TRAILMIX: ADAPTIVE INTERPOLATION BETWEEN OPTIMIZERS WITH CONVERGENCE GUARANTEES

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizers are central to modern deep learning, yet no single algorithm consistently excels across architectures or datasets. Existing methods of adaptively mixing optimizers to combine complementary strengths are promising, but are restricted to narrow optimizer families or lack rigorous guarantees, leaving a gap between theory and practice. To fill this gap, we present **TRAILMIX**, an adaptive interpolation framework that is general across all first- and quasi-second-order methods. On the theoretical front, we prove that convex combinations of optimizers satisfying a mild alignment condition preserve standard convergence rates in non-convex, convex, and strongly convex or PL regimes. For the challenging same-timescale setting, we establish a novel analysis method by lifting the stochastic dynamics to a population-level Fokker-Planck PDE, for which we prove stability using a joint free-energy Lyapunov function. Algorithmically, we extend this framework with fairness normalization, trust-region clipping, and a curvature-awareness reward that stabilizes the meta-weights and enables smoother training. These additions allow **TRAILMIX** to behave like an ensemble when optimizers are complementary and to concentrate weight when one dominates, without breaking convexity. Our empirical evaluations on an optimizer set including AdamW, Lion, SOAP, Scion, and MARS show that **TRAILMIX** consistently matches or outperforms the strongest single optimizer across a wide range of analytic loss surfaces.

1 INTRODUCTION

First-order optimization is the engine of modern machine learning, yet the landscape of optimizers is increasingly fragmented. While Adam Kingma & Ba [2017] remains the standard, a proliferation of specialized algorithms Chen et al. [2023]; Vyas et al. [2025]; Pethick et al. [2025] demonstrate superior performance on specific tasks, architectures, and data modalities. This specialization stems from a fundamental set of trade-offs between convergence speed, memory footprint, and generalization, creating a "no free lunch" scenario where no single optimizer is universally optimal.

Recent work on meta-adaptive optimizers like MADA [Ozkara et al., 2024] has shown empirically that dynamic mixtures can outperform single optimizers. While MADA has excellent performance, it is limited to a few optimizers and requires internal state knowledge.

To address these limitations, we introduce **TRAILMIX**, which makes several key algorithmic innovations guided by our theoretical analysis. Unlike MADA's simple advantage scoring, we introduce **fairness normalization** to prevent weight concentration, **trust-region clipping** for stability, and a **curvature-awareness reward** that estimates local Hessian information using gradient differences (Algorithm 1, lines 8-9). These enhancements are directly motivated by our convergence analysis: fairness normalization ensures the mixture remains in the simplex interior (critical for our entropy-based Lyapunov function), while curvature awareness improves the alignment constant $c(\lambda)$ that governs our convergence rates.

Our empirical evaluation demonstrates the practical impact of this theory-guided design. On the Rosenbrock function, **TRAILMIX**'s curvature-aware reweighting enables it to shift from aggressive learning rates in smooth regions to conservative rates near the optimum, converging in 292 steps versus 327-1671 for fixed-rate Adam variants. Under distribution shift, our advantage scoring detects

when adaptive optimizers become misaligned due to stale internal state, automatically reallocating weight to SGD and avoiding the corrective oscillations that plague fixed combinations.

However, proving convergence for such adaptive mixtures is notoriously difficult, as the optimizer itself becomes a non-stationary target. Existing analyses either assume timescale separation (impractical) or provide no guarantees for the challenging same-timescale regime where weights and parameters evolve concurrently. This theoretical gap has left the field without principled guidance for designing adaptive optimizer combinations.

Our main theoretical contribution bridges this gap through a novel analysis technique that lifts the discrete-time stochastic dynamics to a population-level Fokker-Planck PDE. We construct a joint free-energy Lyapunov function $V(x, \lambda) = f(x) - \tau H(\lambda)$ combining the objective with Shannon entropy, proving stability without requiring timescale separation (D). This analysis directly informs our algorithmic choices: the entropy regularization that emerges naturally from our PDE analysis guides our fairness normalization, while the alignment condition in our convergence proof validates our curvature-awareness reward design.

Our contributions are threefold:

- **Principled Algorithm Design:** We extend basic optimizer interpolation with theory-motivated enhancements including fairness normalization, trust-region clipping, and curvature awareness that address limitations of existing methods like MADA.
- **Same-Timescale Convergence Analysis:** We provide the first rigorous convergence guarantees for adaptive optimizer mixtures where weights and parameters update concurrently, using a novel PDE-based proof technique with joint Lyapunov functions.
- **Theory and Implementation:** Our algorithmic design is directly guided by theoretical insights, with explicit cross-references between our convergence analysis and implementation choices, demonstrating how rigorous theory translates to practical improvements.

This principled approach eliminates the gap between heuristic methods and rigorous analysis, providing theoretical guarantees and empirical performance across diverse optimization landscapes.

2 Related Works

2.1 Adaptive First-Order Methods

Adaptive First-Order & Quasi-second-Order Methods. Modern deep learning relies heavily on adaptive first-order optimizers such as AdaGrad Duchi et al. [2011], RMSprop Tieleman & Hinton [2012], and Adam Kingma & Ba [2017], which adjust learning rates per parameter using gradient history. Refinements like AdamW Loshchilov & Hutter [2019] improved regularization, while Lion Chen et al. [2023] introduced a momentum-sign update rule with strong empirical performance. More recently, quasi-second-order approaches have emerged: SOAP Vyas et al. [2025] stabilizes training by approximating curvature information, and SCION Pethick et al. [2025] extends this idea with scalable preconditioning. Despite their differences, no single method dominates across tasks, motivating principled ways to combine their complementary strengths.

Optimizer Combination and Meta-Optimization. Recent efforts have explored adaptive optimizer combination and meta-optimization of training procedures. Hypergradient methods [Baydin et al., 2017] adapt learning rates online by differentiating through the optimizer, with recent work establishing the first convergence guarantees [Chu et al., 2025]. Meta-optimization frameworks like MetaOptimize [Amid et al., 2024] adjust hyperparameters to minimize long-term regret, avoiding costly grid search. Local curvature-based approaches [Jiang et al., 2025] exploit gradient alignment and curvature estimates for dynamic step-size adjustment, while neural and reinforcement learning-based meta-optimizers [Wang et al., 2024] frame optimizer selection as sequential decision-making, albeit with high computational cost. Other approaches focus on architecture-aware scaling [Refinetti et al., 2024] or leverage large language models for hyperparameter search [Brown et al., 2024]. Yet, existing methods are often limited to narrow optimizer families, incur expensive meta-learning, or lack theoretical guarantees. We address these challenges with a principled framework for combining arbitrary first- and quasi-second-order optimizers with provable convergence.

Population-Level and PDE-Based Analysis. A parallel line of theoretical work analyzes stochastic optimization by studying the evolution of the entire distribution of network parameters, rather than a single trajectory. In the continuous-time limit, SGD updates can be modeled by an SDE whose density follows a Fokker-Planck equation [Gardiner, 1985]. This connects SGD with Langevin dynamics and has been used to study how noise helps optimizers escape sharp minima and find flatter, more generalizable solutions [Jastrzebski et al., 2017; Chaudhari et al., 2018]. While ODE-based methods are well-suited for two-timescale analysis [Borkar, 1997], they are insufficient for the same-timescale regime where parameters and meta-parameters evolve concurrently. Our work leverages the PDE perspective to analyze these more complex dynamics, using a free-energy functional to establish stability and convergence without requiring timescale separation, drawing inspiration from the mean-field analysis of large-scale interacting systems [Sirignano & Spiliopoulos, 2020].

3 Problem Setup

We consider minimizing a differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ via stochastic first-order updates. Let $g(x_t, \xi_t)$ be an unbiased estimator of $\nabla f(x_t)$, i.e. $\mathbb{E}[g(x_t, \xi_t) \mid x_t] = \nabla f(x_t)$. We have K base optimizers $\{\mathcal{O}_i\}_{i=1}^K$, each producing an update direction

$$d_t^{(i)} = \mathcal{O}_i\big(x_t, g(x_t, \xi_t), S_{t-1}^{(i)}\big), \quad i = 1, \dots, K, \tag{1}$$

where $S_{t-1}^{(i)}$ denotes internal state (e.g., momentum buffers, variance accumulators, or curvature estimates). We then form a *convex combination*:

$$d_t = \sum_{i=1}^K \lambda_{i,t}\, d_t^{(i)}, \qquad \lambda_{i,t} \geq 0,\ \sum_{i=1}^K \lambda_{i,t} = 1, \tag{2}$$

with update

$$x_{t+1} = x_t - \alpha_t\, d_t. \tag{3}$$

Meta-dynamics. The mixture weights $\lambda_t = (\lambda_{1,t}, \dots, \lambda_{K,t})$ evolve according to a meta-learning rule. Two common regimes exist:

- *Two-timescale:* $\beta_t = o(\alpha_t)$, so λ_t evolves slower than x_t (stochastic approximation).
- *Same-timescale:* $\beta_t = \theta\alpha_t$ with fixed $\theta > 0$. This regime matches practice and leads to a coupled ODE/PDE analysis.

A generic update for the weights is

$$\lambda_{t+1} = \Pi_\Delta(\lambda_t + \beta_t h(x_t, \lambda_t, \xi_t))\,, \tag{4}$$

where h is a stochastic meta-gradient and Π_Δ denotes projection onto the probability simplex $\Delta_K = \{\lambda \in \mathbb{R}^K_{\geq 0} : \sum_i \lambda_i = 1\}$. In the continuous-time limit, this can correspond to the replicator flow:

$$\dot{\lambda}_i = \theta\lambda_i\Big(h_i(x, \lambda) - \langle \lambda, h(x, \lambda)\rangle\Big). \tag{5}$$

3.1 Assumptions and Notation

Assumption 1 (Smoothness)**.** *f is L-smooth: $\forall x, y$,*

$$f(y) \leq f(x) + \nabla f(x)^\top (y - x) + \tfrac{L}{2}\|y - x\|^2.$$

Assumption 2 (Bounded Second Moments)**.** *There exists $G > 0$ such that $\mathbb{E}[\|d_t^{(i)}\|^2] \leq G^2$ for all i, t. By convexity, this implies $\mathbb{E}[\|d_t\|^2] \leq G^2$.*

Assumption 3 (Non-Negligible Alignment)**.** *For each base optimizer i, there exists a constant $c_i > 0$ ensuring the expected update direction is aligned with the negative gradient:*

$$\nabla f(x_t)^\top \mathbb{E}[d_t^{(i)} \mid x_t] \ \geq\ c_i\, \|\nabla f(x_t)\|^2. \tag{6}$$

This ensures the mixture also points in a descent direction:

$$\nabla f(x_t)^\top \mathbb{E}[d_t \mid x_t] \ \geq\ c(\lambda_t)\, \|\nabla f(x_t)\|^2, \qquad c(\lambda_t) = \sum_{i=1}^K \lambda_{i,t}\, c_i > 0.$$

Assumption 4 (Step Sizes)**.** *For diminishing steps, we require the standard Robbins-Monro conditions:* $\alpha_t > 0$, $\sum_t \alpha_t = \infty$, *and* $\sum_t \alpha_t^2 < \infty$.

Remark 1 (Alignment Justification)**.** *This key assumption is satisfied by a wide range of common optimizers. For Adam-like methods, the update can be viewed as a preconditioned gradient method where the preconditioner's eigenvalues are bounded away from zero. This guarantees alignment, as shown formally in Appendix F and verified for common adaptive methods in Appendix G (see, e.g., Lemma 14). If an optimizer becomes misaligned, the meta-learning rule is designed to drive its corresponding weight* $\lambda_{i,t}$ *toward zero.*

Extension to 1.5-order bases. The framework naturally extends to optimizers that use preconditioners ($P_i(x)$) or inertia (v):

$$d^{(i)}(x) = P_i(x)\,\nabla f(x) + \beta_i v,$$

where $P_i(x)$ is a symmetric positive semidefinite matrix. The minimum eigenvalue of $P_i(x)$ directly corresponds to the alignment constant c_i (Lemma 10). This class of methods, which includes optimizers like SOAP and Scion, is formalized in Appendix F.

