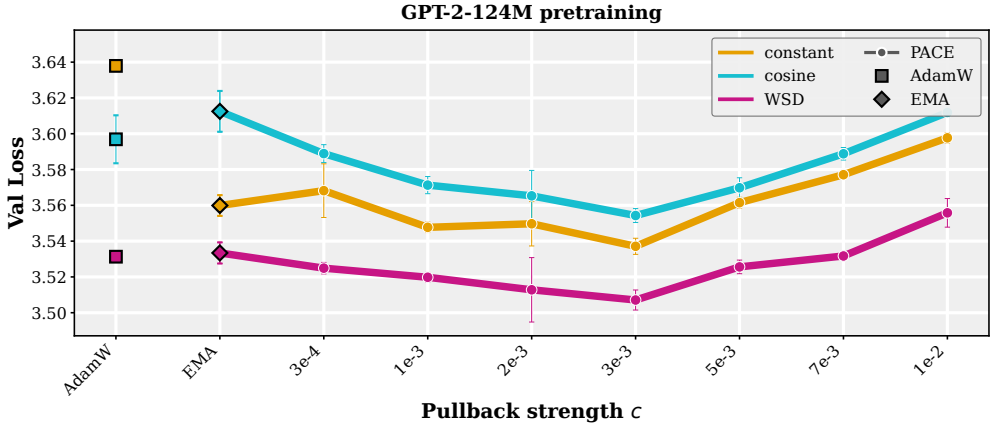

Figure 20: **Pullback-strength sweep on GPT-2-124M pretraining.** Validation loss against the pullback strength c under constant, cosine, and WSD learning-rate schedules; the AdamW and EMA baselines are the two leftmost ticks. Each point is the mean over three seeds and the error bar is one standard deviation. Under every schedule the best c sits below both baselines by more than the seed spread.

drop in loss at the end of training, until it begins the loss curve overstates the loss the model could achieve, so that model quality is difficult to assess mid-training, and once it begins, the remainder of the run is committed to a single token budget. An optimizer whose returned iterate already lies on the river floor incurs neither cost: its loss curve reflects post-decay performance at every step of training, and training may be stopped or continued at any point.

The token-budget comparison tests exactly this property. Because a decay branch primarily removes oscillation while making little further progress along the river, the endpoint of a branch launched at a given budget estimates the position of the river at that budget. Comparing the endpoints of branches

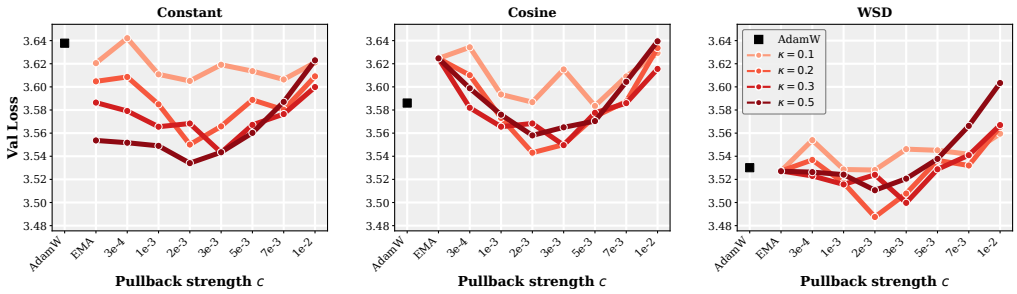

Figure 21: **Pullback-strength sweep resolved by EMA power** on GPT-2-124M pretraining (seed 42), under a constant learning rate (**left**), cosine decay (**middle**), and WSD (**right**). A basin spanning $c \approx 2\text{–}3\times10^{-3}$ appears at every κ under every schedule.

launched at several budgets to a candidate trajectory thus determines whether that trajectory tracks the river: endpoints that coincide with the trajectory indicate that it already lies on the floor, while endpoints that fall below it reveal residual oscillation that a decay phase would have removed. We run this comparison in both regimes—branches at $50\%/75\%/100\%$ of the pretraining budget, and at three budgets and two learning rates in fine-tuning—with all runs sharing data, batch size, and base learning rate.

Pretraining. In Figure 5a, we overlay the AdamW decay branches on the PACE constant-learning-rate trajectory. Every branch lands on the trajectory: the endpoints $(3.721/3.597/3.530)$ match PACE at the same budgets ($\approx 3.707/3.595/3.534$) to within 0.014, and to within 0.004 at the full budget—closer than the seed-to-seed spread. At every budget, decaying the learning rate only recovers the loss that PACE already reaches without a schedule.

Fine-tuning. The fine-tuning comparison in Figure 5b yields a stronger conclusion: every budget-matched branch ends above the PACE trajectory, with endpoints of $1.277/1.233/1.205$ against PACE at 1.153, and the gap widens at larger learning rates. In this regime, no decay branch reaches the loss of the constant-learning-rate PACE run at any budget.

Peeling off intermediate checkpoints to decay. Across every budget, learning rate, and regime that we tested, no decay branch ends below the PACE trajectory, which is precisely the behavior this comparison associates with a trajectory that already lies on the river floor. The practical consequences are those identified above: the validation curve of PACE reflects post-decay performance at every step, so that model quality is visible throughout training, and the loss that WSD attains only at the designated endpoint of a branch is available at every budget, without fixing a training horizon in advance. In fine-tuning, we find the stronger conclusion that the PACE trajectory lies below every budget-matched branch endpoint, suggesting that the modified trajectory makes faster progress along the river than the trajectory from which the branches descend.

D Technical Preliminaries

In this appendix, we review some technical preliminaries that will be useful for our analysis. In Section D.1, we review some classical results from the theory of linear quadratic stochastic control, which we will use to derive our algorithm. Then in Section D.2 we describe some classical results on autoregressive processes, which we will use to analyze the convergence of our algorithm in the quadratic optimization setting.

D.1 Linear Quadratic Gaussian (LQG) Problems

In this section, we briefly review several classical results from the theory of linear quadratic stochastic control, which will be useful for our analysis. Classical references for this material include Kwakernaak and Sivan [20], Yong and Zhou [42] among many others.

A *Linear Quadratic Gaussian* (LQG) control problem is a stochastic control problem where the dynamics of the system are linear, the cost function is quadratic, and the noise is Gaussian. More particularly, we suppose that $x_t^u \in \mathbb{R}^d$ is a stochastic process satisfying

$$dx_t^u = (\mathbf{A}x_t^u + \mathbf{B}u_t)\,dt + \mathbf{C}W_t, \tag{7}$$

where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are matrices of appropriate dimensions, u_t is a control input (and thus progressively measurable with respect to the filtration generated by W_t), and W_t is a standard Brownian motion in $\mathbb{R}^d$. For fixed time horizon T, we then seek to minimize a cost function of the form

$$J(u) = \mathbb{E}\left[x_T^{u\top}\mathbf{Q}_f x_T^u + \int_0^T \left(x_t^{u\top}\mathbf{Q}x_t^u + u_t^\top \mathbf{R}u_t\right)dt\right], \tag{8}$$

where $\mathbf{Q}_f, \mathbf{Q}, \mathbf{R}$ are positive semidefinite matrices of appropriate dimensions. The intuition for this cost is that we wish to minimize the expected deviation of the *state* x_t^u from the origin, while also ensuring that the control inputs u_t are not too large; we pay both a *final* cost for the end state x_T^u being far from the origin as well as a *running* cost for both intermediate states being far from the origin and control inputs being large. This problem is classical and has been well-studied in the control theory literature. We recall the fundamental result that the optimal control strategy is given by a linear state feedback controller, where the feedback gain is given by the solution to a matrix Riccati differential equation [12, 20].

Theorem 4 ([42] Theorem 6.1)**.** *Suppose that x_t^u evolves according to (7) for some progressively measurable u_t. Suppose further that $\mathbf{Q}, \mathbf{Q}_f, \mathbf{R}$ are all symmetric and positive semidefinite and that $\mathbf{R}$ is positive definite. Let $\mathbf{P} : [0, T] \to \mathbb{R}^{d\times d}$ be the solution to the matrix Riccati differential equation*

$$-\dot{\mathbf{P}}_t = \mathbf{A}^\top \mathbf{P}_t + \mathbf{P}_t\mathbf{A} - \mathbf{P}_t\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P}_t + \mathbf{Q}, \quad \mathbf{P}_T = \mathbf{Q}_f, \tag{9}$$

where $\dot{\mathbf{P}}_t$ denotes the time derivative of $\mathbf{P}_t$. Then it holds that the optimal control u_t that minimizes the cost function $J(u)$ defined in (8) is given by

$$u_t = -\mathbf{R}^{-1}\mathbf{B}^\top\mathbf{P}_t x_t^u$$

and the optimal cost is given by

$$J(u) = x_0^\top \mathbf{P}_0 x_0 + \int_0^T \mathrm{Tr}\left(\mathbf{P}_t\mathbf{C}\mathbf{C}^\top\right)dt.$$

While the Riccati equation provides the optimal controller in full generality, we will not require this in the sequel. We thus state two corollaries, corresponding to special cases of the LQG problem, that will be sufficient for our purposes. In the first, we will ignore any final state cost (so $\mathbf{Q}_f = 0$), and assume that both $\mathbf{Q}$ and $\mathbf{R}$ are scalar multiples of the identity.