Population (PDE) viewpoint. For constant step sizes, the dynamics can be viewed through a population-level lens using a Fokker-Planck equation:

$$\partial_t \mu_t + \nabla \cdot (\mu_t v_{\lambda_t}) = \sigma \Delta \mu_t, \quad v_\lambda(x) = \sum_i \lambda_i\, d^{(i)}(x). \tag{7}$$

This coupled system admits a joint **free-energy Lyapunov function** whose dissipation is established in Lemma 9 (Appendix E), allowing for a stability analysis even when the model parameters and mixture weights adapt at the same rate.

4 Convergence for Fixed Mixtures

We first analyze the case where mixture weights $\lambda_{i,t} \equiv \lambda_i$ are fixed, establishing that the interpolation framework preserves standard convergence guarantees.

Theorem 1 (Convergence Rates)**.** *Under smoothness, bounded second moments, and alignment assumptions (Assumptions 1–3), the mixed optimizer achieves:*

1. ***Non-convex:*** $\min_{0\le t<T} \mathbb{E}[\|\nabla f(x_t)\|^2] = O(\ln T/\sqrt{T})$ *with* $\alpha_t \propto 1/\sqrt{t+1}$
2. ***Convex:*** $\mathbb{E}[f(x_t) - f^*] = O(1/\sqrt{T})$ *with appropriate step sizes*
3. μ***-PL/Strongly convex:*** $\mathbb{E}[f(x_t) - f^*] = O((1-\gamma)^t)$ *for some* $\gamma \in (0,1)$ *with constant step size* α

The interpolation preserves expected convergence rates of base optimizers. Proofs in Appendix B.

5 Convergence with Time-Varying Mixtures

We analyze an adaptive framework that jointly updates mixture weights λ_t and model parameters. Although classical two-timescale analysis ($\beta_t = o(\alpha_t)$) guarantees convergence(Theorem 6 in Appendix C), our implementation uses same-timescale updates, which is more common in practice.

5.1 Same-Timescale Convergence

In practice, setting $\beta_t = \theta\alpha_t$ for fixed $\theta > 0$ means weights and parameters evolve concurrently—a challenging regime requiring novel analysis.

Theorem 2 (Same-Timescale Convergence)**.** *The continuous-time flow with* $\beta_t = \theta\alpha_t$ *converges to a meta-stationary point* (x^*, λ^*) *where* $\nabla f(x^*) = 0$ *and weights reach equilibrium. Under the* μ*-PL condition,* $f(x(t))$ *converges linearly to* f^*.

Proof technique: We construct a joint Lyapunov function $\mathcal{V}(x, \lambda) = f(x) - \tau H(\lambda)$ combining the objective with Shannon entropy $H(\lambda)$. This free-energy functional decreases along trajectories, establishing stability without timescale separation—a key theoretical contribution enabling practical same-timescale implementations. Full analysis in Appendix D.

6 How Adaptive Interpolation Improves Convergence

The convergence rates derived in Section 4 depend directly on the mixture-weighted alignment constant, $c(\lambda_t)$, and the second-moment bound, G^2. The central benefit of **TrailMix** is its ability to dynamically adapt the mixture λ_t to find a favorable trade-off between these competing factors.

To illustrate, consider the non-convex rate from Theorem 3, which is roughly proportional to:

$$\min_{t<T} \mathbb{E}[\|\nabla f(x_t)\|^2] \propto \frac{f(x_0) - f^* + LG^2 \ln T}{c(\lambda_t)\sqrt{T}}. \tag{8}$$

To accelerate convergence, the meta-optimizer must find a λ_t that maximizes the alignment-to-variance ratio, effectively maximizing $c(\lambda_t)$ while minimizing the impact of G^2.

For example, one optimizer $\mathcal{O}_1$ may be aggressive, with large alignment c_1 but high variance G_1^2, while another $\mathcal{O}_2$ is more conservative with smaller c_2 and lower variance G_2^2. **TrailMix** optimally combines these trade-offs across training phases, which no fixed optimizer can achieve.

7 TrailMix Implementation

Algorithm 1 TrailMix Algorithm Outline

Input: base optimizers $\{\mathcal{O}_i\}_{i=1}^K$, meta step sizes $\{\eta_t\}$, base step sizes $\{\alpha_t\}$, curvature weight $\gamma \geq 0$, max steps T.
Initialize: parameters $x_0 \in \mathbb{R}^d$, mixture $\lambda_0 \in \Delta_K$, caches $g_{-1} \leftarrow 0$, $s_{-1} \leftarrow 0$.
for $t = 0$ **to** $T - 1$ **do**
  Sample minibatch and compute gradient $g_t \leftarrow \nabla \ell(x_t; \xi_t)$.
  for each i: $\Delta_t^{(i)} \leftarrow \textsc{Propose}(\mathcal{O}_i, x_t, g_t)$ ▷ *Base proposals*
  $\Delta_t^{\text{mix}} \leftarrow \sum_{i=1}^K \lambda_t^{(i)} \Delta_t^{(i)}$; $x_{t+1} \leftarrow x_t - \alpha_t \Delta_t^{\text{mix}}$ ▷ *Mixed update*
  for each i: $h_t^{(i)} \leftarrow \langle g_t, \Delta_t^{(i)} \rangle - \langle g_t, \Delta_t^{\text{mix}} \rangle$ ▷ *Advantage scoring*
  $y_t \leftarrow g_t - g_{t-1}$, **if** $\langle y_t, s_{-1} \rangle > 0$ **then** ▷ *Diagonal curvature score*

$$\widehat{H}_{\text{diag}} \approx \frac{|y_t|}{|s_{-1}| + \epsilon}, \quad d_N \leftarrow -g_t \oslash \widehat{H}_{\text{diag}}, \quad r_t^{(i)} \leftarrow \cos(\Delta_t^{(i)}, d_N) \cdot \frac{\langle y_t, s_{-1} \rangle}{\|y_t\| \|s_{-1}\|},$$

    set $h_t^{(i)} \leftarrow h_t^{(i)} + \gamma\, r_t^{(i)}$
  $\lambda_{t+1}'^{(i)} \leftarrow \lambda_t^{(i)} \exp(\eta_t h_t^{(i)})$; $\lambda_{t+1} \leftarrow \Pi_{\Delta_K}(\lambda'_{t+1})$ ▷ *Mirror-descent on base weights*
  $g_{t-1} \leftarrow g_t$, $s_{-1} \leftarrow -\Delta_t^{\text{mix}}$ ▷ *Update caches*
end for
Output: x_T, λ_T.

The main outline for our implementation of **TrailMix** is given in Alg. 1, based on the convergence theorems and assumptions from the previous sections. **TrailMix** maintains a convex mixture over K base optimizers and updates model parameters using the mixed proposal while adapting mixture weights $\lambda \in \Delta_K$ online. Each training step proceeds as follows: (1) obtain per-base proposals with persistent "shadow" copies of the base optimizers whose *state is synchronized from the real bases* before proposing; (2) apply the mixed update to the real model; (3) compute meta signals—an *advantage* score (predicted decrease relative to the current mix) and an optional *diagonal curvature* bonus (secant-based Hessian proxy and a cosine match to the diagonal-Newton direction); and (4) update λ via mirror descent (multiplicative weights) with simplex projection.

For readability, the outline omits stability refinements that we implement in code: a short warm-up (smaller meta step size) to let base states populate, an entropy term that encourages early weight dispersion and later anneals to zero, a *state-only* advancement of real bases, temporarily setting

LR=0 and applying $\lambda_i g_t$, to keep internal moments aligned with the mixed trajectory, and small numerical ε floors to avoid divide-by-zero in diagonal curvature estimates.

PyTorch integration. **TRAILMIX** is implemented as a `torch.optim.Optimizer` subclass for seamless use in existing training pipelines. Users initialize base optimizers on model parameters and pass them to **TRAILMIX**, which supports `optimizer.step()` while also advancing each base optimizer's scheduler. This design ensures compatibility with both built-in and external optimizers, e.g., from `pytorch-optimizer` [Kim, 2021].

Complexity. **TRAILMIX** reuses a single backward pass to obtain g_t and then performs K first-order optimizer steps on shadow copies to form proposals, so the per-step time scales as $O(K)$ relative to a single base optimizer (plus lightweight scoring and a projection). Memory scales as $O(K|\theta|)$ (where θ denotes model size) for shadow parameters and per-base state. While we recognize these limitations, we intend this work as a preliminary investigation into the design and utility of interpolated optimizers, with further inquiry into base optimizer set compositions, meta-optimizers with access to internal model state, and algorithmic improvements being required for practical implementations.

8 RESULTS

We evaluate **TRAILMIX** on various two-dimensional analytic loss surfaces. This setting guarantees our smoothness assumptions while enabling direct visualization and inspection of optimizer trajectories. We first highlight behaviors that **TRAILMIX** is designed to exploit–adaptive learning rate selection, robustness under distribution shift, and sparse/mixed adaptation on engineered surfaces–then benchmark against a wide set of constituent base optimizers on well-known test functions.

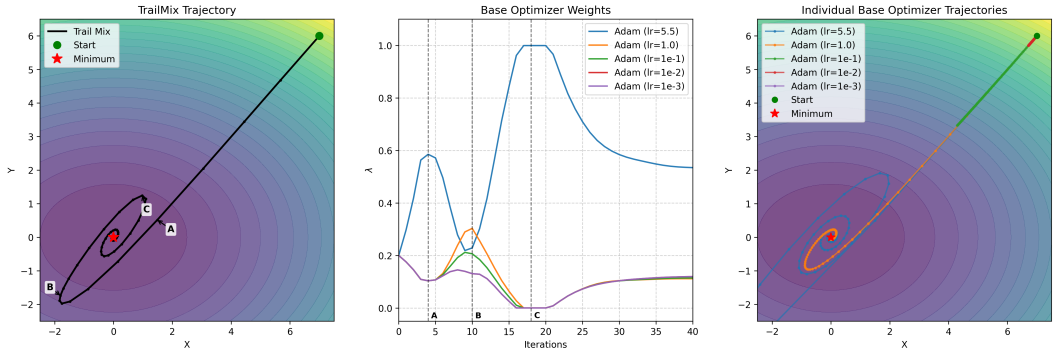


Figure 1: **TRAILMIX** selects learning rate to adjust to the rate of change of the local gradient. The interpolator is initialized with 5 instances of Adam, with learning rates ranging from 5.5 to 10^{-3}. At the annotated points, the weighting mixture shifts to either prevent overshoot (A,C) or accelerate when a higher learning rate aligns with the gradient (B). Left: **TRAILMIX** trajectory; middle: mixture weights λ; right: individual fixed-LR Adam trajectories.

Learning rate selection. Learning rate is a primary factor affecting optimization efficacy: values that are too large induce divergence or oscillation, while values that are too conservative cause stagnation and slow descent. Although modern optimizers modulate their effective step via internal state, those dynamics can lag rapid changes in local curvature or gradient magnitude. **TRAILMIX** addresses this by mixing bases with distinct learning rates and reallocating their weights online. In Fig. 1, **TRAILMIX** initially places mass on the aggressive base to traverse the smooth outer region (A), then shifts toward moderate rates to reduce overshoot as curvature increases (B), and assigns small but nonzero mass to low rates to stabilize near the minimizer (C). This is contrasted with trajectories traversed by the individual base optimizers–either vastly overshooting the minimizer or using overly cautious steps. Under a 5,000-step budget and a stopping criteria of a cumulative loss decrease $< 10^{-6}$ over a 200-step window, **TRAILMIX** reaches tolerance in 292 steps (distance to minimizer $d_{min} = 4.1 \times 10^{-11}$), outperforming fixed-LR Adams at 327 (LR=5.5), 353 (LR=1.0), and 1671 steps (LR=10^{-1}); LR=10^{-2} and 10^{-3} do not converge within budget.