Corollary 1. *Suppose that x_t^u evolves according to (7) for some progressively measurable u_t, and that $\mathbf{Q}_f = 0$, $\mathbf{Q} = q \cdot \mathbf{I}$, and $\mathbf{R} = r \cdot \mathbf{I}$ for some $q, r > 0$. Suppose further that $\mathbf{B} = \mathbf{I}$. Then the optimal control u_t that minimizes the cost function $J(u)$ defined in (8) is given by*

$$u_t = -r^{-1}\mathbf{P}_t x_t^u, \quad \textit{where} \quad -\dot{\mathbf{P}}_t = \mathbf{A}^\top P_t + P_t\mathbf{A} - r^{-1}P_t^2 + q\cdot\mathbf{I}, \quad P_T = 0.$$

In the second corollary, we will now ignore the *running* state cost (so $\mathbf{Q} = 0$), and assume that $\mathbf{Q}_f$ and $\mathbf{R}$ are scalar multiples of the identity.

Corollary 2. *Suppose that x_t^u evolves according to (7) for some progressively measurable u_t, and that $\mathbf{Q} = 0$, $\mathbf{Q}_f = q \cdot \mathbf{I}$, and $\mathbf{R} = r \cdot \mathbf{I}$ for some $q, r > 0$. Suppose further that $\mathbf{B} = \mathbf{I}$. Then the optimal control u_t that minimizes the cost function $J(u)$ defined in (8) is given by*

$$u_t = -r^{-1}\mathbf{P}_t x_t^u, \quad \textit{where} \quad \dot{\mathbf{P}}_t = \mathbf{A}^\top P_t + P_t\mathbf{A} - r^{-1}P_t^2, \quad P_T = q \cdot \mathbf{I}.$$

Finally, we specialize to the fully scalar case, where $\mathbf{A}$ is diagonal and prove here a closed-form solution to the Riccati equation. While this calculation is standard, we include it here for completeness.

Lemma 1. *Let $p : [0, T] \to \mathbb{R}$ be the solution to the scalar Riccati equation*

$$-\dot{p}(t) = 2ap(t) - r^{-1}p(t)^2 + q, \quad p(T) = q_f.$$

Let

$$\beta_\pm = r(a \pm \omega), \quad \textit{where} \quad \omega = \sqrt{a^2 + r^{-1}q}.$$

Let

$$\kappa(t) = \frac{q_f - \beta_+}{q_f - \beta_-} \cdot e^{-2\omega(T-t)}.$$

Then

$$p(t) = \frac{\beta_+ - \kappa(t)\beta_-}{1 - \kappa(t)}.$$

Proof. Note that by factoring we have

$$\dot{p}(t) = r^{-1}\left(p(t) - \beta_+\right)\left(p(t) - \beta_-\right).$$

Thus, when $\omega > 0$, we have by a standard separation of variables argument that

$$\frac{p(t) - \beta_+}{p(t) - \beta_-} = \frac{q_f - \beta_+}{q_f - \beta_-} \cdot e^{-2\omega(T-t)}.$$

The result follows by rearranging. □

D.2 Autoregressive Processes and the Yule-Walker Equations

We now review some classical results on autoregressive processes, which we will use to analyze the convergence of our algorithm in the quadratic optimization setting. We refer to [6, 34] for a more detailed review of this material. We begin with a definition of autoregressive processes.

Definition 1. We say a sequence of random variables $\{X_k\}_{k\geq 0}$ is an autoregressive process of order p (AR(p)) if it satisfies, for $k \geq p$,

$$X_k = \sum_{i=1}^{p} \phi_i X_{k-i} + \varepsilon_k,$$

for some $\phi_1, \ldots, \phi_p \in \mathbb{R}$ and some sequence of i.i.d. random variables $\{\varepsilon_k\}_{k\geq 0}$ with mean zero and finite variance. The *characteristic polynomial* of the AR(p) process is given by

$$\phi(z) = 1 - \sum_{i=1}^{p} \phi_i z^i.$$

Moreover, we say that the process is *causal* if it can be represented as

$$X_k = \sum_{j=0}^{\infty} \psi_j \varepsilon_{k-j},$$

with the coefficients ψ_j satisfying $\sum_{j=0}^{\infty} |\psi_j| < \infty$.

Formally causality is necessary for us to ensure a limiting variance; fortunately there exists a simple characterization of causality in terms of the roots of the characteristic polynomial.

Proposition 4 (Property 3.1 [34]). *An AR(p) process is causal if and only if the roots of the characteristic polynomial $\phi(z)$ lie outside the unit circle in the complex plane. In this case, X_k has a well-defined limiting variance.*

We now recall the Yule-Walker equations, which relate the coefficients ϕ_i of an AR(p) process to the autocovariance function $\gamma(j) = \mathbb{E}[X_k X_{k+j}]$.

Theorem 5 (Section 3 [34]). *Let $\{X_k\}_{k\geq 0}$ be a causal AR(p) process with coefficients $\phi_1, \ldots, \phi_p$ such that ε_k has variance σ^2. Then the autocovariance function $\gamma(j) = \mathbb{E}[X_k X_{k+j}]$ satisfies the following equations for all $j \geq 0$:*

$$\gamma(j) = \sum_{i=1}^{p} \phi_i \gamma(j-i) + \sigma^2 \cdot \delta_{j,0},$$

In particular, if $p = 2$, it holds that

$$\begin{aligned}
\gamma(1) &= \phi_1 \cdot \gamma(0) \\
\gamma(2) &= \phi_1 \cdot \gamma(1) + \phi_2 \cdot \gamma(0) \\
\gamma(0) &= \phi_1 \cdot \gamma(1) + \phi_2 \cdot \gamma(2) + \sigma^2.
\end{aligned}$$

Finally, we recall the following result on the limiting variance of an AR(2) process.

Lemma 2. *Let $\{X_k\}_{k\geq 0}$ be a causal AR(2) process with coefficients ϕ_1, ϕ_2 such that ε_k has variance σ^2 with fixed, deterministic X_1, X_2. If $\phi(z) \neq 0$ for all complex $|z| \leq 1$, then the limiting variance of X_k is given by*

$$\lim_{k\to\infty} \mathbb{E}[X_k^2] = \gamma(0).$$

Proof. This follows immediately from Definition 3.7 and Property 3.1 of Shumway and Stoffer [34]. Indeed, let X'_k denote the AR(2) process with the same coefficients ϕ_1, ϕ_2 and noise ε_k but with zero initial conditions $X'_1 = X'_2 = 0$. By causality, X'_k has a well-defined limiting variance given by $\gamma(0)$. Thus it suffices to show that $Y_k = X_k - X'_k$ has zero variance in the limit. Observe that

$$Y_k = \phi_1 Y_{k-1} + \phi_2 Y_{k-2},$$

so it remains to show that $Y_k \to 0$ as $k \to \infty$. Writing out the transition matrix, we see that the eigenvalues thereof are the reciprocals of the roots of $\phi(z)$. Because $\phi(z) \neq 0$ for all $|z| \leq 1$, the eigenvalues of the transition matrix are strictly less than 1 in magnitude, so $Y_k \to 0$ as $k \to \infty$. □

E Mathematical Derivation of PACE

In this appendix we provide a complete formal derivation of our proposed algorithm. Recall that we abstract the master optimization problem as the following stochastic control problem:

$$\begin{aligned}
&\min_u J_T(u) = \mathbb{E}\left[\left\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\right\|^2 + \lambda \cdot \int_0^T \|u_t\|^2 \, dt\right], \qquad (10) \\
&d\theta_t^u = \left[\mathbf{A}\left(\mu^\star - \theta_t^u\right) + u_t\right] dt + \mathbf{\Sigma}^{1/2} dW_t.
\end{aligned}$$

In other words, we are running (a continuous time limit of) stochastic gradient descent on a quadratic objective $f(\theta) = \frac{1}{2}\left(\theta - \mu^\star\right)^\top \mathbf{A}\left(\theta - \mu^\star\right)$ perturbed by some control input u_t, and we wish to choose u_t so as to minimize the mean squared error of the returned iterate $\widehat{\theta}_{\mathsf{EMA},T}$ as an estimator of $\mu^\star$, plus a regularization term ensuring that the control inputs u_t are not too large. We provide a complete characterization of the optimal control strategy for this problem, and then show how to derive a simple approximation to this optimal control strategy that is both practical and empirically effective.

In Section E.1 we first consider a relaxed objective, which leads to a simpler optimal control strategy that helps illustrate the main ideas of the analysis. Then, in Section E.2, we derive the optimal control strategy for the actual objective defined in (10), subject to the assumption that $\mu^\star$ is known to the

learner. Because this is, of course, an unrealistic assumption, we then show how the principle of *certainty equivalence* can be used to derive the optimal controller even when $\mu^\star$ is unknown, and how this optimal controller can be approximated to yield a simple and practical algorithm. Finally, in Section E.3, we show how the optimal controller can be approximated even when $\mu^\star$ is unknown, and how this approximation leads to the form of the controller used in our proposed algorithm.

E.1 Optimal Control for the Relaxed Objective

We begin with the simplest variation of the problem, where we assume $\mu^\star$ is known to the controller and we wish to minimize the relaxed objective

$$J'_T(u) = \mathbb{E}\left[T^{-1} \cdot \int_0^T \|\theta_t^u - \mu^\star\|^2 \, dt + \lambda \cdot \int_0^T \|u_t\|^2 \, dt\right]. \tag{11}$$

While we actually want to minimize the $J_T(u)$ defined in (10), which we will do in the sequel, but observe that (i) $J_T(u) \leq J'_T(u)$ for all u by Jensen's inequality and (ii) minimizing $J'_T(u)$ leads to a more analytically tractable problem.

Recall that θ_t^u evolves according to the SDE (1),

$$d\theta_t^u = [\mathbf{A}\,(\mu^\star - \theta_t^u) + u_t]\,dt + \mathbf{\Sigma}^{1/2} dW_t.$$

We claim the following result.