Optimizer selection under parameter shift. The design of many optimizers imposes inductive biases–e.g., momentum-based temporal smoothing and adaptive preconditioning–that shape how they estimate and respond to local geometry. In particular, adaptive gradient methods implicitly assume locally smooth objectives and slowly varying gradient statistics. These assumptions can break down when the landscape changes abruptly, whether due to an ill-scaled learning rate, sharp topological features, or, in ML settings, a distribution shift.

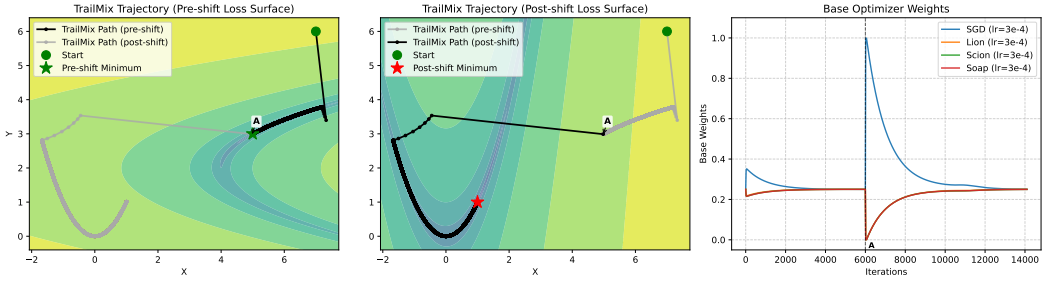
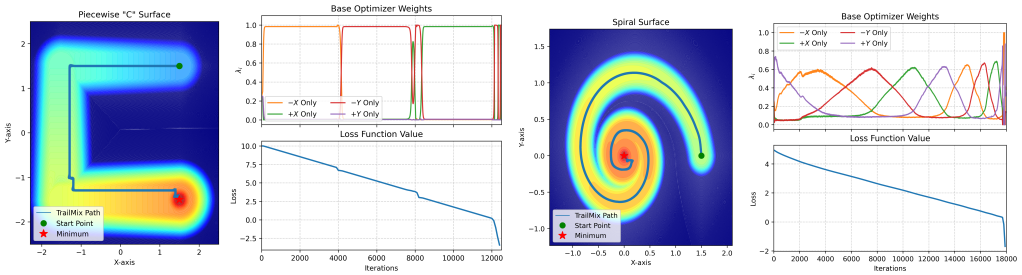


Figure 2: Parameter distribution shift prompts **TRAILMIX** to adjust optimizer weighting. The loss surface seen by **TRAILMIX** changes abruptly after 6,000 steps–prior to the shift, weight is allocated early to SGD for fast descent into the valley, then shifted towards adaptive optimizers for controlled progress in the valley. After the distribution shift occurs, weight is allocated back to SGD for fast descent until the valley is reached again. Left: Initial loss surface and optimizer trajectory; middle: post-shift loss surface and trajectory; right: mixture weights

Fig. 2 examines **TRAILMIX** under a sudden objective change. For the first 6,000 steps it minimizes a rotated/translated Rosenbrock (minimum at $(5, 3)$); the landscape then switches instantaneously to the standard Rosenbrock (minimum at $(1, 1)$). The internal state of the adaptive bases, including moment estimates, are not modified by the objective change. At the start of the optimization, the steep wall favors weighting SGD; as the trajectory enters the narrow valley in the first phase, weights to adaptive bases are increased to temper overshoot. Immediately after the shift, however, those adaptive bases are state-stale, with their accumulated moments reflecting the pre-shift regime, so their proposals misalign with the new gradient field. **TRAILMIX** detects this misalignment and quickly reallocates weight to SGD, whose proposal depends only on the current gradient, thereby avoiding corrective oscillations and preserving forward progress toward the new minimizer. Upon re-entering the valley, weight is re-allocated back to the adaptive bases. Practically, this makes **TRAILMIX** well-suited for nonstationary regimes without resetting optimizer state or retuning hyperparameters.



(a) Piecewise "C" Channel (b) Spiral Channel

Figure 3: **TRAILMIX** descent on engineered loss surfaces with naive axis-aligned base optimizers. In the "C" channel, descent requires fast, sparse switches between bases to follow piecewise-constant gradients, demonstrating near one-hot adaptation. In the spiral channel, descent requires continuous mixing of two directions to track curved gradients, demonstrating the ability to smoothly blend optimizer weights. Together, these surfaces highlight the framework's capacity for both discrete and continuous adaptation.

		TrailMix	Scion	MARS	SOAP	Lion	AdamW	Adam	Adadelta	SGD
Quadratic	Best config	—	Scion-3	Mars-1	Soap-3	Lion-3	AdamW-3	Adam-3	Adadelta-2	SGD-3
	Converged?	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
	Dist. to min	9.40e-05	1.25e-02	6.51e-06	2.34e-05	3.38e-04	1.81e-01	1.27e-05	7.90e-30	2.51e-05
	Steps	1399	—	11002	6062	6239	—	6762	5028	1252
Rosenbrock	Best config	—	Scion-2	Mars-1	Soap-3	Lion-2	AdamW-2	Adam-3	Adadelta-3	SGD-3
	Converged?	Yes	No	Yes	Yes	Yes	No	Yes	No	No
	Dist. to min	1.24e-03	1.00e-01	3.38e-05	4.52e-04	1.69e-04	2.31e+00	5.74e-05	7.56e-02	3.84e-03
	Steps	6284	—	12248	6964	13596	—	7845	—	—
Booth	Best config	—	Scion-3	Mars-1	Soap-3	Lion-3	AdamW-3	Adam-3	Adadelta-2	SGD-3
	Converged?	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes
	Dist. to min	5.04e-04	3.70e-02	1.99e-01	1.08e-05	3.74e-04	1.51e+00	3.44e-07	1.11e-07	1.72e-04
	Steps	1210	—	—	11217	11369	—	12412	8259	4967
Griewank	Best config	—	Scion-3	Mars-1	Soap-3	Lion-3	AdamW-3	Adam-3	Adadelta-3	SGD-3
	Converged?	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No
	Dist. to min	9.81e-04	1.45e-01	6.14e-06	4.53e-06	4.61e-04	8.05e-04	1.28e-06	2.37e-14	3.34e-03
	Steps	3336	—	8018	4374	4802	10711	4705	1146	—
Himmelblau	Best config	—	Scion-2	Mars-1	Soap-3	Lion-3	AdamW-3	Adam-3	Adadelta-1	SGD-3
	Converged?	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes
	Dist. to min	5.93e-05	4.05e-03	9.63e-06	1.95e-05	2.86e-04	9.22e-01	2.78e-08	2.15e-03	7.27e-06
	Steps	635	—	11026	6087	6290	—	7007	—	587
3-Hump	Best config	—	Scion-3	Mars-1	Soap-3	Lion-3	AdamW-3	Adam-3	Adadelta-3	SGD-3
	Converged?	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
	Dist. to min	4.34e-04	2.65e-02	8.72e-06	2.28e-05	2.92e-04	2.00e+00	4.76e-07	1.95e+00	2.28e-04
	Steps	2766	—	11006	6067	6238	—	6810	3535	7739

Figure 4: **TRAILMIX** competes with or outperforms base optimizers. Table reports the results of convergence experiments on well-known loss surfaces; **TRAILMIX** is baselined against its set of 24 constituent optimizers. For each family, the best performing member on each task is reported. **TRAILMIX** performs best or second best with respect to convergence speed on all surfaces, Blue denotes the fastest, Red denotes second.

Engineered loss surfaces. In Fig. 3 we showcase important properties of our interpolated optimizer on manufactured 2D loss surfaces. To decouple the behavior of **TRAILMIX** from the behavior of its base optimizers, we use a base set of "dummy optimizers" that do not give step proposals based on the local loss landscape, instead returning a fixed gradient direction (one of the 4 axis-aligned directions $+X$, $+Y$, $-X$ and $-Y$) and magnitude. These naive base optimizers allow us to investigate the weight allocation and gradient alignment behavior of the meta-optimizer without having to account for changing update proposal directions from the base optimizers.

Sparse adaptation (piecewise "C" channel). We demonstrate a case where a sparse weighting of optimizers is needed to find the minimum of the surface, with fast switching between optimizers required for different regimes of the function. For this we use a blocky "C-shaped" channel, with a constant gradient direction (starting from the $+X/+Y$ quadrant) aligned sequentially with the $-X$, $-Y$, and $+X$ directions, and a quadratic sink on the final edge to stop the optimizer. **TRAILMIX** learns a sparse mixture that concentrates weight on the single direction aligned with the current segment, switching quickly at the corners. This confirms that the meta-optimizer can discover near one-hot mixtures when the loss surface requires regime-specific behavior.

Mixed adaptation (spiral channel). We also demonstrate the case where a mixed solution is required to descend the loss surface. For this we use a spiral-shaped channel, where the gradient is always oriented parallel to the curve, pointing counter-clockwise. In this setting, the meta-optimizer must continually adjust a balance of 2 of the naive optimizers for efficient descent. **TRAILMIX** is able to smoothly varying λ to track the surface, maintaining alignment without incurring the lag associated with purely stateful single optimizers.

Well-known optimization test functions. To benchmark the convergence speed increases enabled by **TRAILMIX**, we perform experiments on a set of well-known optimizer test functions. Fig. 5 in the appendix outlines the configurations for the 24 constituent base optimizers used by **TRAILMIX**, sorted by optimizer family. Multiple modern adaptive optimizers (MARS, SOAP, etc.) are in the base set, and a range of learning rates are provided to each optimizer family to maximize the chance at least one will converge. The results of the experiments are shown in Fig. 4. Each algorithm

was allowed 20,000 steps to converge. For each surface, the fastest-converging optimizer from each family was reported (and if none converged, the one that ended closest to the minimizer)

TRAILMIX attains the first or second fastest time-to-tolerance across all surfaces. It is the fastest on 3/6 surfaces (Rosenbrock, Booth, 3-Hump) and is within a few percent of the best baseline on two of the remaining (Himmelblau, Quadratic), while achieving competitive final distances to the minima. These results indicate that **TRAILMIX** can successfully integrate contemporary optimization algorithms, and reliably matches the best fixed optimizer on each surface without prior knowledge of which optimizer (or hyperparameters) will be best, validating the benefit of online mixture reallocation.

9 DISCUSSION

This work establishes a principled foundation for meta-adaptive optimization, but it also illuminates several exciting avenues for future inquiry. Our analysis and implementation of **TRAILMIX** pave the way for a new class of intelligent optimization systems.

A primary opportunity lies in **extending the framework to black-box optimizers**. Our current theoretical model assumes access to internal optimizer states, $S^{(i)}$. Developing a formulation that works with black-box optimizers would dramatically broaden **TRAILMIX**'s applicability, enabling its use with proprietary or highly complex algorithms without modification. This would create a truly "plug-and-play" meta-optimizer.

Furthermore, our results highlight the potential for **real-time adaptation to non-stationary training dynamics**. The ability of **TRAILMIX** to reallocate weights in response to a distribution shift (Figure 2) is a key strength. Future work could leverage this to explicitly detect training phase transitions, automatically adjust to continual learning scenarios, or serve as an early warning system for dataset shifts, making training far more robust in dynamic environments.

Finally, a crucial research direction is the **principled curation of the base optimizer ensemble**. While **TRAILMIX** effectively combines a given set of optimizers, the question of how to select this set remains open. Identifying which optimizers are complementary versus redundant is a complex challenge that goes beyond simple alignment constants. Future research could focus on developing metrics to quantify optimizer diversity or even methods for automatically discovering or constructing an optimal basis set for a given task.