Theorem 6. *Suppose that θ_t^u evolves according to (1) for some progressively measurable u_t. Then the optimal control u_t that minimizes the relaxed objective $J'_T(u)$ defined in (11) is given by*

$$u_t = \frac{\mathbf{P}_t\,(\mu^\star - \theta_t^u)}{\lambda}, \quad \textit{where} \quad \dot{\mathbf{P}}_t = \mathbf{A}^\top P_t + P_t\mathbf{A} + \lambda^{-1}P_t^2 - T^{-1}\mathbf{I}, \quad \textit{and} \quad P_T = 0. \tag{12}$$

In the special case that $\mathbf{A} = \operatorname{diag}(\alpha_1, \ldots, \alpha_d)$ is diagonal, we have that

$$u_i(t) = \frac{1}{\lambda T} \cdot \frac{1 - e^{-2\sqrt{\alpha_i^2 + 1/\lambda T}(T-t)}}{\left(\sqrt{\alpha_i^2 + 1/\lambda T} + \alpha_i\right) + \left(\sqrt{\alpha_i^2 + 1/\lambda T} - \alpha_i\right) e^{-2\sqrt{\alpha_i^2 + 1/\lambda T}(T-t)}} \left(\mu_i^\star - \theta_i(t)\right).$$

Proof. The proof proceeds by first centering the dynamics and then applying the standard Riccati equation given in Theorem 4 to solve for the optimal control. Indeed, let

$$x_t = \theta_t^u - \mu^\star, \quad \text{so} \quad dx_t = [-\mathbf{A}x_t + u_t]\,dt + \mathbf{\Sigma}^{1/2} dW_t. \tag{13}$$

We can then rewrite $J'_T(u)$ as

$$J'_T(u) = \mathbb{E}\left[\frac{1}{T}\int_0^T \|x_t\|^2 \, dt + \lambda \int_0^T \|u_t\|^2 \, dt\right].$$

Now, writing

$$\mathbf{Q} = T^{-1} \cdot \mathbf{I} \quad \text{and} \quad \mathbf{R} = \lambda \cdot \mathbf{I},$$

we can again rewrite $J'_T(u)$ as

$$J'_T(u) = \mathbb{E}\left[\int_0^T \left(x_t^\top \mathbf{Q} x_t + u_t^\top \mathbf{R} u_t\right) dt\right],$$

We now observe that combining (13) with the above expression for $J'_T(u)$, we have reduced the problem of minimizing $J'_T(u)$ to a standard linear-quadratic-Gaussian (LQG) control problem as in Theorem 4. Applying the Riccati equation from Theorem 4 yields the desired result, in particular the case found in Corollary 1.

For the special case, we use Lemma 1. Indeed, setting

$$r = \lambda, \quad q = T^{-1}, \quad q_f = 0, \quad \text{and} \quad a = -\alpha_i,$$

that result tells us that we can solve the scalar Riccati equation to get

$$p_i(t) = -\frac{\lambda\left(\sqrt{\alpha_i^2 + 1/\lambda T} - \alpha_i\right)\left(1 - e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}\right)}{1 + \frac{\sqrt{\alpha_i^2+1/\lambda T}-\alpha_i}{\sqrt{\alpha_i^2+1/\lambda T}+\alpha_i} e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}}.$$

Thus,

$$\begin{aligned}
u_i(t) &= -\lambda^{-1} p_i(t) x_t^{(i)} \\
&= \frac{\left(\sqrt{\alpha_i^2 + 1/\lambda T} - \alpha_i\right)\left(1 - e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}\right)}{1 + \frac{\sqrt{\alpha_i^2+1/\lambda T}-\alpha_i}{\sqrt{\alpha_i^2+1/\lambda T}+\alpha_i} e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}} (\mu_i^\star - \theta_t^{(i)}) \\
&= \frac{1}{\lambda T} \cdot \frac{1 - e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}}{\sqrt{\alpha_i^2 + 1/\lambda T} + \alpha_i + \left(\sqrt{\alpha_i^2 + 1/\lambda T} - \alpha_i\right) e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)}}.
\end{aligned}$$

The result follows. □

We now construct an approximation to the above optimal control where we assume that $t \ll T$. Indeed, we suppose that

$$T - t \gg \frac{1}{2\sqrt{\alpha_i^2 + 1/\lambda T}} \geq \frac{1}{2\alpha_i},$$

so that we can approximate $e^{-2\sqrt{\alpha_i^2+1/\lambda T}(T-t)} \approx 0$. Then we have that

$$u_i(t) \approx \frac{1}{\lambda T} \cdot \frac{1}{\sqrt{\alpha_i^2 + 1/\lambda T} + \alpha_i},$$

which is the form of the control used in our proposed algorithm.

E.2 Optimal Algorithm for the Original Objective

We now continue by considering the simplified setting where the controller is aware of the true minimizer $\mu^\star$, but where we wish to minimize the original objective $J_T(u)$ defined in (10) rather than the relaxed objective $J'_T(u)$ defined in (11).

Theorem 7. *Suppose that θ_t^u evolves according to (1) for some progressively measurable u_t. Then the optimal control u_t that minimizes the original objective $J_T(u)$ defined in (10) is given by*

$$u_t^\star = \lambda^{-1}\left(\mathbf{M}_t(\mu^\star - \theta_t) + \mathbf{N}_t \int_0^t (\mu^\star - \theta_s) ds\right),$$

where

$$\begin{aligned}
-\dot{\mathbf{M}}_t &= -\mathbf{A}^\top \mathbf{M}_t - \mathbf{M}_t \mathbf{A} + \mathbf{N}_t + \mathbf{N}_t^\top - \lambda^{-1}\mathbf{M}_t^2, \quad \mathbf{M}_T = 0, \\
-\dot{\mathbf{N}}_t &= -\mathbf{A}^\top \mathbf{N}_t + \mathbf{S}_t - \lambda^{-1}\mathbf{M}_t\mathbf{N}_t, \quad \mathbf{N}_T = 0, \\
-\dot{\mathbf{S}}_t &= -\lambda^{-1}\mathbf{N}_t^\top \mathbf{N}_t, \quad \mathbf{S}_T = T^{-2}\mathbf{I}.
\end{aligned}$$

Moreover, in the special case that $\mathbf{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_d)$ is diagonal, we have that for $\alpha_i \neq 0$,

$$\begin{aligned}
u_i^\star(t) = \frac{1 - e^{-\alpha_i(T-t)}}{\alpha_i \lambda T^2 + {}^{(T-t)}/_{\alpha_i} - \frac{2}{\alpha_i^2}\cdot\left(1 - e^{-\alpha_i(T-t)}\right) + \frac{1-e^{-2\alpha_i(T-t)}}{2\alpha_i^2}} \\
\times \left(\frac{1 - e^{-\alpha_i(T-t)}}{\alpha_i}(\mu^\star - \theta_t) + \int_0^t (\mu^\star - \theta_s)\, ds\right)
\end{aligned} \tag{14}$$

whereas for $\alpha_i = 0$, we have

$$u_i^\star(t) = \frac{(T-t)^2}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot (\mu^\star - \theta_t) + \frac{(T-t)}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot \int_0^t (\mu^\star - \theta_s)\, ds. \tag{15}$$

Proof. We introduce the accumulated centered process

$$y_t^u = \int_0^t (\theta_s^u - \mu^\star)\, ds.$$

Then we observe that

$$J_T(u) = \mathbb{E}\left[\frac{1}{T^2}\left\|y_T\right\|^2 + \lambda \int_0^T \left\|u_t\right\|^2 dt\right], \quad \text{and} \quad dy_t^u = x_t dt.$$

This is not quite in the form of an LQG problem, but suppose that we augment the state to be $z_t^u = (x_t^u, y_t^u)$. Then we have that

$$dz_t^u = (\mathbf{F} z_t + \mathbf{G} u_t)\, dt + \mathbf{E} dW_t, \quad \text{where} \quad \mathbf{F} = \begin{bmatrix} -\mathbf{A} & 0 \\ \mathbf{I} & 0 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} \mathbf{\Sigma}^{1/2} \\ 0 \end{bmatrix}.$$

We then may take $\mathbf{Q}$ to be zero and

$$\mathbf{Q}_f = \begin{bmatrix} 0 & 0 \\ 0 & T^{-2}\mathbf{I} \end{bmatrix}, \quad \mathbf{R} = \lambda \mathbf{I}.$$

This then again becomes an LQG problem, and we can apply the Riccati equation from Theorem 4 to solve for the optimal control. Indeed, the Riccati equation in this case is given by

$$-\dot{\mathbf{P}}_t = \mathbf{F}^\top \mathbf{P}_t + \mathbf{P}_t \mathbf{F} - \lambda^{-1} \mathbf{P}_t \mathbf{G} \mathbf{G}^\top \mathbf{P}_t, \quad \text{and} \quad \mathbf{P}_T = \mathbf{Q}_f.$$

Writing

$$\mathbf{P}_t = \begin{bmatrix} \mathbf{M}_t & \mathbf{N}_t \\ \mathbf{N}_t^\top & \mathbf{S}_t \end{bmatrix}, \tag{16}$$

we get that

$$u_t^\star = -\lambda^{-1} \left(\mathbf{M}_t x_t + \mathbf{N}_t y_t\right)$$

or, alternatively

$$u_t^\star = -\lambda^{-1} \left(\mathbf{M}_t(\theta_t - \mu^\star) + \mathbf{N}_t \int_0^t (\theta_s - \mu^\star)\, ds\right), \tag{17}$$

where

$$\begin{aligned} -\dot{\mathbf{M}}_t &= -\mathbf{A}^\top \mathbf{M}_t - \mathbf{M}_t \mathbf{A} + \mathbf{N}_t + \mathbf{N}_t^\top - \lambda^{-1} \mathbf{M}_t^2, \quad \mathbf{M}_T = 0, \\ -\dot{\mathbf{N}}_t &= -\mathbf{A}^\top \mathbf{N}_t + \mathbf{S}_t - \lambda^{-1} \mathbf{M}_t \mathbf{N}_t, \quad \mathbf{N}_T = 0, \\ -\dot{\mathbf{S}}_t &= -\lambda^{-1} \mathbf{N}_t^\top \mathbf{N}_t, \quad \mathbf{S}_T = T^{-2}\mathbf{I} \end{aligned} \tag{18}$$

is derived by plugging in the decomposition in (16) into the Riccati equation in (9) and rearranging. The first result follows.

For the special case that $\mathbf{A}$ is diagonal, the coordinates decouple and we derive the solution in Lemma 3. □

Lemma 3. *Suppose that $\mathbf{A} = \mathrm{diag}(\alpha_1, \ldots, \alpha_d)$ is diagonal. Then the above system of ODEs for $\mathbf{M}_t$, $\mathbf{N}_t$, and $\mathbf{S}_t$ decouples into d independent systems of ODEs, one for each coordinate. In particular, for each $i \in [d]$, we have that (18) decouples and can be solved to yield a closed-form solution for $u_i^\star(t)$, given in (14) and (15).*

Proof. In the case where $\mathbf{A}$ is diagonal, it is immediate that the system decouples into d independent systems of ODEs, one for each coordinate because now every matrix is diagonal (note that $\mathbf{\Sigma}$ may be taken to be diagonal without loss of generality by the rotational invariance of Brownian motion). Thus, we can solve for each coordinate separately.

In particular, we know that the optimal controller is given by (17), i.e. for the ith coordinate,

$$u_i^\star(t) = \lambda^{-1} \left(M_t(\mu_i^\star - \theta_t^{(i)}) + N_t \int_0^t (\mu_i^\star - \theta_s^{(i)}) ds\right),$$

where M_t and N_t solve the system of ODEs from (18) in the scalar setting, i.e.

$$\begin{aligned} -\dot{M}_t &= -2\alpha_i M_t + 2N_t - \lambda^{-1} M_t^2, \quad M_T = 0, \\ -\dot{N}_t &= -\alpha_i N_t + S_t - \lambda^{-1} M_t N_t, \quad N_T = 0, \\ -\dot{S}_t &= -\lambda^{-1} N_t^2, \quad S_T = T^{-2}. \end{aligned} \tag{19}$$

Note that we are now in the scalar setting, so we are not bolding the variables; for notational simplicity, we omit the subscript i in the remainder of this proof. Now, let $\tau = T - t$ be the time to go and define the following functions:

$$h(\tau) = \int_0^\tau e^{-\alpha s} ds = \begin{cases} \frac{1-e^{-\alpha\tau}}{\alpha}, & \alpha \neq 0, \\ \tau, & \alpha = 0, \end{cases}$$

and

$$g(\tau) = \int_0^\tau h(s)^2 ds = \begin{cases} \frac{\tau}{\alpha^2} - \frac{2(1-e^{-\alpha\tau})}{2\alpha^3} + \frac{1-e^{-2\alpha\tau}}{2\alpha^3}, & \alpha \neq 0, \\ \frac{\tau^3}{3}, & \alpha = 0. \end{cases}$$

We claim that the solution to the above system of ODEs is given by

$$M_t = \frac{h(\tau)^2}{T^2 + {}^{g(\tau)}/_{\lambda}}, \quad N_t = \frac{h(\tau)}{T^2 + {}^{g(\tau)}/_{\lambda}}, \quad S_t = \frac{1}{T^2 + {}^{g(\tau)}/_{\lambda}}. \tag{20}$$

Let $D(\tau) = T^2 + {}^{g(\tau)}/_{\lambda}$. We verify that the above is indeed a solution by plugging into the ODEs. First, we have that

$$h'(\tau) = e^{-\alpha\tau} = 1 - \alpha h(\tau), \quad g'(\tau) = h(\tau)^2, \quad \text{and} \quad D'(\tau) = \lambda^{-1} h(\tau)^2.$$

Observing that ${}^{d}/_{d\tau} = -{}^{d}/_{dt}$, can plug in (20) into (19) to observe that this is indeed the solution.

Thus, plugging back in, we see that the optimal control is given by

$$u^\star(t) = \frac{h(T-t)}{\lambda T^2 + g(T-t)} \cdot \left(h(T-t)(\mu^\star - \theta_t) + \int_0^t (\mu^\star - \theta_s)\, ds \right).$$

In particular, for $\alpha = 0$, we have that

$$u^\star(t) = \frac{(T-t)^2}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot (\mu^\star - \theta_t) + \frac{(T-t)}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot \int_0^t (\mu^\star - \theta_s)\, ds.$$

For $\alpha \neq 0$, we have that

$$\begin{aligned} u^\star(t) = &\frac{1 - e^{-\alpha(T-t)}}{\alpha\lambda T + {}^{(T-t)}/_{\alpha} - \frac{2}{\alpha^2} \cdot \left(1 - e^{-\alpha(T-t)}\right) + \frac{1-e^{-2\alpha_i(T-t)}}{2\alpha_i^2}} \\ &\times \left(\frac{1 - e^{-\alpha(T-t)}}{\alpha} (\mu^\star - \theta_t) + \int_0^t (\mu^\star - \theta_s)\, ds \right). \end{aligned}$$

The result follows. □

E.3 Generalizing to the Case Where $\mu^\star$ is Unknown

The above results are a promising start but all rely on the assumption that $\mu^\star$ is known to the controller, which is certainly not the case in practice. Indeed, the entire goal of optimization is to *find* $\mu^\star$; thus, if the learner had access to $\mu^\star$ from the start there would be no need to run any optimization algorithm at all. In this section we show how to modify the above optimal controllers to the case where $\mu^\star$ is unknown, and in particular we show that the same form of controller is still optimal, but with $\mu^\star$ replaced by an estimate of $\mu^\star$.

The key factor in generalizing to the case where $\mu^\star$ is unknown is the *separation principle* from control theory, which is also known as *certainty equivalence* [12, 20, 36]. This principle essentially states that for an LQG control problem where the state is not directly observed, the optimal control can be obtained by first estimating the state as the conditional expectation of the state given the observations and then plugging this estimate into the optimal control for the fully observed case. More formally, we have the following result.

Theorem 8. *Consider an LQG problem as in (7) and (8) where the state x_t^u is not directly observed, but rather we have access to an observation process $y_t = \phi(x_t^u, \mu^\star)$ for some unknown $\mu^\star$ drawn from a known prior distribution with ϕ invertible in the first argument. Suppose that the corresponding full-information LQG problem where the state is directly observed has an optimal control given by $u_t = \mathbf{K}_t x_t$ for some matrix $\mathbf{K}_t$. Then the optimal control for the partially observed problem is given by $u_t = \mathbf{K}_t \hat{x}_t$, where $\hat{x}_t = \mathbb{E}\left[\phi^{-1}(y_t, \mu^\star) \mid y_s, s \le t\right]$ is the conditional expectation of the state given the observations.*

Proof. This is a classical result in control theory, and we refer the reader to Georgiou and Lindquist [12], Kwakernaak and Sivan [20], Van de Water and Willems [36] for a complete proof. In order to translate those works to our setting, we may apply Georgiou and Lindquist [12, Theorem 14] to the original control problem augmented with the state μ_t such that $\mu_0 = \mu^\star$ and $d\mu_t = 0$ for all t; the observation process $\theta_t = \mu^\star + x_t$ is then a linear function of the state, and the optimal control for the fully observed problem is given by $u_t = \mathbf{K}_t x_t$ where $\mathbf{K}_t$ is the solution to the Riccati equation as in Theorem 4. Applying Georgiou and Lindquist [12, Theorem 14] then yields the desired result. □

We thus have the following two corollaries, for the relaxed and original objectives respectively.

Corollary 3. *Let $J_T'(u)$ denote the relaxed objective in (11) and suppose that θ_t^u evolves according to (1) for some progressively measurable u_t. Let $\widehat{\mu}_t = \mathbb{E}\left[\mu^\star \mid \theta_s^u, s \le t\right]$ be the conditional expectation of $\mu^\star$ given the trajectory of θ_t^u up to time t. Then the optimal control u_t that minimizes $J_T'(u)$ is given by*

$$u^\star(t) = \frac{\mathbf{P}_t\left(\widehat{\mu}_t - \theta_t^u\right)}{\lambda}$$

where $\mathbf{P}_t$ is the solution to the same Riccati equation as in (12). In the special case that $\mathbf{A} = \operatorname{diag}(\alpha_1, \ldots, \alpha_d)$ is diagonal, we have that

$$u_i^\star(t) = \frac{1}{\lambda T} \cdot \frac{1 - e^{-2\sqrt{\alpha_i^2 + 1/\lambda T}(T-t)}}{\left(\sqrt{\alpha_i^2 + 1/\lambda T} + \alpha_i\right) + \left(\sqrt{\alpha_i^2 + 1/\lambda T} - \alpha_i\right) e^{-2\sqrt{\alpha_i^2 + 1/\lambda T}(T-t)}} \left(\widehat{\mu}_i(t) - \theta_i(t)\right)$$

for each $i \in [d]$.

Proof. This follows immediately by combining Theorem 6 with Theorem 8. □

For the original objective, we have the following result.