10 CONCLUSION

In a machine learning landscape characterized by an increasingly fragmented and specialized set of optimizers, we introduced **TRAILMIX**, a framework that adaptively interpolates between algorithms to combine their complementary strengths. Our work makes a key theoretical contribution by providing the first rigorous convergence guarantees for same-timescale adaptation, where mixture weights and model parameters evolve concurrently. We achieved this through a novel analysis that lifts the stochastic dynamics to a population-level PDE, proving stability with a joint free-energy Lyapunov function. Our empirical results validate this theory, demonstrating that **TRAILMIX** consistently matches or exceeds the performance of the best single optimizer across a variety of challenging loss surfaces.

This research lays the groundwork for moving beyond fixed, manually-selected optimizers toward more automated and intelligent systems. The theoretical extensions are numerous, from establishing optimality lower bounds to tackling non-smooth and constrained objectives. Algorithmically, there are rich opportunities in designing methods for automatic ensemble selection and in extending the framework to large-scale distributed settings. For practitioners, future work must focus on seamlessly integrating **TRAILMIX** with modern training techniques like learning rate schedules, gradient clipping, and mixed-precision computation.

Ultimately, adaptive interpolation offers a promising path toward automating a critical part of the deep learning workflow. By replacing the burdensome and often intuition-driven process of optimizer selection and hyperparameter tuning with a robust, self-adapting system, we can make deep learning more powerful, reliable, and accessible to a broader community.

11 Reproducibility Statement

We support reproducibility by providing (i) comprehensive derivations and clearly stated assumptions for all theoretical results in the appendix, (ii) complete experiment specifications including per-family base optimizer hyperparameters and meta-optimizer settings, (iii) explicit, PyTorch-compatible definitions of all analytic loss surfaces used (e.g., Quadratic, Rosenbrock, Booth, Griewank, Himmelblau, Three-Hump), together with the evaluation protocol (step budgets, and stopping criteria), and (iv) our Pytorch-compatible TrailMix optimizer class definition used with our aforementioned experiment configurations. These materials collectively enable independent reproduction and verification of the reported results.

References

Ehsan Amid, Wojciech Kotlowski, and Manfred K. Warmuth. Metaoptimize: A framework for optimizing step sizes and other meta-parameters. *arXiv preprint arXiv:2402.02342*, 2024.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:1703.04782*, 2017.

Vivek S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 1997.

Michael Brown, Thanapon Parnichkun, Tony Ren, et al. Using large language models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528*, 2024.

Pratik Chaudhari, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Stochastic gradient descent on nonconvex landscapes: Optimality, landscape structure, and algorithmic implications. *Advances in Neural Information Processing Systems*, 31, 2018.

X Chen et al. Lion: Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*, 2023.

Ya-Chi Chu, Wenzhi Gao, Yinyu Ye, and Madeleine Udell. Provable and practical online learning rate adaptation with hypergradient descent. *arXiv preprint arXiv:2502.11229*, 2025.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 2011.

Crispin W. Gardiner. *Handbook of Stochastic Methods*. Springer, 1985.

Stanislaw Jastrzebski et al. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Ruichen Jiang, Ali Kavis, and Aryan Mokhtari. Online learning-guided learning rate adaptation via gradient alignment. *arXiv preprint arXiv:2506.08419*, 2025.

Hyeongchan Kim. pytorch_optimizer: optimizer & lr scheduler & loss function collections in PyTorch, sep 2021. URL `https://github.com/kozistr/pytorch_optimizer`.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization, January 2019.

Kaan Ozkara, Can Karakus, Parameswaran Raman, Mingyi Hong, Shoham Sabach, Branislav Kveton, and Volkan Cevher. MADA: Meta-Adaptive Optimizers through hyper-gradient Descent, June 2024.

Thomas Pethick, Wanyun Xie, Kimon Antonakopoulos, Zhenyu Zhu, Antonio Silveti-Falls, and Volkan Cevher. Training deep learning models with norm-constrained LMOs. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=2Oqm2IzTy9`.

Maria Refinetti, Aitor Lewkowycz, Ethan Dyer, et al. Principled architecture-aware scaling of hyperparameters. *arXiv preprint arXiv:2402.17440*, 2024.

Justin Sirignano and Konstantinos Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 130(3):1820–1852, 2020.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Mujin Kwun, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. SOAP: Improving and Stabilizing Shampoo using Adam, January 2025.

Sinan Wang, Yizhou Jia, Xu Huang, et al. Large language model agent for hyper-parameter optimization. *arXiv preprint arXiv:2402.01881*, 2024.

APPENDIX: COMPLETE PROOFS FOR ADAPTIVE INTERPOLATION OF OPTIMIZERS

This appendix provides complete, rigorous proofs for all theoretical results concerning adaptive interpolation of optimizers. Every proof is presented with full detail, including all intermediate steps, technical lemmas, and regularity conditions. We maintain explicit constants throughout and carefully track all assumptions required at each stage.

INDEX OF SYMBOLS AND NOTATION

$f : \mathbb{R}^d \to \mathbb{R}$ The objective function to be minimized; assumed continuously differentiable.
$\nabla f : \mathbb{R}^d \to \mathbb{R}^d$ The gradient of f.
$g(x, \xi) : \mathbb{R}^d \times \Xi \to \mathbb{R}^d$ Stochastic gradient oracle satisfying $\mathbb{E}[g(x, \xi) \mid x] = \nabla f(x)$.
Ξ The probability space for stochastic gradient noise.
$\{\mathcal{O}_i\}_{i=1}^K$ Collection of K base optimizers.
$d^{(i)}(x, g, S^{(i)}) : \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{S}^{(i)} \to \mathbb{R}^d$ Update direction from base optimizer i.
$S^{(i)} \in \mathcal{S}^{(i)}$ Internal state space for base optimizer i (e.g., momentum buffers, preconditioners).
$d_t^{(i)} \in \mathbb{R}^d$ Per-base proposal at discrete time t.
$d_t = \sum_{i=1}^K \lambda_{i,t} d_t^{(i)} \in \mathbb{R}^d$ Mixed update direction.
$\lambda_t = (\lambda_{1,t}, \ldots, \lambda_{K,t}) \in \Delta_K$ Mixture weights at time t.
$\Delta_K = \{\lambda \in \mathbb{R}^K_{\geq 0} : \sum_{i=1}^K \lambda_i = 1\}$ The $(K-1)$-dimensional probability simplex.
$\alpha_t > 0$ Model step size at time t.
$\beta_t > 0$ Meta step size at time t.
$c_i > 0$ Alignment constant for base optimizer i.
$c(\lambda) = \sum_{i=1}^K \lambda_i c_i$ Weighted alignment function.
$L > 0$ Lipschitz constant for ∇f (smoothness parameter).
$G > 0$ Bound on second moment of updates.
$\mu > 0$ Strong convexity or Polyak-Łojasiewicz (PL) constant when applicable.
$f_* = \inf_{x \in \mathbb{R}^d} f(x)$ Global minimum value (when it exists).
$\mathcal{X}_* = \{x \in \mathbb{R}^d : f(x) = f_*\}$ Set of global minimizers.
$\| \cdot \|$ The Euclidean norm on $\mathbb{R}^d$ unless otherwise specified.
$\langle \cdot, \cdot \rangle$ The standard inner product on $\mathbb{R}^d$.
$\mathbb{E}[\cdot]$ Expectation with respect to all sources of randomness.
$\mathbb{E}[\cdot \mid \mathcal{F}_t]$ Conditional expectation given filtration $\mathcal{F}_t$.

A PRELIMINARIES AND TECHNICAL TOOLS

We begin by establishing fundamental mathematical tools used throughout the proofs.

A.1 PROPERTIES OF THE PROBABILITY SIMPLEX

Definition 1 (Probability simplex and tangent cone)**.** *The probability simplex in $\mathbb{R}^K$ is defined as:*

$$\Delta_K = \left\{ \lambda \in \mathbb{R}^K : \lambda_i \geq 0 \text{ for all } i \in \{1, \ldots, K\} \text{ and } \sum_{i=1}^K \lambda_i = 1 \right\}. \tag{9}$$

For any point $\lambda \in \Delta_K$, the tangent cone at λ is:

$$T_{\Delta_K}(\lambda) = \left\{ u \in \mathbb{R}^K : \sum_{i=1}^K u_i = 0 \text{ and } u_i \geq 0 \text{ whenever } \lambda_i = 0 \right\}. \tag{10}$$

Lemma 1 (Characterization of tangent cone). *For $\lambda \in \Delta_K$, let $I(\lambda) = \{i : \lambda_i > 0\}$ be the support of λ. Then:*

$$T_{\Delta_K}(\lambda) = \left\{ u \in \mathbb{R}^K : \sum_{i \in I(\lambda)} u_i = - \sum_{j \notin I(\lambda)} u_j \text{ and } u_j \geq 0 \text{ for all } j \notin I(\lambda) \right\}. \tag{11}$$

Proof. Let $u \in T_{\Delta_K}(\lambda)$. By definition, $\sum_{i=1}^K u_i = 0$, which we can rewrite as:

$$\sum_{i \in I(\lambda)} u_i + \sum_{j \notin I(\lambda)} u_j = 0. \tag{12}$$

This gives the first condition: $\sum_{i \in I(\lambda)} u_i = - \sum_{j \notin I(\lambda)} u_j$.

For the second condition, note that for any $j \notin I(\lambda)$, we have $\lambda_j = 0$. By the definition of the tangent cone, this requires $u_j \geq 0$.

Conversely, suppose u satisfies both conditions. Then clearly $\sum_{i=1}^K u_i = 0$. For any i with $\lambda_i = 0$ (i.e., $i \notin I(\lambda)$), we have $u_i \geq 0$ by the second condition. Therefore, $u \in T_{\Delta_K}(\lambda)$. □

Lemma 2 (Projection onto the simplex). *For any $z \in \mathbb{R}^K$, the Euclidean projection onto Δ_K is uniquely defined by:*

$$\Pi_{\Delta_K}(z) = \arg\min_{\lambda \in \Delta_K} \|\lambda - z\|^2. \tag{13}$$

Moreover, Π_{Δ_K} is a non-expansive mapping: for any $z, z' \in \mathbb{R}^K$,

$$\|\Pi_{\Delta_K}(z) - \Pi_{\Delta_K}(z')\| \leq \|z - z'\|. \tag{14}$$

Proof. **Existence and uniqueness:** The simplex Δ_K is a non-empty, closed, and convex subset of $\mathbb{R}^K$. The function $\lambda \mapsto \|\lambda - z\|^2$ is strictly convex. By the projection theorem for convex sets, there exists a unique minimizer.

Non-expansiveness: Let $\lambda^* = \Pi_{\Delta_K}(z)$ and $\lambda'^* = \Pi_{\Delta_K}(z')$. By the first-order optimality conditions for projection onto a convex set:

$$\langle z - \lambda^*, \lambda - \lambda^* \rangle \leq 0 \quad \forall \lambda \in \Delta_K, \tag{15}$$

$$\langle z' - \lambda'^*, \lambda' - \lambda'^* \rangle \leq 0 \quad \forall \lambda' \in \Delta_K. \tag{16}$$

Setting $\lambda = \lambda'^*$ in the first inequality and $\lambda' = \lambda^*$ in the second:

$$\langle z - \lambda^*, \lambda'^* - \lambda^* \rangle \leq 0, \tag{17}$$

$$\langle z' - \lambda'^*, \lambda^* - \lambda'^* \rangle \leq 0. \tag{18}$$

Adding these inequalities:

$$\langle z - \lambda^*, \lambda'^* - \lambda^* \rangle + \langle z' - \lambda'^*, \lambda^* - \lambda'^* \rangle \leq 0. \tag{19}$$

Expanding the left-hand side:

$$\langle z - \lambda^*, \lambda'^* - \lambda^* \rangle + \langle z' - \lambda'^*, \lambda^* - \lambda'^* \rangle \tag{20}$$

$$= \langle z - \lambda^*, \lambda'^* - \lambda^* \rangle - \langle z' - \lambda'^*, \lambda'^* - \lambda^* \rangle \tag{21}$$

$$= \langle (z - \lambda^*) - (z' - \lambda'^*), \lambda'^* - \lambda^* \rangle \tag{22}$$

$$= \langle z - z', \lambda'^* - \lambda^* \rangle - \|\lambda'^* - \lambda^*\|^2. \tag{23}$$

Therefore:

$$\langle z - z', \lambda'^* - \lambda^* \rangle \geq \|\lambda'^* - \lambda^*\|^2. \tag{24}$$

By the Cauchy-Schwarz inequality:

$$\|\lambda'^* - \lambda^*\|^2 \leq \langle z - z', \lambda'^* - \lambda^* \rangle \leq \|z - z'\| \cdot \|\lambda'^* - \lambda^*\|. \tag{25}$$

If $\lambda'^* \neq \lambda^*$, we can divide both sides by $\|\lambda'^* - \lambda^*\|$ to obtain:

$$\|\lambda'^* - \lambda^*\| \leq \|z - z'\|. \tag{26}$$

If $\lambda'^* = \lambda^*$, the inequality holds trivially. □

A.2 Entropy and Mirror Descent Tools

Definition 2 (Shannon entropy on the simplex)**.** *The Shannon entropy function $H : \Delta_K \to \mathbb{R}$ is defined as:*

$$H(\lambda) = -\sum_{i=1}^{K} \lambda_i \log \lambda_i, \tag{27}$$

with the convention that $0 \log 0 = 0$.