Corollary 4. *Let $J_T(u)$ denote the original objective in (10) and suppose that θ_t^u evolves according to (1) for some progressively measurable u_t. Let $\widehat{\mu}_t = \mathbb{E}\left[\mu^\star \mid \theta_s^u, s \le t\right]$ be the conditional expectation of $\mu^\star$ given the trajectory of θ_t^u up to time t. Then the optimal control u_t that minimizes $J_T(u)$ is given by*

$$u_t^\star = \lambda^{-1}\left(\mathbf{M}_t(\widehat{\mu}_t - \theta_t) + \mathbf{N}_t\left(t \cdot \widehat{\mu}_t - \int_0^t \theta_s ds\right)\right),$$

where $\mathbf{M}_t$, $\mathbf{N}_t$, and $\mathbf{S}_t$ are the solutions to the same system of ODEs as in Theorem 7. In the special case that $\mathbf{A} = \operatorname{diag}(\alpha_1, \ldots, \alpha_d)$ is diagonal, we have that for $\alpha_i \neq 0$,

$$u_i^\star(t) = \frac{1 - e^{-\alpha_i(T-t)}}{\alpha_i \lambda T^2 + {}^{(T-t)}/_{\alpha_i} - \frac{3}{2\alpha_i^2} \cdot \left(1 - e^{-\alpha_i(T-t)}\right)} \times \left(\frac{1 - e^{-\alpha_i(T-t)}}{\alpha_i}\left(\widehat{\mu}_i(t) - \theta_t^{(i)}\right) + \left(t \cdot \widehat{\mu}_i(t) - \int_0^t \theta_s^{(i)} ds\right)\right)$$

whereas for $\alpha_i = 0$, we have

$$u_i^\star(t) = \frac{(T-t)^2}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot \left(\widehat{\mu}_i(t) - \theta_t^{(i)}\right) + \frac{(T-t)}{\lambda T^2 + {}^{(T-t)^3}/_3} \cdot \left(t \cdot \widehat{\mu}_i(t) - \int_0^t \theta_s^{(i)} ds\right).$$

Proof. This follows immediately by combining Theorem 7 with Theorem 8. □

Thus, at least in theory, we have a complete answer to the question of how to control optimization algorithms in order to minimize the mean squared error of the returned iterate average as an estimator of $\mu^\star$. Unfortunately, the above optimal controllers are not implementable in practice because they require computing $\widehat{\mu}_t = \mathbb{E}\left[\mu^\star \mid \theta_s^u, s \le t\right]$, even in the simplified diagonal setting where the coordinates decouple. In the following subsection, we show how to derive a practical algorithm by approximating $\widehat{\mu}_t$ with a simple estimator of $\mu^\star$ that can be computed from the trajectory of θ_t^u up to time t.

E.4 Estimating the optimum under controlled dynamics

We now derive an explicit expression for the posterior mean in the quadratic model and relate it to the running trajectory average. Recall the controlled dynamics

$$d\theta_t = \left[A(\mu^\star - \theta_t) + u_t\right] dt + \Sigma^{1/2} dW_t,$$

where A is known and nonsingular, Σ is positive definite, and u_t is a known admissible control adapted to the observations. For this calculation, take θ_0 to be deterministic and the prior on $\mu^\star$ to be independent of the Brownian motion. Define the control-corrected observation

$$Y_t = \theta_t - \theta_0 + \int_0^t A\theta_s \, ds - \int_0^t u_s \, ds.$$

Integrating the dynamics yields

$$Y_t = tA\mu^\star + \Sigma^{1/2} W_t.$$

Conditional on $\mu^\star$, the controlled trajectory provides the same information about $\mu^\star$ as the Gaussian observation Y_t. The posterior is formed under the controlled dynamics; no mixing under the original, uncontrolled dynamics is required.

For a Gaussian prior $\mu^\star \sim \mathcal{N}(m_0, V_0)$, the posterior mean is

$$\hat{\mu}_t = \left(V_0^{-1} + tA^\top \Sigma^{-1} A\right)^{-1} \left(V_0^{-1} m_0 + A^\top \Sigma^{-1} Y_t\right).$$

For $t > 0$, taking the diffuse-prior limit gives

$$\begin{aligned}
\hat{\mu}_t &= \frac{1}{t} A^{-1} Y_t \\
&= \frac{1}{t} \int_0^t \theta_s \, ds + \frac{1}{t} A^{-1}(\theta_t - \theta_0) - \frac{1}{t} A^{-1} \int_0^t u_s \, ds.
\end{aligned}$$

Thus, under the controlled dynamics, the time average appears in the posterior mean alongside a boundary correction and a correction for cumulative control. Computing this more exact estimator at LM scale would require an estimate of the local curvature matrix and its inverse, as well as the initial point and the cumulative control. The quadratic model is a local approximation, so its curvature and relevant optimum may change during training. For these reasons, we use the running average as a practical approximation to the posterior mean in PACE.

E.5 Deriving a Practical Algorithm through Approximation

As remarked above, the optimal controllers derived in the previous subsections are not implementable in practice. For the general setting, solving the Riccati equations is already highly nontrivial in high dimensions and certainly beyond reach at the scale of language models. Even in the diagonal setting, the optimal controllers require computing $\widehat{\mu}_t = \mathbb{E}\left[\mu^\star \mid \theta_s^u, s \le t\right]$, which is not computable in closed form and would require solving a high-dimensional integral at every time step. Moreover, the updates themselves are quite complicated. We thus propose three simplifications:

1. We will consider only the diagonal setting, where the coordinates decouple and we can solve for the optimal controller in closed form.
2. We will approximate $\widehat{\mu}_t = \frac{1}{t}\int_0^t \theta_s^u ds$, which is a natural estimator of $\mu^\star$ given the trajectory of θ_t^u up to time t.
3. We will approximate the optimal controller by its form when $t \ll T$, which is the regime we are in for most of the optimization trajectory.

The first approximation allows for the closed-form solution to the optimal controller, which we derived above. The second approximation is a natural one, as $\widehat{\mu}_t$ becomes simple to estimate and is naturally aligned with the core motivation of this work: using iterate averaging to get better estimates of $\mu^\star$. The third approximation is also natural and can be used to greatly simplify the form of the optimal controller. In particular we will make the following approximations:

$$T - t \approx T \quad \text{and} \quad e^{-\alpha_i(T-t)} \approx 0,$$

where the second of these relies on the assumption that $T - t \gg \frac{1}{2\sqrt{\alpha_i^2 + 1/\lambda T}}$. Critically, by replacing $\widehat{\mu}_t$ with the iterate average, the second term in Corollary 4 drops out, which is a significant simplification and makes the resulting controller much more practical to implement.

With these approximations, we have that the optimal controller in Corollary 4 is approximated by

$$u_i^\star(t) = \frac{1}{T\left(1 + \lambda\alpha_i^2 T\right)}\left(\widehat{\mu}_i(t) - \theta_i(t)\right),$$

which is a simple and practical controller that can be empirically implemented. In particular, we can write the above in vector form as

$$u^\star(t) = \frac{1}{T} \cdot \left(\mathbf{I} + \lambda \mathbf{A}^2 T\right)^{-1}\left(\widehat{\mu}(t) - \theta(t)\right).$$

This is the form of the controller used in our proposed algorithm, albeit with a discrete time approximation and additional simplifications.

F Proofs of Convergence Results

In this section we provide rigorous proofs of the convergence results stated in Section 2. We begin with a proof of Theorem 3 in Section F.1, before proceeding to prove our two propositions about the improvement over SGD in the quadratic setting in Sections F.2 and F.3.

F.1 Proof of Theorem 3

We now prove that the update strategy given in (5) converges to the minimum of F when F is convex. In particular, we instantiate the update with a diagonal $\mathbf{C}$ and a fixed EMA parameter β. More precisely, we suppose

$$\mathbf{C} = \operatorname{diag}(c_1, \ldots, c_d), \quad \text{where} \quad 0 \le c_i \le 1, \quad \text{and} \quad \beta \in (0, 1). \tag{21}$$

We denote by $c_{\min} = \min_{1 \le i \le d} c_i$ the smallest diagonal entry of $\mathbf{C}$. We will further make the following assumptions about the objective F and the stochastic gradients g_k:

Assumption F.1. The objective $F : \mathbb{R}^d \to \mathbb{R}$ is convex and G-Lipschitz with minimum $\mu^\star$.

Assumption F.2. At any iteration k, the gradient estimate $g_k \in \mathbb{R}^d$ satisfies $\mathbb{E}_{k-1}[g_k] = \nabla F(\theta_k)$ and $\mathbb{E}_{k-1}\left[\|g_k - \nabla F(\theta_k)\|^2\right] \le \sigma^2$ for some $\sigma > 0$, where $\mathbb{E}_k[\cdot]$ denotes expectation conditional on the filtration generated by θ_j, g_j for $j \le k$. We denote $V^2 = G^2 + \sigma^2$.

Recall that given parameter θ_k, EMA parameter $\widehat{\theta}_{\mathsf{EMA},k}$, and stochastic gradient g_k, for learning rate η_k we update

$$\theta_{k+1} = \theta_k - \eta_k \cdot g_k + \mathbf{C}\left(\widehat{\theta}_{\mathsf{EMA},k} - \theta_k\right) \quad \text{and} \quad \widehat{\theta}_{\mathsf{EMA},k+1} = (1-\beta)\cdot\widehat{\theta}_{\mathsf{EMA},k} + \beta \cdot \theta_{k+1}. \tag{22}$$

We now prove a more general result for arbitrary non-increasing learning rate schedules from which Theorem 3 will follow.