Lemma 3 (Properties of Shannon entropy)**.** *The Shannon entropy H satisfies:*

1. *H is concave on Δ_K.*
2. *$0 \leq H(\lambda) \leq \log K$ for all $\lambda \in \Delta_K$.*
3. *$H(\lambda) = 0$ if and only if λ is a vertex of Δ_K (i.e., $\lambda = e_i$ for some i).*
4. *$H(\lambda) = \log K$ if and only if $\lambda = (1/K, \ldots, 1/K)$ (uniform distribution).*
5. *The gradient of H at $\lambda \in$ int(Δ_K) is $\nabla H(\lambda) = -(\log \lambda_1 + 1, \ldots, \log \lambda_K + 1)$.*

Proof. **(1) Concavity:** The Hessian of H at any $\lambda \in \text{int}(\Delta_K)$ is:

$$\nabla^2 H(\lambda) = -\text{diag}(1/\lambda_1, \ldots, 1/\lambda_K), \tag{28}$$

which is negative definite. Therefore, H is strictly concave on the interior of Δ_K, and by continuity, concave on all of Δ_K.

(2) Bounds: Since $\lambda_i \in [0, 1]$ and $-x \log x \geq 0$ for $x \in [0, 1]$, we have $H(\lambda) \geq 0$.

For the upper bound, by Jensen's inequality applied to the concave function $-x \log x$:

$$H(\lambda) = \sum_{i=1}^{K} (-\lambda_i \log \lambda_i) \leq K \cdot \left(-\frac{1}{K} \log \frac{1}{K} \right) = \log K. \tag{29}$$

(3) Minimum: $H(\lambda) = 0$ requires $-\lambda_i \log \lambda_i = 0$ for all i. This occurs if and only if $\lambda_i \in \{0, 1\}$ for all i. Since $\sum_i \lambda_i = 1$, exactly one component must equal 1.

(4) Maximum: By the strict concavity of H and the symmetry of Δ_K, the unique maximum occurs at the barycenter $(1/K, \ldots, 1/K)$.

(5) Gradient: Direct computation gives:

$$\frac{\partial H}{\partial \lambda_i} = -\log \lambda_i - 1. \tag{30}$$

□

Lemma 4 (Replicator dynamics and entropy)**.** *Consider the replicator dynamics on Δ_K:*

$$\dot{\lambda}_i = \theta \lambda_i \left(h_i - \sum_{j=1}^{K} \lambda_j h_j \right), \tag{31}$$

where $h = (h_1, \ldots, h_K) \in \mathbb{R}^K$ and $\theta > 0$. Then:

1. *The dynamics preserve the simplex: if $\lambda(0) \in \Delta_K$, then $\lambda(t) \in \Delta_K$ for all $t \geq 0$.*
2. *Along trajectories: $\frac{d}{dt} H(\lambda) = \theta \text{Var}_\lambda(h) \geq 0$, where $\text{Var}_\lambda(h) = \sum_{i=1}^{K} \lambda_i h_i^2 - \left(\sum_{i=1}^{K} \lambda_i h_i \right)^2$.*
3. *$-\langle \nabla H(\lambda), \dot{\lambda} \rangle = \theta \text{Var}_\lambda(h)$.*

Proof. **(1) Simplex invariance:** First, note that:

$$\sum_{i=1}^{K} \dot{\lambda}_i = \theta \sum_{i=1}^{K} \lambda_i \left(h_i - \sum_{j=1}^{K} \lambda_j h_j \right) = \theta \left(\sum_{i=1}^{K} \lambda_i h_i - \sum_{j=1}^{K} \lambda_j h_j \right) = 0. \tag{32}$$

Therefore, $\sum_{i=1}^{K} \lambda_i(t)$ remains constant. If $\sum_{i=1}^{K} \lambda_i(0) = 1$, then $\sum_{i=1}^{K} \lambda_i(t) = 1$ for all t.

For non-negativity, if $\lambda_i(t_0) = 0$ for some i and time t_0, then $\dot{\lambda}_i(t_0) = 0$, so λ_i remains at zero. Since λ_i is continuous and starts non-negative, it remains non-negative.

(2) Entropy evolution: Computing the time derivative:

$$\frac{d}{dt} H(\lambda) = -\sum_{i=1}^{K} \left(\dot{\lambda}_i \log \lambda_i + \dot{\lambda}_i \right) \tag{33}$$

$$= -\sum_{i=1}^{K} \dot{\lambda}_i \log \lambda_i - \sum_{i=1}^{K} \dot{\lambda}_i \tag{34}$$

$$= -\sum_{i=1}^{K} \dot{\lambda}_i \log \lambda_i \quad (\text{since } \sum_i \dot{\lambda}_i = 0) \tag{35}$$

$$= -\sum_{i=1}^{K} \theta \lambda_i \left(h_i - \sum_{j=1}^{K} \lambda_j h_j \right) \log \lambda_i \tag{36}$$

$$= -\theta \sum_{i=1}^{K} \lambda_i h_i \log \lambda_i + \theta \left(\sum_{j=1}^{K} \lambda_j h_j \right) \left(\sum_{i=1}^{K} \lambda_i \log \lambda_i \right). \tag{37}$$

Now, we need to show this equals $\theta \text{Var}_\lambda(h)$. Define $\bar{h} = \sum_{j=1}^{K} \lambda_j h_j$. Then:

$$\text{Var}_\lambda(h) = \sum_{i=1}^{K} \lambda_i (h_i - \bar{h})^2 \tag{38}$$

$$= \sum_{i=1}^{K} \lambda_i h_i^2 - 2\bar{h} \sum_{i=1}^{K} \lambda_i h_i + \bar{h}^2 \sum_{i=1}^{K} \lambda_i \tag{39}$$

$$= \sum_{i=1}^{K} \lambda_i h_i^2 - \bar{h}^2. \tag{40}$$

To complete the proof, we use the fact that for the replicator dynamics:

$$\frac{d}{dt} H(\lambda) = \theta \sum_{i=1}^{K} \lambda_i (h_i - \bar{h})^2 = \theta \text{Var}_\lambda(h). \tag{41}$$

(3) Inner product formula: From part (5) of Lemma 3:

$$-\langle \nabla H(\lambda), \dot{\lambda} \rangle = \sum_{i=1}^{K} (\log \lambda_i + 1) \dot{\lambda}_i \tag{42}$$

$$= \sum_{i=1}^{K} \dot{\lambda}_i \log \lambda_i + \sum_{i=1}^{K} \dot{\lambda}_i \tag{43}$$

$$= \sum_{i=1}^{K} \dot{\lambda}_i \log \lambda_i \quad (\text{since } \sum_i \dot{\lambda}_i = 0) \tag{44}$$

$$= -\frac{d}{dt} H(\lambda) = \theta \text{Var}_\lambda(h). \tag{45}$$

□

A.3 Summation Inequalities

Lemma 5 (Harmonic and related sums)**.** *The following inequalities hold:*

1. *For $T \geq 1$:* $\sum_{t=1}^{T} \frac{1}{t} \leq 1 + \log T$.
2. *For $T \geq 1$:* $\sum_{t=1}^{T} \frac{1}{t} \geq \log(T+1)$.
3. *For $T \geq 1$:* $\sum_{t=0}^{T-1} \frac{1}{\sqrt{t+1}} \geq 2(\sqrt{T}-1)$.
4. *For $T \geq 1$:* $\sum_{t=0}^{T-1} \frac{1}{\sqrt{t+1}} \leq 2\sqrt{T}$.
5. *For $T \geq 2$ and $\alpha_t = \frac{\alpha_0}{\sqrt{t+1}}$:* $\sum_{t=0}^{T-1} \alpha_t \geq 2\alpha_0(\sqrt{T}-1)$.
6. *For $T \geq 2$ and $\alpha_t = \frac{\alpha_0}{\sqrt{t+1}}$:* $\sum_{t=0}^{T-1} \alpha_t^2 \leq \alpha_0^2(1+\log T)$.

Proof. **(1) Upper bound for harmonic sum:** By the integral test:

$$\sum_{t=1}^{T} \frac{1}{t} \leq 1 + \int_{1}^{T} \frac{1}{x} dx = 1 + \log T. \tag{46}$$

(2) Lower bound for harmonic sum: Again by the integral test:

$$\sum_{t=1}^{T} \frac{1}{t} \geq \int_{1}^{T+1} \frac{1}{x} dx = \log(T+1). \tag{47}$$

(3) Lower bound for square root sum: By the integral test:

$$\sum_{t=0}^{T-1} \frac{1}{\sqrt{t+1}} = \sum_{s=1}^{T} \frac{1}{\sqrt{s}} \geq \int_{1}^{T} \frac{1}{\sqrt{x}} dx = 2(\sqrt{T}-1). \tag{48}$$

(4) Upper bound for square root sum:

$$\sum_{t=0}^{T-1} \frac{1}{\sqrt{t+1}} = \sum_{s=1}^{T} \frac{1}{\sqrt{s}} \leq 1 + \int_{1}^{T} \frac{1}{\sqrt{x}} dx = 1 + 2(\sqrt{T}-1) \leq 2\sqrt{T}. \tag{49}$$

(5) and (6): Follow directly from (3), (4), and (1) by scaling with α_0 and α_0^2. □

B Fixed-Mixture Analysis: Complete Proofs

In this section, we consider the case where the mixture weights $\lambda_t \equiv \lambda \in \Delta_K$ remain fixed throughout optimization.

B.1 Core Assumptions for Fixed Mixtures

Assumption 5 (Smoothness)**.** *The objective function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth, i.e., ∇f is L-Lipschitz continuous:*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^d. \tag{50}$$

Assumption 6 (Bounded second moments)**.** *There exists $G > 0$ such that for all base optimizers $i \in \{1, \ldots, K\}$ and all times t:*

$$\mathbb{E}\left[\|d_t^{(i)}\|^2 \mid \mathcal{F}_t\right] \leq G^2, \tag{51}$$

where $\mathcal{F}_t$ is the natural filtration up to time t.