Theorem 9. *Suppose that Assumptions F.1 and F.2 hold and that θ_k is updated as in (22) for some non-increasing sequence $\{\eta_k\}_{k=0}^{\infty}$. Furthermore, suppose that $\mathbf{C}$ and β are as in (21). Let*

$$\overline{\theta}_T = \frac{\sum_{k=1}^{T} \eta_k \cdot \theta_k}{\sum_{k=1}^{T} \eta_k}$$

denote the average. Let

$$D_{\mathbf{C}}^2 = \cdot (\theta_1 - \mu^\star)^\top \left(\mathbf{I} + {}^{1-\beta}/_{\beta} \cdot \mathbf{C}\right)(\theta_1 - \mu^\star) \quad \textit{and} \quad K_{\mathbf{C}} = \max_{1 \le i \le d} \frac{c_i}{\beta + (1-\beta)c_i}. \tag{23}$$

If V is as in Assumption F.2, then

$$\mathbb{E}\left[F(\overline{\theta}_T) - F(\mu^\star)\right] \le \frac{D_{\mathbf{C}}^2 + \left(V^2 + \frac{2G(1-\beta)K_{\mathbf{C}}V}{\beta + (1-\beta)c_{\min}}\right) \cdot \sum_{k=1}^{T} \eta_k^2}{2 \cdot \sum_{k=1}^{T} \eta_k}.$$

Proof. We introduce the notation

$$r_k = \theta_k - \widehat{\theta}_{\mathsf{EMA},k}.$$

From (22), we have

$$\theta_{k+1} = \widehat{\theta}_{\mathsf{EMA},k} + (\mathbf{I} - \mathbf{C})\, r_k - \eta_k \cdot g_k.$$

On the other hand, by the definition of the EMA update, we have

$$\widehat{\theta}_{\mathsf{EMA},k+1} = \widehat{\theta}_{\mathsf{EMA},k} + \beta\left((\mathbf{I} - \mathbf{C})r_k - \eta_k \cdot g_k\right).$$

Combining the preceding two displays, we have

$$r_{k+1} = (1-\beta)\left((\mathbf{I} - \mathbf{C})\, r_k - \eta_k \cdot g_k\right). \tag{24}$$

We now introduce the notation[3]

$$\mathbf{B} = \beta \cdot \mathbf{I} + (1-\beta) \cdot \mathbf{C}.$$

Now, letting

$$x_k = \widehat{\theta}_{\mathsf{EMA},k} + \beta \mathbf{B}^{-1}(\mathbf{I} - \mathbf{C})r_k, \tag{25}$$

we claim that

$$x_{k+1} = x_k - \beta\eta_k \cdot \mathbf{B}^{-1} g_k.$$

To see this, note that

$$\begin{aligned}
x_{k+1} &= \widehat{\theta}_{\mathsf{EMA},k+1} + \beta \mathbf{B}^{-1}(\mathbf{I} - \mathbf{C})r_{k+1} \\
&= \widehat{\theta}_{\mathsf{EMA},k} + \left(\beta \cdot \mathbf{I} + (1-\beta)\beta \mathbf{B}^{-1}(\mathbf{I} - \mathbf{C})\right)\left((\mathbf{I} - \mathbf{C})r_k - \eta_k \cdot g_k\right) \\
&= \widehat{\theta}_{\mathsf{EMA},k} + \beta \cdot \mathbf{B}^{-1}(\mathbf{I} - \mathbf{C})r_k - \beta\eta_k \cdot \mathbf{B}^{-1} g_k \\
&= x_k - \beta\eta_k \cdot \mathbf{B}^{-1} g_k.
\end{aligned}$$

We now claim that

$$\mathbb{E}_k\left[g_k^\top \mathbf{B}^{-1} g_k\right] \le \beta^{-1} \cdot V^2. \tag{26}$$

To see this, note first that because $\mathbf{C}$ is diagonal, it holds that $\mathbf{B}^{-1}$ is diagonal with entries

$$0 \le b_i = \frac{1}{\beta + (1-\beta)c_i} \le \beta^{-1}.$$

Thus it holds that

$$g_k^\top \mathbf{B}^{-1} g_k \le \beta^{-1} \left\|g_k\right\|^2.$$

[3]This will never be confused with the $\mathbf{B}$ in the definition of LQG control problems in (7), as this notation is only used in this section.

Thus, (26) then follows by Assumption F.2.

We now use the convexity of F to bound control the instantatneous suboptimality of θ_k by a function of x_k. First, note that

$$\mathbf{I} - \beta\mathbf{B}^{-1}(\mathbf{I} - \mathbf{C}) = (\beta\mathbf{I} + (1-\beta)\mathbf{C})^{-1}\,\mathbf{C} = \mathbf{B}^{-1}\mathbf{C}.$$

Thus,

$$\left\|\mathbf{I} - \beta\mathbf{B}^{-1}(\mathbf{I} - \mathbf{C})\right\|_{\text{op}} = \left\|\mathbf{B}^{-1}\mathbf{C}\right\|_{\text{op}} = K_{\mathbf{C}}. \tag{27}$$

Now, using (25), we see that

$$\begin{aligned}
\langle \nabla F(\theta_k), x_k - \mu^\star\rangle &= \langle \nabla F(\theta_k), \theta_k - \mu^\star\rangle - \left\langle \nabla F(\theta_k), \left(\mathbf{I} - \beta\mathbf{B}^{-1}(\mathbf{I}-\mathbf{C})r_k\right)\right\rangle \\
&\geq F(\theta_k) - F(\mu^\star) - \left\langle \nabla F(\theta_k), \left(\mathbf{I} - \beta\mathbf{B}^{-1}(\mathbf{I}-\mathbf{C})r_k\right)\right\rangle \\
&\geq F(\theta_k) - F(\mu^\star) - GK_{\mathbf{C}}\left\|r_k\right\|,
\end{aligned}$$

where the first inequality follows from convexity of F and the second inequality follows from Cauchy-Schwarz coupled with the G-Lipschitzness of F and (27).

Now, expanding the square, we see that

$$\begin{aligned}
&\mathbb{E}_k\left[\beta^{-1}\left(x_{k+1} - \mu^\star\right)^\top \mathbf{B}\left(x_{k+1} - \mu^\star\right)\right] \\
&\qquad = \mathbb{E}_k\left[\beta^{-1}\left(x_k - \mu^\star - \beta\eta_k\cdot\mathbf{B}^{-1}g_k\right)^\top \mathbf{B}\left(x_k - \mu^\star - \beta\eta_k\cdot\mathbf{B}^{-1}g_k\right)\right] \\
&\qquad = \beta^{-1}\left(x_k - \mu^\star\right)^\top \mathbf{B}\left(x_k - \mu^\star\right) \\
&\qquad\quad - 2\eta_k\cdot\langle\nabla F(\theta_k), x_k - \mu^\star\rangle + \beta\eta_k^2\cdot\mathbb{E}_k\left[g_k^\top\mathbf{B}^{-1}g_k\right] \\
&\qquad \leq \beta^{-1}\left(x_k - \mu^\star\right)^\top \mathbf{B}\left(x_k - \mu^\star\right) \\
&\qquad\quad - 2\eta_k\cdot\langle\nabla F(\theta_k), x_k - \mu^\star\rangle + \beta\eta_k^2\cdot V^2.
\end{aligned}$$

Plugging this into the preceding display and rearranging, we see that

$$\begin{aligned}
2\eta_k\cdot\mathbb{E}\left[F(\theta_k) - F(\mu^\star)\right] \leq\ &\mathbb{E}\left[\beta^{-1}\left(x_k - \mu^\star\right)^\top \mathbf{B}\left(x_k - \mu^\star\right)\right] \\
&- \mathbb{E}\left[\beta^{-1}\left(x_{k+1} - \mu^\star\right)^\top \mathbf{B}\left(x_{k+1} - \mu^\star\right)\right] \\
&+ 2\eta_k GK_{\mathbf{C}}\cdot\mathbb{E}\left[\|r_k\|\right] + V^2\cdot\eta_k^2.
\end{aligned}$$

Now, summing over k and telescoping, we see that

$$\begin{aligned}
2\cdot\sum_{k=1}^T \eta_k\cdot\mathbb{E}\left[F(\theta_k) - F(\mu^\star)\right] \leq\ &\beta^{-1}\cdot\mathbb{E}\left[\left(x_1 - \mu^\star\right)^\top \mathbf{B}\left(x_1 - \mu^\star\right)\right] \\
&+ V^2\cdot\sum_{k=1}^T\eta_k^2 + 2GK_{\mathbf{C}}\cdot\sum_{k=1}^T\eta_k\cdot\mathbb{E}\left[\|r_k\|\right].
\end{aligned}$$

Because $\theta_1 = \widehat{\theta}_{\mathsf{EMA},1}$ we know that $r_1 = 0$ and thus $x_1 = \theta_1$. Thus the first term on the right hand side can be bounded by $D_{\mathbf{C}}^2$ by the definition of $D_{\mathbf{C}}^2$ in (23). For the left hand side, observe that by convexity of F we have that

$$2\cdot\sum_{k=1}^T \eta_k\cdot\mathbb{E}\left[F(\theta_k) - F(\mu^\star)\right] \geq \left(2\cdot\sum_{k=1}^T\eta_k\right)\cdot\mathbb{E}\left[F(\overline{\theta}_T) - F(\mu^\star)\right].$$

Thus, all that remains is to bound the final term on the right hand side. Applying the convexity of the norm to (24), we have that

$$\|r_{k+1}\| \leq (1-\beta)\cdot\|\mathbf{I} - \mathbf{C}\|_{\text{op}}\cdot\|r_k\| + (1-\beta)\cdot\eta_k\cdot\|g_k\|.$$

By the construction of $\mathbf{C}$, we know that $\|\mathbf{I} - \mathbf{C}\|_{\text{op}} = 1 - c_{\min}$. Thus,

$$\mathbb{E}\left[\|r_{k+1}\|\right] \leq (1-\beta)(1-c_{\min})\cdot\mathbb{E}\left[\|r_k\|\right] + (1-\beta)\cdot\eta_k\cdot V.$$

By induction, it thus holds that

$$\mathbb{E}\left\{\|r_k\|\right\} \leq (1-\beta)V \sum_{j=0}^{k-1} \left((1-\beta)(1-c_{\min})\right)^j \cdot \eta_{k-1-j}.$$

Summing, and using the fact that η_k is non-increasing, we then have that

$$\begin{aligned}\sum_{k=1}^{T} \eta_k \cdot \mathbb{E}\left[\|r_k\|\right] &\leq (1-\beta)V \cdot \sum_{k=1}^{T-1} \sum_{j=0}^{k-2} \eta_{k-1-j}^2 \left((1-\beta)(1-c_{\min})\right)^j \\ &\leq \frac{(1-\beta)V}{1-(1-\beta)(1-c_{\min})} \cdot \sum_{k=1}^{T} \eta_k^2.\end{aligned}$$

The result concludes by plugging this into the preceding display and rearranging. □

We now prove Theorem 3 as a corollary of Theorem 9.