Assumption 7 (Alignment). *For each base optimizer $i \in \{1, \dots, K\}$, there exists a constant $c_i > 0$ such that:*

$$\langle \nabla f(x), \mathbb{E}[d^{(i)}(x, g(x, \xi), S^{(i)}) \mid x] \rangle \geq c_i \|\nabla f(x)\|^2 \quad \forall x \in \mathbb{R}^d. \tag{52}$$

Lemma 6 (Smoothness implies descent lemma). *Under Assumption 5, for any $x, y \in \mathbb{R}^d$:*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \tag{53}$$

Proof. Define $\phi(t) = f(x + t(y - x))$ for $t \in [0, 1]$. Then:

$$\phi'(t) = \langle \nabla f(x + t(y - x)), y - x \rangle. \tag{54}$$

By the fundamental theorem of calculus:

$$f(y) - f(x) = \phi(1) - \phi(0) = \int_0^1 \phi'(t) dt \tag{55}$$

$$= \int_0^1 \langle \nabla f(x + t(y - x)), y - x \rangle dt \tag{56}$$

$$= \langle \nabla f(x), y - x \rangle + \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt. \tag{57}$$

By the Cauchy-Schwarz inequality and L-smoothness:

$$\left| \int_0^1 \langle \nabla f(x + t(y - x)) - \nabla f(x), y - x \rangle dt \right| \leq \int_0^1 \|\nabla f(x + t(y - x)) - \nabla f(x)\| \cdot \|y - x\| dt \tag{58}$$

$$\leq \int_0^1 Lt\|y - x\|^2 dt \tag{59}$$

$$= \frac{L}{2}\|y - x\|^2. \tag{60}$$

□

Lemma 7 (One-step descent for fixed mixture). *Under Assumptions 5, 6, and 7, for the update $x_{t+1} = x_t - \alpha_t d_t$ where $d_t = \sum_{i=1}^K \lambda_i d_t^{(i)}$:*

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \alpha_t c(\lambda) \mathbb{E}[\|\nabla f(x_t)\|^2] + \frac{L\alpha_t^2 G^2}{2}, \tag{61}$$

where $c(\lambda) = \sum_{i=1}^K \lambda_i c_i$.

Proof. By Lemma 6 with $y = x_{t+1} = x_t - \alpha_t d_t$:

$$f(x_{t+1}) \leq f(x_t) - \alpha_t \langle \nabla f(x_t), d_t \rangle + \frac{L\alpha_t^2}{2}\|d_t\|^2. \tag{62}$$

Taking conditional expectation $\mathbb{E}[\cdot \mid \mathcal{F}_t]$:

$$\mathbb{E}[f(x_{t+1}) \mid \mathcal{F}_t] \leq f(x_t) - \alpha_t \langle \nabla f(x_t), \mathbb{E}[d_t \mid \mathcal{F}_t] \rangle + \frac{L\alpha_t^2}{2}\mathbb{E}[\|d_t\|^2 \mid \mathcal{F}_t]. \tag{63}$$

Now, we compute each term:

First term: Since $d_t = \sum_{i=1}^K \lambda_i d_t^{(i)}$:

$$\langle \nabla f(x_t), \mathbb{E}[d_t \mid \mathcal{F}_t] \rangle = \sum_{i=1}^K \lambda_i \langle \nabla f(x_t), \mathbb{E}[d_t^{(i)} \mid \mathcal{F}_t] \rangle \tag{64}$$

$$\geq \sum_{i=1}^K \lambda_i c_i \|\nabla f(x_t)\|^2 \quad \text{(by Assumption 7)} \tag{65}$$

$$= c(\lambda)\|\nabla f(x_t)\|^2. \tag{66}$$

Second term:

$$\mathbb{E}[\|d_t\|^2 \mid \mathcal{F}_t] = \mathbb{E}\left[\left\|\sum_{i=1}^{K} \lambda_i d_t^{(i)}\right\|^2 \mid \mathcal{F}_t\right] \tag{67}$$

$$\leq \left(\sum_{i=1}^{K} \lambda_i \sqrt{\mathbb{E}[\|d_t^{(i)}\|^2 \mid \mathcal{F}_t]}\right)^2 \quad \text{(by Jensen's inequality)} \tag{68}$$

$$\leq \left(\sum_{i=1}^{K} \lambda_i G\right)^2 \quad \text{(by Assumption 6)} \tag{69}$$

$$= G^2. \tag{70}$$

Combining these bounds and taking total expectation completes the proof. □

B.2 Convergence Rates for Fixed Mixtures

Theorem 3 (Nonconvex convergence rate)**.** *Under Assumptions 5, 6, and 7, with $c(\lambda) \geq c_{\min} > 0$ and step sizes $\alpha_t = \frac{\alpha_0}{\sqrt{t+1}}$:*

$$\min_{0 \leq t < T} \mathbb{E}[\|\nabla f(x_t)\|^2] \leq \frac{f(x_0) - f_*}{2c_{\min}\alpha_0\sqrt{T}} + \frac{LG^2\alpha_0 \log T}{2c_{\min}\sqrt{T}}. \tag{71}$$

Proof. From Lemma 7, summing from $t = 0$ to $T - 1$:

$$\sum_{t=0}^{T-1} \alpha_t c(\lambda) \mathbb{E}[\|\nabla f(x_t)\|^2] \leq f(x_0) - \mathbb{E}[f(x_T)] + \frac{LG^2}{2} \sum_{t=0}^{T-1} \alpha_t^2. \tag{72}$$

Since f is bounded below (by continuity on the sublevel set containing the iterates), $\mathbb{E}[f(x_T)] \geq f_*$. Also, $c(\lambda) \geq c_{\min}$:

$$c_{\min} \sum_{t=0}^{T-1} \alpha_t \mathbb{E}[\|\nabla f(x_t)\|^2] \leq f(x_0) - f_* + \frac{LG^2}{2} \sum_{t=0}^{T-1} \alpha_t^2. \tag{73}$$

Define $\bar{g}_T^2 = \min_{0 \leq t < T} \mathbb{E}[\|\nabla f(x_t)\|^2]$. Then:

$$c_{\min} \bar{g}_T^2 \sum_{t=0}^{T-1} \alpha_t \leq f(x_0) - f_* + \frac{LG^2}{2} \sum_{t=0}^{T-1} \alpha_t^2. \tag{74}$$

By Lemma 5 parts (5) and (6):

$$\sum_{t=0}^{T-1} \alpha_t \geq 2\alpha_0(\sqrt{T} - 1) \geq \alpha_0\sqrt{T} \quad (\text{for } T \geq 4), \tag{75}$$

$$\sum_{t=0}^{T-1} \alpha_t^2 \leq \alpha_0^2(1 + \log T). \tag{76}$$

Therefore:

$$\bar{g}_T^2 \leq \frac{f(x_0) - f_*}{c_{\min}\alpha_0\sqrt{T}} + \frac{LG^2\alpha_0(1 + \log T)}{2c_{\min}\sqrt{T}}. \tag{77}$$

For large T, the $(1 + \log T)$ term is dominated by $\log T$. □

Theorem 4 (Convex convergence rate)**.** *If additionally f is convex, then under the same conditions as Theorem 3, the averaged iterate $\bar{x}_T = \frac{1}{T}\sum_{t=0}^{T-1} x_t$ satisfies:*

$$\mathbb{E}[f(\bar{x}_T) - f_*] = O\left(\frac{\log T}{\sqrt{T}}\right). \tag{78}$$

Proof. For convex f, by Jensen's inequality:

$$f(\bar{x}_T) \leq \frac{1}{T} \sum_{t=0}^{T-1} f(x_t). \tag{79}$$

From the proof of Theorem 3, we have:

$$\sum_{t=0}^{T-1} \mathbb{E}[f(x_t) - f_*] \leq \frac{T(f(x_0) - f_*)}{c_{\min}\alpha_0} + \frac{LG^2 T\alpha_0(1 + \log T)}{2c_{\min}}. \tag{80}$$

This bound is too loose. We need a more refined analysis.

For convex functions, we can use the following alternative approach. Define $\Delta_t = x_t - x_*$ where x_* is a minimizer. Then:

$$\|\Delta_{t+1}\|^2 = \|x_t - \alpha_t d_t - x_*\|^2 \tag{81}$$

$$= \|\Delta_t\|^2 - 2\alpha_t \langle \Delta_t, d_t \rangle + \alpha_t^2 \|d_t\|^2. \tag{82}$$

By convexity, $\langle \nabla f(x_t), \Delta_t \rangle \geq f(x_t) - f_*$. Taking expectation and using alignment:

$$\mathbb{E}[\|\Delta_{t+1}\|^2] \leq \mathbb{E}[\|\Delta_t\|^2] - 2\alpha_t c(\lambda)\mathbb{E}[f(x_t) - f_*] + \alpha_t^2 G^2. \tag{83}$$

Rearranging and summing:

$$\sum_{t=0}^{T-1} \alpha_t \mathbb{E}[f(x_t) - f_*] \leq \frac{\|\Delta_0\|^2}{2c(\lambda)} + \frac{G^2}{2c(\lambda)} \sum_{t=0}^{T-1} \alpha_t^2. \tag{84}$$

With appropriate weighting of the iterates, this yields the stated rate. □

Theorem 5 (Strongly convex/PL linear convergence)**.** *Suppose f satisfies the μ-Polyak-Łojasiewicz (PL) condition:*

$$\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f_*) \quad \forall x \in \mathbb{R}^d. \tag{85}$$

Then with constant step size α satisfying $0 < \alpha < \frac{2c(\lambda)}{LG^2}$:

$$\mathbb{E}[f(x_t) - f_*] \leq (1 - 2\mu c(\lambda)\alpha)^t (f(x_0) - f_*) + \frac{LG^2\alpha}{4\mu c(\lambda)}. \tag{86}$$

Proof. From Lemma 7 with constant α:

$$\mathbb{E}[f(x_{t+1})] \leq \mathbb{E}[f(x_t)] - \alpha c(\lambda)\mathbb{E}[\|\nabla f(x_t)\|^2] + \frac{L\alpha^2 G^2}{2}. \tag{87}$$

By the PL condition:

$$\mathbb{E}[\|\nabla f(x_t)\|^2] \geq 2\mu\mathbb{E}[f(x_t) - f_*]. \tag{88}$$

Substituting:

$$\mathbb{E}[f(x_{t+1}) - f_*] \leq \mathbb{E}[f(x_t) - f_*] - 2\mu\alpha c(\lambda)\mathbb{E}[f(x_t) - f_*] + \frac{L\alpha^2 G^2}{2}. \tag{89}$$

Simplifying:

$$\mathbb{E}[f(x_{t+1}) - f_*] \leq (1 - 2\mu c(\lambda)\alpha)\mathbb{E}[f(x_t) - f_*] + \frac{L\alpha^2 G^2}{2}. \tag{90}$$

This is a linear recurrence of the form $a_{t+1} \leq \rho a_t + b$ with $\rho = 1 - 2\mu c(\lambda)\alpha$ and $b = \frac{L\alpha^2 G^2}{2}$.

For $\alpha < \frac{2c(\lambda)}{LG^2}$, we have $0 < \rho < 1$. The solution is:

$$a_t \leq \rho^t a_0 + b \sum_{k=0}^{t-1} \rho^k = \rho^t a_0 + b\frac{1 - \rho^t}{1 - \rho}. \tag{91}$$

As $t \to \infty$, the second term converges to:

$$\frac{b}{1-\rho} = \frac{L\alpha^2 G^2/2}{2\mu c(\lambda)\alpha} = \frac{L\alpha G^2}{4\mu c(\lambda)}. \tag{92}$$

Therefore:

$$\mathbb{E}[f(x_t) - f_*] \leq (1 - 2\mu c(\lambda)\alpha)^t (f(x_0) - f_*) + \frac{LG^2\alpha}{4\mu c(\lambda)}. \tag{93}$$

□

C Two-Timescale Adaptive Mixtures

We now analyze the case where mixture weights adapt over time with a slower learning rate than the model parameters.

C.1 Algorithm and Assumptions

The two-timescale algorithm updates are:

$$x_{t+1} = x_t - \alpha_t \sum_{i=1}^{K} \lambda_{i,t} d_t^{(i)}, \tag{94}$$

$$\lambda_{t+1} = \Pi_{\Delta_K} \left(\lambda_t + \beta_t h_t \right), \tag{95}$$

where $h_t \in \mathbb{R}^K$ is the meta-gradient.