We do this by stating a more explicit convergence guarantee for the special case of a fixed learning rate.

Corollary 5. *Suppose that Assumptions F.1 and F.2 hold and θ_k is updated as in (22) for $\eta_k = T^{-1/2} \cdot \eta$ for some fixed η. Then,*

$$\overline{\theta}_T = \frac{1}{T} \sum_{k=1}^{T} \theta_k$$

and

$$\mathbb{E}\left[F(\overline{\theta}_T) - F(\mu^\star)\right] \leq \frac{D_{\mathbf{C}}^2 + \eta^2 \left(V^2 + \frac{2G(1-\beta)K_{\mathbf{C}}V}{\beta+(1-\beta)c_{\min}}\right)}{2\eta\sqrt{T}}.$$

In particular, for the optimal choice of η,

$$\mathbb{E}\left[F(\overline{\theta}_T) - F(\mu^\star)\right] \leq \frac{D_{\mathbf{C}}\sqrt{V^2 + \frac{2G(1-\beta)K_{\mathbf{C}}V}{\beta+(1-\beta)c_{\min}}}}{\sqrt{T}} \leq \frac{\sqrt{2-\beta}}{\beta} \cdot \frac{\|\theta_1 - \mu^\star\| V}{\sqrt{T}}$$

Proof. The first statement on $\overline{\theta}_T$ is immediate from the definition. The final inequaltiy for the tuned bound follows from the fact that

$$D_{\mathbf{C}} \leq \frac{\|\theta_1 - \mu^\star\|}{\sqrt{\beta}}, \quad K_{\mathbf{C}} \leq 1, \quad \text{and} \quad \beta + (1-\beta)c_{\min} \geq \beta.$$

The main bound for arbitrary fixed η then follows by plugging the fixed learning rate schedule into Theorem 9. □

Note that Theorem 3 follows immediately from Corollary 5.

F.2 Proof of Proposition 2

We now consider the special case of a diagonal quadratic objective, where we can obtain a more explicit convergence guarantee. Suppose that

$$F(\theta) = \frac{1}{2}(\theta - \mu^\star)^\top \mathbf{A}(\theta - \mu^\star), \quad \text{where} \quad \mathbf{A} = \operatorname{diag}(\alpha_1, \ldots, \alpha_d) \quad \text{with} \quad \alpha_i > 0. \tag{28}$$

Suppose that at each iteration we have access to a stochastic gradient of the form

$$g_k = \mathbf{A}(\theta_k - \mu^\star) + \xi_k, \quad \text{where} \quad \mathbb{E}\left[\xi_k\right] = 0 \quad \text{and} \quad \operatorname{Cov}\left(\xi_k\right) = \operatorname{diag}\left(\sigma_1^2, \ldots, \sigma_d^2\right). \tag{29}$$

We will consider the update in (22) for an arbitrary diagonal $\mathbf{C}$ with nonnegative entries and a fixed EMA parameter β and learning rate η. We make the following assumption on the learning rate, which is standard in the analysis of stochastic optimization algorithms for quadratic objectives [43] and ensures that the algorithm is stable.

Assumption F.3. We assume that the learning rate η satisfies $0 < \eta\alpha_i < 1$ for all $1 \le i \le d$.

Note that in the special case the $\mathbf{C} = 0$, we recover the dynamics of an EMA on SGD precisely.

Proposition 5. *Suppose that F is as in (28) and gradient estimates g_k are as in (29). Let $\widehat{\theta}_T$ denote the update at time T given by (22) for fixed η satisfying Assumption F.3. Let $\widehat{\theta}^{\mathrm{SGD}}$ evolve such that $\widehat{\theta}^{\mathrm{SGD}}_{k+1} = \widehat{\theta}^{\mathrm{SGD}}_k - \eta \cdot g_k$ and let $\widehat{\theta}_{\mathsf{EMA},k+1} = (1-\beta)\widehat{\theta}_{\mathsf{EMA},k} + \beta\theta_k$. Then there exists some $\mathbf{C}$ such that*

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right] < \lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\right\|^2\right] < \lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}^{\mathrm{SGD}}_T - \mu^\star\right\|^2\right].$$

Thus, in particular,

$$\lim_{T\to\infty} \mathbb{E}\left[F(\widehat{\theta}_T)\right] < \lim_{T\to\infty} \mathbb{E}\left[F(\widehat{\theta}_{\mathsf{EMA},T})\right] < \lim_{T\to\infty} \mathbb{E}\left[F(\widehat{\theta}^{\mathrm{SGD}}_T)\right].$$

Proof. Note first that we may set $\mu^\star = 0$ without loss of generality; if $\mu^\star \neq 0$, we can simply shift the objective by $\mu^\star$ to arrive at the same setting and the same convergence guarantee. Second, note that the ultimate statement follows immediately from the penultimate statement by the definition of F in (28). Third, note that due to the diagonal structure of the objective and $\mathbf{C}$, the dynamics of each coordinate are decoupled, and thus it suffices to prove the result for the one-dimensional case. Thus, we will now assume that $d = 1$ and write $\alpha = \alpha_1$, $\sigma^2 = \sigma_1^2$, and $c = c_1$. For the sake of notational convenience, let $a = \eta \cdot \alpha$. The update then becomes

$$\theta_{k+1} = (1 - a - c)\theta_k + c\widehat{\theta}_k 0 - \eta\xi_k \quad \text{and} \quad \widehat{\theta}_{k+1} = (1-\beta)\widehat{\theta}_k + \beta\theta_{k+1}. \tag{30}$$

We may write θ_t as a function of $\widehat{\theta}$ through the second equation in (30) and, plugging this into the first equation and rearranging, we get

$$\widehat{\theta}_{k+1} = (2 - a - \beta - c(1-\beta))\,\widehat{\theta}_k - (1-\beta)\,(1 - a - c)\widehat{\theta}_{k-1} - \beta\eta\xi_k. \tag{31}$$

We may rewrite (31) as an explicity AR(2) process with noise by observing that

$$\begin{bmatrix}\widehat{\theta}_{k+1}\\ \widehat{\theta}_k\end{bmatrix} = \begin{bmatrix}2 - a - \beta - c(1-\beta) & -(1-\beta)(1-a-c)\\ 1 & 0\end{bmatrix}\begin{bmatrix}\widehat{\theta}_k\\ \widehat{\theta}_{k-1}\end{bmatrix} + \begin{bmatrix}-\beta\eta\xi_k\\ 0\end{bmatrix}.$$

An elementary shows that the eigenvalues of the transition matrix at $c = 0$ are $1-\beta$ and $1-a$, which are both in the unit disk for sufficiently small c by continuity. Letting $\gamma_j(c) = \mathbb{E}\left[\widehat{\theta}_k\widehat{\theta}_{k-j}\right]$ denote the autocovariance of the process, the Yule-Walker equations for this AR(2) process Theorem 5 are given by

$$\begin{aligned}
\gamma_1(c) &= (2 - a - \beta - c(1-\beta))\,\gamma_0(c) - (1-\beta)\,(1-a-c)\gamma_1(c)\\
\gamma_2(c) &= (2 - a - \beta - c(1-\beta))\,\gamma_1(c) - (1-\beta)\,(1-a-c)\gamma_0(c)\\
\gamma_0(c) &= (2 - a - \beta - c(1-\beta))\,\gamma_1(c) - (1-\beta)\,(1-a-c)\gamma_2(c) + \beta^2\eta^2\sigma^2.
\end{aligned}$$

Solving for $\gamma_0(c)$ yields

$$\gamma_0(c) = \frac{\beta\eta^2\sigma^2\,(2 - a - \beta + a\beta - c(1-\beta))}{a\,(a + \beta - a\beta + c(1-\beta))\,(4 - 2a - 2\beta + a\beta - 2c(1-\beta))}. \tag{32}$$

By Lemma 2, it holds that

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right] = \gamma_0(c).$$

Taking the derivative of $\gamma_0(c)$ with respect to c at $c = 0$, we see that

$$\gamma_0'(0) = -\frac{2\beta(1-\beta)\eta^2\sigma^2\left((1-\beta)^2(1-a)^2 + (1-\beta)(2-a) + (1-a)\right)}{a(2-a)^2(2-\beta)^2(a+\beta-a\beta)^2} < 0.$$

Thus it holds that for sufficiently small $c > 0$,

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right] = \gamma_0(c) < \gamma_0(0) = \lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\right\|^2\right].$$

On the other hand, because

$$\widehat{\theta}^{\mathrm{SGD}}_{k+1} = (1-a)\widehat{\theta}^{\mathrm{SGD}}_k - \eta\xi_k,$$

it holds that

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}^{\mathrm{SGD}}_T - \mu^\star\right\|^2\right] = \frac{\eta^2\sigma^2}{1-(1-a)^2} = \frac{\eta^2\sigma^2}{a(2-a)} > \gamma_0(0),$$

where the inequality follows immediately from an elementary computation. □

F.3 Proof of Proposition 3

We now further specialize applying (5) to the problem outlined (28) and (29) in the setting where $\mathbf{C} = \mathbf{I}$. We state a more explicit convergence guarantee in this setting, which will allow us to show that the improvement over EMA on SGD can be arbitrarily large.