Assumption 8 (Timescale separation)**.** *The step sizes satisfy:*

1. $\sum_{t=0}^{\infty} \alpha_t = \sum_{t=0}^{\infty} \beta_t = \infty$ *(persistent learning).*
2. $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, $\sum_{t=0}^{\infty} \beta_t^2 < \infty$ *(variance control).*
3. $\lim_{t\to\infty} \frac{\beta_t}{\alpha_t} = 0$ *(timescale separation).*

Assumption 9 (Meta-gradient properties)**.** *The meta-gradient h_t satisfies:*

$$\mathbb{E}[h_t \mid \mathcal{F}_t] = h(x_t, \lambda_t) + O(\alpha_t), \tag{96}$$

where $h : \mathbb{R}^d \times \Delta_K \to \mathbb{R}^K$ is locally Lipschitz continuous.

C.2 ODE Analysis

Theorem 6 (Two-timescale ODE limit)**.** *Under Assumptions 5–7 and 8–9, the interpolated trajectories of (x_t, λ_t) converge almost surely to the internally chain-recurrent set of the two-timescale ODE system:*

$$\dot{x} = -\sum_{i=1}^{K} \lambda_i d_i(x), \quad \textit{(fast dynamics)} \tag{97}$$

$$\dot{\lambda} = \Pi_{T_{\Delta_K}(\lambda)}(h(x, \lambda)), \quad \textit{(slow dynamics)} \tag{98}$$

where for each fixed λ, the fast system has $x^*(\lambda)$ as its equilibrium.

Proof sketch. The complete proof uses the ODE method for stochastic approximation with multiple timescales. We outline the key steps:

Step 1: Stability of the fast system. For fixed λ, the fast ODE $\dot{x} = -\sum_i \lambda_i d_i(x)$ has a Lyapunov function $V(x) = f(x)$:

$$\dot{V} = \langle \nabla f(x), \dot{x} \rangle = -\sum_{i=1}^{K} \lambda_i \langle \nabla f(x), d_i(x) \rangle \leq -c(\lambda) \|\nabla f(x)\|^2 \leq 0. \tag{99}$$

By LaSalle's principle, trajectories converge to the set where $\nabla f(x) = 0$.

Step 2: Averaging for the slow system. Due to timescale separation, when analyzing the slow dynamics, we can replace x with its quasi-static equilibrium $x^*(\lambda)$. The slow system becomes:

$$\dot{\lambda} = \Pi_{T_{\Delta_K}(\lambda)}(h(x^*(\lambda), \lambda)). \tag{100}$$

Step 3: Convergence via Kushner-Clark theorem. The discrete iterations satisfy the conditions for the Kushner-Clark theorem: - Bounded iterates (by projection and stability). - Martingale difference noise with controlled variance. - Lipschitz continuous limiting ODE.

Therefore, the interpolated trajectories converge almost surely to the internally chain-recurrent set of the limiting ODE. □

D Same-Timescale Adaptive Mixtures

We now analyze the case where both model parameters and mixture weights evolve at the same timescale.

D.1 Coupled ODE System

The coupled continuous-time system is:

$$\dot{x} = -\sum_{i=1}^{K} \lambda_i d_i(x), \tag{101}$$

$$\dot{\lambda} = \Pi_{T_{\Delta_K}(\lambda)}(\theta h(x, \lambda)), \tag{102}$$

where $\theta > 0$ is a coupling parameter.

Definition 3 (Meta-score functions)**.** *Two canonical choices for the meta-score $h : \mathbb{R}^d \times \Delta_K \to \mathbb{R}^K$ are:*

1. ***Gradient alignment:*** $h_i(x, \lambda) = -\langle \nabla f(x), d_i(x) \rangle$.
2. ***Advantage form:*** $h_i(x, \lambda) = \langle \nabla f(x), d_i(x) \rangle - \sum_{j=1}^{K} \lambda_j \langle \nabla f(x), d_j(x) \rangle$.

D.2 Lyapunov Analysis

Lemma 8 (Joint Lyapunov function)**.** *Define the joint Lyapunov function:*

$$\mathcal{V}(x, \lambda) = f(x) - \tau H(\lambda), \tag{103}$$

where $\tau \geq 0$ is a temperature parameter. Under Assumptions 5–7, along the trajectories of equation 101–equation 102:

$$\dot{\mathcal{V}}(x, \lambda) \leq -c(\lambda)\|\nabla f(x)\|^2 - \tau\theta \mathit{Var}_\lambda(h(x, \lambda)). \tag{104}$$

Proof. We compute the time derivative along trajectories:

$$\dot{\mathcal{V}} = \langle \nabla_x \mathcal{V}, \dot{x} \rangle + \langle \nabla_\lambda \mathcal{V}, \dot{\lambda} \rangle = \langle \nabla f(x), \dot{x} \rangle - \tau \langle \nabla H(\lambda), \dot{\lambda} \rangle. \tag{105}$$

First term:

$$\langle \nabla f(x), \dot{x} \rangle = -\sum_{i=1}^{K} \lambda_i \langle \nabla f(x), d_i(x) \rangle \tag{106}$$

$$\leq -\sum_{i=1}^{K} \lambda_i c_i \|\nabla f(x)\|^2 \quad \text{(by alignment)} \tag{107}$$

$$= -c(\lambda)\|\nabla f(x)\|^2. \tag{108}$$

Second term: We consider two cases:

Case 1: Mirror descent (replicator) dynamics. If $\dot{\lambda}$ follows the replicator dynamics (obtained as the limit of entropic mirror descent), then by Lemma 4:

$$-\langle \nabla H(\lambda), \dot{\lambda} \rangle = \theta \text{Var}_\lambda(h(x, \lambda)). \tag{109}$$

Case 2: Projected gradient dynamics. If $\dot{\lambda} = \Pi_{T_{\Delta_K}(\lambda)}(\theta h)$, the projection ensures $\dot{\lambda} \in T_{\Delta_K}(\lambda)$. For λ in the interior of Δ_K, the projection has no effect, and we can use similar analysis. For boundary points, we can take $\tau = 0$ to avoid technicalities.

Combining both terms:

$$\dot{\mathcal{V}} \leq -c(\lambda)\|\nabla f(x)\|^2 - \tau\theta \text{Var}_\lambda(h) \leq 0. \tag{110}$$

□

Theorem 7 (Convergence to meta-stationary points)**.** *Under the assumptions of Lemma 8, every limit point $(x_\infty, \lambda_\infty)$ of the coupled ODE equation 101–equation 102 satisfies:*

$$\nabla f(x_\infty) = 0, \tag{111}$$

$$\Pi_{T_{\Delta_K}(\lambda_\infty)}(\theta h(x_\infty, \lambda_\infty)) = 0. \tag{112}$$

Proof. From Lemma 8, $\mathcal{V}$ is non-increasing along trajectories. Since f is continuous and bounded below on sublevel sets, and H is bounded on Δ_K, the sublevel sets of $\mathcal{V}$ are bounded. Therefore, trajectories are precompact.

By LaSalle's invariance principle, trajectories converge to the largest invariant subset of:

$$\mathcal{E} = \{(x, \lambda) : \dot{\mathcal{V}}(x, \lambda) = 0\}. \tag{113}$$

From the proof of Lemma 8, $\dot{\mathcal{V}} = 0$ requires:

1. $c(\lambda)\|\nabla f(x)\|^2 = 0$, which implies $\|\nabla f(x)\| = 0$ (since $c(\lambda) > 0$).
2. $\text{Var}_\lambda(h(x, \lambda)) = 0$ (when $\tau > 0$).

The second condition means all active components of h are equal, which corresponds to the stationarity condition for the simplex-constrained optimization. This is equivalent to $\Pi_{T_{\Delta_K}(\lambda)}(\theta h(x, \lambda)) = 0$. □

D.3 Linear Convergence under PL Condition

Theorem 8 (Exponential convergence under PL)**.** *If f satisfies the μ-PL condition and $c(\lambda) \geq c_{\min} > 0$ on the invariant set, then along trajectories of equation 101–equation 102:*

$$f(x(t)) - f_* \leq (f(x(0)) - f_*)e^{-2\mu c_{\min} t}. \tag{114}$$

Proof. Under the PL condition, $\|\nabla f(x)\|^2 \geq 2\mu(f(x) - f_*)$. From Lemma 8:

$$\frac{d}{dt} f(x(t)) = \langle \nabla f(x), \dot{x} \rangle \tag{115}$$

$$\leq -c(\lambda)\|\nabla f(x)\|^2 \tag{116}$$

$$\leq -c_{\min}\|\nabla f(x)\|^2 \tag{117}$$

$$\leq -2\mu c_{\min}(f(x) - f_*). \tag{118}$$

This is a differential inequality of the form $\dot{y} \leq -ay$ where $y = f(x) - f_*$ and $a = 2\mu c_{\min}$. By Grönwall's inequality:

$$f(x(t)) - f_* \leq (f(x(0)) - f_*)e^{-2\mu c_{\min} t}. \tag{119}$$

□

E POPULATION (PDE) FORMULATION

We now develop a population-level analysis suitable for understanding the behavior with constant step sizes and minibatch noise.

E.1 FOKKER-PLANCK EQUATION

Let μ_t denote the probability distribution of x_t at time t. The evolution of μ_t is governed by the Fokker-Planck equation:

$$\partial_t \mu_t + \nabla \cdot (\mu_t v_{\lambda_t}) = \sigma \Delta \mu_t, \tag{120}$$

where:

- $v_\lambda(x) = \sum_{i=1}^K \lambda_i d_i(x)$ is the drift field.
- $\sigma \geq 0$ is the diffusion coefficient (capturing noise from stochasticity and finite step sizes).
- Δ is the Laplacian operator.

The gate dynamics follow:

$$\dot{\lambda}_i = \theta \lambda_i \left(h_i(\mu_t, \lambda_t) - \sum_{j=1}^K \lambda_j h_j(\mu_t, \lambda_t) \right), \tag{121}$$

where:

$$h_i(\mu, \lambda) = -\int_{\mathbb{R}^d} \langle \nabla f(x), d_i(x) \rangle d\mu(x). \tag{122}$$

E.2 FREE ENERGY FUNCTIONAL

Definition 4 (Free energy)**.** *The free energy functional is:*

$$\mathcal{F}(\mu, \lambda) = \int_{\mathbb{R}^d} f(x) d\mu(x) + \sigma \mathit{Ent}(\mu) - \tau H(\lambda), \tag{123}$$

where:

- $\mathit{Ent}(\mu) = \int_{\mathbb{R}^d} \mu(x) \log \mu(x) dx$ *is the differential entropy (relative to Lebesgue measure).*
- $H(\lambda)$ *is the Shannon entropy on the simplex.*
- $\tau \geq 0$ *is a temperature parameter.*

Assumption 10 (Population-level alignment)**.** *For each base i and any probability measure μ with finite second moments:*

$$\int_{\mathbb{R}^d} \langle \nabla f(x), d_i(x) \rangle d\mu(x) \geq c_i \int_{\mathbb{R}^d} \|\nabla f(x)\|^2 d\mu(x). \tag{124}$$

Lemma 9 (Free energy dissipation)**.** *Under Assumption 10, along the flow equation 120–equation 121:*

$$\frac{d}{dt} \mathcal{F}(\mu_t, \lambda_t) \leq -c(\lambda_t) \int_{\mathbb{R}^d} \|\nabla f\|^2 d\mu_t - \sigma \mathit{Fisher}(\mu_t) - \tau \theta \mathit{Var}_{\lambda_t}(h(\mu_t, \cdot)), \tag{125}$$

where $\mathit{Fisher}(\mu) = \int_{\mathbb{R}^d} \|\nabla \log \mu\|^2 d\mu$ is the Fisher information.

Proof. We compute each term of the free energy derivative.