Proposition 6. *Suppose that F is as in (28) and gradient estimates g_k are as in (29). Let $\mathbf{A} = \operatorname{diag}(\alpha_1, \ldots, \alpha_d)$ let η, β such that*

$$\beta < \eta \cdot \alpha_i < 1/8$$

for all $1 \leq i \leq d$. Let $\widehat{\theta}_T$ denote the update at time T given by (22) for $\mathbf{C} = \mathbf{I}$ and fixed η. Let $\widehat{\theta}^{\mathrm{SGD}}$ evolve such that $\widehat{\theta}^{\mathrm{SGD}}_{k+1} = \widehat{\theta}^{\mathrm{SGD}}_k - \eta \cdot g_k$ and let $\widehat{\theta}_{\mathsf{EMA},k+1} = (1-\beta)\widehat{\theta}_{\mathsf{EMA},k} + \beta\theta_k$. Then

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right] \leq 8 \lim_{T\to\infty} \eta \cdot \mathbb{E}\left[\left\|\mathbf{A}^{1/2}\left(\widehat{\theta}^{\mathrm{SGD}}_T - \mu^\star\right)\right\|^2\right].$$

Proof. As in the proof of Proposition 5, we may reduce to a one-dimensional setting, writing $\alpha = \alpha_1$, $\sigma^2 = \sigma_1^2$, and $a = \eta\alpha$. Observe that for $c = 1$, the coefficients are

$$\phi_1 = 1 - a \quad \text{and} \quad \phi_2 = a(1-\beta).$$

Thus, in this setting, by (32), it holds that

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right] = \gamma_0(1) = \frac{\beta\eta^2\sigma^2\,(1 - a + a\beta)}{a(1 + a - a\beta)(2 - 2a + a\beta)},$$

whereas

$$\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\right\|^2\right] = \gamma_0(0) = \frac{\beta\eta^2\sigma^2\,(2 - a - \beta + a\beta)}{a(a + \beta - a\beta)(4 - 2a - 2\beta + a\beta)}.$$

Thus,

$$\begin{aligned}
\frac{\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_T - \mu^\star\right\|^2\right]}{\lim_{T\to\infty} \mathbb{E}\left[\left\|\widehat{\theta}_{\mathsf{EMA},T} - \mu^\star\right\|^2\right]} &= \frac{(2-a)(2-\beta)(a+\beta-a\beta)(1-a+a\beta)}{(1+a-a\beta)(2-2a+a\beta)(2-a-\beta+a\beta)} \\
&\leq \frac{8a}{(1+a-a\beta)(2-2a+a\beta)(2-a-\beta+a\beta)} \\
&\leq 8a,
\end{aligned}$$

where in the first inequality we used the fact that

$$2 - a \leq 2, \quad 2 - \beta \leq 2, \quad a + \beta - a\beta \leq a + \beta \leq 2a, \quad \text{and} \quad 1 - a + a\beta \leq 1.$$

In the second inequality, we used the fact that

$$1 + a - a\beta \geq 1 \quad \text{and} \quad 2 - 2a + a\beta \geq 2 - 2a \geq 1.$$

The result then follows. □

The proof of Proposition 3 then follows immediately from the preceding proposition.

F.4 Stability and relative improvement in the quadratic model

We now use the covariance formulas to exhibit a stable regime in which both errors vanish while their ratio tends to zero. In the scalar setting, write $a = \eta\alpha$. For $C = I$, the returned EMA satisfies an AR(2) recursion with coefficients

$$\phi_1 = 1 - a, \qquad \phi_2 = a(1-\beta).$$

The corresponding stability conditions are

$$
\begin{aligned}
1 - \phi_1 - \phi_2 &= a\beta > 0, \\
1 + \phi_1 - \phi_2 &= 2 - 2a + a\beta > 0, \\
1 - \phi_2 &= 1 - a(1 - \beta) > 0.
\end{aligned}
$$

These inequalities hold for $0 < \beta < a < 1/8$, so the process is causal and has finite stationary covariance throughout this region under the stated noise assumptions.

To make the relative improvement explicit, fix $\alpha = 1$ and a positive noise variance σ^2, and take

$$
\eta = a, \qquad \beta = a^2, \qquad a \downarrow 0.
$$

Let $\gamma_0(c)$ denote the stationary squared error of the returned EMA for pullback strength c in the scalar stylized update. Substituting into the covariance formulas gives

$$
\gamma_0(0) \sim \frac{\sigma^2 a^2}{2}, \qquad \gamma_0(1) \sim \frac{\sigma^2 a^3}{2}.
$$

Consequently, both absolute errors tend to zero, while

$$
\frac{\gamma_0(1)}{\gamma_0(0)} \sim a \longrightarrow 0.
$$

For each fixed $a > 0$, we take the stationary limit before letting $a \downarrow 0$. This is an extremal existence result, not a prescription for the pullback strength used in practice. The small-pullback proposition provides the complementary result: under its assumptions, a sufficiently small positive pullback improves the limiting EMA error at any fixed stable learning rate and averaging parameter.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state two contributions — (i) the optimal-control derivation of the per-coordinate pullback toward the iterate average, and (ii) the empirical demonstration that the discretized optimizer of Algorithm 1 strictly improves on AdamW and EMA-evaluated AdamW across SmolLM2, Qwen3, and Gemma3 at $\geq$1B parameters. Both are matched by the corresponding sections (theory in Section 3, experiments in Section 4); no additional claim is made.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We extensively detail limitations in Section 5, including theoretical limitations of the assumptions under which PACE was derived, empirical limitations of the scale and scope of our experiments, and practical limitations of the memory overhead of PACE. We also discuss directions for future work that would address these limitations.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The continuous-time controlled SDE setup is stated in Section 2 (assumptions: positive-definite Hessian, additive Gaussian noise, quadratic objective). The derivation of the optimal control via the matrix Riccati equation is in the appendix (Section E); the convex convergence guarantee is in Section F, with all assumptions explicit.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section A contains the full optimizer update rule, the per-experiment hyperparameter table (Table 2), data preprocessing details, hardware, seeds, and the calibrated c-grid construction. The pre-trained checkpoints and datasets used are all publicly available.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The optimizer implementation, training scripts, configuration templates, and reproduction instructions are available at https://github.com/leonou2010/pace-optimizer. All datasets and pre-trained models used are publicly available under their original licenses.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section A reports model cards (Table 1), per-experiment hyperparameter grids (Table 2), the explicit optimizer update rule, the calibration protocol used to choose c, weight-decay and Adam beta defaults, learning-rate selection, evaluation protocol, and seed setup.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Each headline cell in the body figures is reported at a single seed (seed 42) due to per-run wall-time cost at 1.7B parameters; we acknowledge this in Section A. This is consistent with the standard practice in the literature on LM training and optimization, as can be seen in [23, 37]. Indeed, training with warmup and relatively small learning rates means that, unlike RL and on-policy settings, standard cross-entropy minimization is relatively low variance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section A discloses the GPU type (NVIDIA RTX PRO 6000 Blackwell, 96 GB), the TPU type used for the GPT-2 pretraining track (Cloud TPU v6e), and per-run training budgets (Table 1).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics. The work is purely an optimizer-design contribution evaluated on standard public datasets and pre-trained checkpoints; it involves no human subjects, no scraped data, and no deployed system.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A]

Justification: This work is foundational research and has no societal impact beyond that which is standard for improving the training of LMs. We do not identify any direct path to negative applications, but we acknowledge that any general-purpose acceleration for LM training indirectly amplifies dual-use risks of large language models; the mitigations are the same as for the LM ecosystem more broadly (responsible release of resulting models, content filters, etc.), and we do not release any new model artifact in this paper.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: We release optimizer code, training logs, and configuration files; we do not release any new pre-trained model, scraped dataset, or generative artifact with high misuse risk. All evaluation models (SmolLM2, Qwen3, Gemma3, GPT-2) and datasets (`smol-smoltalk`, `tulu-3`, FineWeb) used in this paper are existing public assets covered by their respective release safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets are properly cited at their first mention: SmolLM2 [1] (Apache-2.0), Qwen3 [41] (Apache-2.0), Gemma3 [11] (Gemma terms-of-use), `smol-smoltalk` [2] (Apache-2.0), `tulu-3` [21] (Open Data Commons), FineWeb [30] (ODC-By), GPT-2 [32], and Schedule-Free AdamW [9] (Apache-2.0). All assets are used within their license terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new asset is the PACE optimizer (Algorithm 1). The release on acceptance will include the optimizer source, documentation, configuration templates, and a README giving the commands and environment needed to reproduce the reported sweeps. We do not release model checkpoints or private per-run training logs.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper involves no crowdsourcing and no research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper involves no human subjects, so no IRB approval is applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We do not use LMs in any non-standard way other than to help write and edit the paper. We do use LMs to help write code as well as surface references.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.