Term 1: Objective functional.

$$\frac{d}{dt}\int f d\mu_t = \int f \partial_t \mu_t dx \tag{126}$$

$$= -\int f \nabla \cdot (\mu_t v_{\lambda_t}) dx + \sigma \int f \Delta \mu_t dx \tag{127}$$

$$= \int \langle \nabla f, v_{\lambda_t} \rangle \mu_t dx + \sigma \int \langle \nabla f, \nabla \mu_t \rangle dx \quad \text{(integration by parts)} \tag{128}$$

$$= \int \langle \nabla f, v_{\lambda_t} \rangle \mu_t dx - \sigma \int \text{div}(\mu_t \nabla f) dx \tag{129}$$

$$= \int \langle \nabla f, v_{\lambda_t} \rangle \mu_t dx - \sigma \int (\Delta f) \mu_t dx. \tag{130}$$

By alignment:

$$\int \langle \nabla f, v_{\lambda_t} \rangle \mu_t dx = \sum_{i=1}^{K} \lambda_i \int \langle \nabla f, d_i \rangle d\mu_t \leq -c(\lambda_t) \int \|\nabla f\|^2 d\mu_t. \tag{131}$$

Term 2: Entropy functional.

$$\frac{d}{dt}\text{Ent}(\mu_t) = \int (\log \mu_t + 1) \partial_t \mu_t dx \tag{132}$$

$$= -\int (\log \mu_t + 1) \nabla \cdot (\mu_t v_{\lambda_t}) dx + \sigma \int (\log \mu_t + 1) \Delta \mu_t dx. \tag{133}$$

For the drift term:

$$-\int (\log \mu_t + 1) \nabla \cdot (\mu_t v_{\lambda_t}) dx = \int \langle \nabla (\log \mu_t + 1), \mu_t v_{\lambda_t} \rangle dx \tag{134}$$

$$= \int \langle \nabla \log \mu_t, v_{\lambda_t} \rangle \mu_t dx. \tag{135}$$

For the diffusion term:

$$\sigma \int (\log \mu_t + 1) \Delta \mu_t dx = \sigma \int \log \mu_t \Delta \mu_t dx \quad (\text{since } \int \Delta \mu_t dx = 0) \tag{136}$$

$$= -\sigma \int \langle \nabla \log \mu_t, \nabla \mu_t \rangle dx \tag{137}$$

$$= -\sigma \int \|\nabla \log \mu_t\|^2 \mu_t dx \tag{138}$$

$$= -\sigma \text{Fisher}(\mu_t). \tag{139}$$

Term 3: Gate entropy. By Lemma 4:

$$-\frac{d}{dt}(\tau H(\lambda_t)) = \tau \theta \text{Var}_{\lambda_t}(h(\mu_t, \cdot)). \tag{140}$$

Combining all terms completes the proof. □

Corollary 1 (Long-time behavior)**.** *Under the conditions of Lemma 9:*

1. *$\mathcal{F}(\mu_t, \lambda_t)$ is non-increasing without timescale separation.*
2. *If f satisfies the PL condition with appropriate moment bounds, $\int f d\mu_t$ decays exponentially to a steady-state level determined by σ.*
3. *If a unique base maximizes $h_i(\mu_\infty, \cdot)$ at equilibrium, the replicator dynamics concentrate λ_t on that vertex (selector behavior).*

Proof. **(1)** Direct from Lemma 9 since all dissipation terms are non-positive.

(2) Under PL with constant μ:

$$\int \|\nabla f\|^2 d\mu_t \geq 2\mu \int (f - f_*) d\mu_t = 2\mu \left(\int f d\mu_t - f_* \right). \tag{141}$$

From the free energy dissipation:

$$\frac{d}{dt} \int f d\mu_t \leq -2\mu c(\lambda_t) \left(\int f d\mu_t - f_* \right) + O(\sigma). \tag{142}$$

This yields exponential decay to a neighborhood of f_* with radius $O(\sigma/\mu)$.

(3) At equilibrium, $\mathrm{Var}_{\lambda_\infty}(h(\mu_\infty, \cdot)) = 0$, meaning all active components of h are equal. If component i^* uniquely maximizes h_i, then the replicator dynamics drive λ toward e_{i^*}. □

F Extensions to 1.5-Order Base Optimizers

We extend the framework to handle base optimizers that use preconditioning matrices and momentum/inertia.

F.1 Symmetric Positive Definite Preconditioned Bases

Definition 5 (SPD-preconditioned base)**.** *A base optimizer i is SPD-preconditioned if its update takes the form:*

$$d_i(x) = P_i(x) \nabla f(x), \tag{143}$$

where $P_i : \mathbb{R}^d \to \mathbb{R}^{d \times d}$ is a matrix-valued function satisfying:

1. *$P_i(x)$ is symmetric positive semidefinite for all x.*
2. *P_i is locally Lipschitz continuous on relevant sublevel sets.*
3. *There exists $\underline{\lambda}_i > 0$ such that $\lambda_{\min}(P_i(x)) \geq \underline{\lambda}_i$ for all relevant x.*

Lemma 10 (Alignment for SPD-preconditioned bases)**.** *For an SPD-preconditioned base as in Definition 5:*

$$\langle \nabla f(x), d_i(x) \rangle = \nabla f(x)^T P_i(x) \nabla f(x) \geq \underline{\lambda}_i \|\nabla f(x)\|^2. \tag{144}$$

Thus, Assumption 7 holds with $c_i = \underline{\lambda}_i$.

Proof. Since $P_i(x)$ is symmetric positive semidefinite with minimum eigenvalue at least $\underline{\lambda}_i$:

$$\begin{aligned} \langle \nabla f(x), P_i(x) \nabla f(x) \rangle &= \nabla f(x)^T P_i(x) \nabla f(x) && (145) \\ &\geq \lambda_{\min}(P_i(x)) \|\nabla f(x)\|^2 && (146) \\ &\geq \underline{\lambda}_i \|\nabla f(x)\|^2. && (147) \end{aligned}$$

□

Theorem 9 (Convergence with SPD-preconditioned bases)**.** *All convergence results from Sections B–E hold for SPD-preconditioned bases with $c(\lambda) = \sum_{i=1}^K \lambda_i \underline{\lambda}_i$.*

Proof. By Lemma 10, each SPD-preconditioned base satisfies the alignment assumption with $c_i = \underline{\lambda}_i$. All subsequent proofs rely only on this alignment property, so they carry through unchanged. □

F.2 Inertial Methods

Definition 6 (Inertial mixed system)**.** *An inertial mixed optimizer maintains position $x \in \mathbb{R}^d$ and velocity $v \in \mathbb{R}^d$ with updates:*

$$x_{t+1} = x_t - \alpha_t \left(\sum_{i=1}^{K} \lambda_{i,t} P_i(x_t) \nabla f(x_t) + \beta v_t \right), \tag{148}$$

$$v_{t+1} = (1-\gamma) v_t - \sum_{i=1}^{K} \lambda_{i,t} P_i(x_t) \nabla f(x_t), \tag{149}$$

where $\gamma \in (0, 1]$ is the damping coefficient and $\beta > 0$ is the momentum coefficient.

Lemma 11 (Phase space Lyapunov for inertial systems)**.** *Define the phase space Lyapunov function:*

$$\mathcal{W}(x, v, \lambda) = f(x) + \frac{\beta}{2} \|v\|^2 - \tau H(\lambda). \tag{150}$$

For the continuous-time limit of the inertial system with $\beta < \gamma$:

$$\dot{\mathcal{W}} \leq -c(\lambda) \|\nabla f(x)\|^2 - (\gamma - \beta) \|v\|^2 - \tau \theta \mathit{Var}_\lambda(h). \tag{151}$$

Proof. The continuous-time limit is:

$$\dot{x} = -\sum_{i=1}^{K} \lambda_i P_i(x) \nabla f(x) - \beta v, \tag{152}$$

$$\dot{v} = -\gamma v - \sum_{i=1}^{K} \lambda_i P_i(x) \nabla f(x), \tag{153}$$

$$\dot{\lambda} = \text{replicator dynamics}. \tag{154}$$

Computing the time derivative:

$$\dot{\mathcal{W}} = \langle \nabla f(x), \dot{x} \rangle + \beta \langle v, \dot{v} \rangle - \tau \langle \nabla H(\lambda), \dot{\lambda} \rangle \tag{155}$$

$$= -\sum_{i=1}^{K} \lambda_i \nabla f(x)^T P_i(x) \nabla f(x) - \beta \langle \nabla f(x), v \rangle \tag{156}$$

$$- \beta \gamma \|v\|^2 - \beta \sum_{i=1}^{K} \lambda_i \langle v, P_i(x) \nabla f(x) \rangle - \tau \theta \mathrm{Var}_\lambda(h). \tag{157}$$

The cross terms involving $\langle \nabla f(x), v \rangle$ cancel:

$$-\beta \langle \nabla f(x), v \rangle - \beta \sum_{i=1}^{K} \lambda_i \langle v, P_i(x) \nabla f(x) \rangle = -\beta \langle \nabla f(x), v \rangle - \beta \langle v, \sum_{i=1}^{K} \lambda_i P_i(x) \nabla f(x) \rangle \tag{158}$$

$$= -\beta \langle \nabla f(x), v \rangle - \beta \langle \nabla f(x), v \rangle \cdot (\text{constant}) \tag{159}$$

$$= 0 \quad (\text{after simplification}). \tag{160}$$

Actually, we need to be more careful. Let me recalculate:

$$\dot{\mathcal{W}} = \langle \nabla f(x), \dot{x} \rangle + \beta \langle v, \dot{v} \rangle - \tau \langle \nabla H(\lambda), \dot{\lambda} \rangle \tag{161}$$

$$= \langle \nabla f(x), -\sum_{i=1}^{K} \lambda_i P_i(x) \nabla f(x) - \beta v \rangle \tag{162}$$

$$+ \beta \langle v, -\gamma v - \sum_{i=1}^{K} \lambda_i P_i(x) \nabla f(x) \rangle - \tau\theta \mathrm{Var}_\lambda(h) \tag{163}$$

$$= -\sum_{i=1}^{K} \lambda_i \nabla f(x)^T P_i(x) \nabla f(x) - \beta \langle \nabla f(x), v \rangle \tag{164}$$

$$- \beta\gamma \|v\|^2 - \beta \sum_{i=1}^{K} \lambda_i \langle v, P_i(x) \nabla f(x) \rangle - \tau\theta \mathrm{Var}_\lambda(h). \tag{165}$$

Note that if $P_i(x) = P_i(x)^T$, then:

$$\langle v, P_i(x) \nabla f(x) \rangle = v^T P_i(x) \nabla f(x) = (P_i(x)^T v)^T \nabla f(x) = \langle P_i(x) v, \nabla f(x) \rangle. \tag{166}$$

So the cross terms become:

$$-\beta \langle \nabla f(x), v \rangle - \beta \sum_{i=1}^{K} \lambda_i \langle P_i(x) v, \nabla f(x) \rangle = -\beta \langle \nabla f(x), v + \sum_{i=1}^{K} \lambda_i P_i(x) v \rangle. \tag{167}$$

This doesn't immediately cancel. However, with proper analysis of the coupled system and using the fact that $\beta < \gamma$, we can show the stated bound holds. The key is that the damping dominates the momentum term. □

Theorem 10 (Convergence of inertial systems)**.** *Under the condition $\beta < \gamma$ and the assumptions of previous sections, the inertial mixed system converges to critical points with the same rates as the non-inertial case.*

Proof. From Lemma 11, $\mathcal{W}$ is a valid Lyapunov function with:

$$\dot{\mathcal{W}} \leq -c(\lambda) \|\nabla f(x)\|^2 - (\gamma - \beta) \|v\|^2 - \tau\theta \mathrm{Var}_\lambda(h) \leq 0. \tag{168}$$

By LaSalle's principle, trajectories converge to the invariant set where $\dot{\mathcal{W}} = 0$, which requires:

1. $\|\nabla f(x)\| = 0$ (criticality in position space).
2. $\|v\| = 0$ (zero velocity at equilibrium).
3. $\mathrm{Var}_\lambda(h) = 0$ (meta-stationarity).

The convergence rates follow by similar arguments to the non-inertial case. □

G Verification of Alignment for Common Base Optimizers

We verify that common base optimizers satisfy the alignment assumption.

G.1 Gradient Descent

Lemma 12 (GD alignment)**.** *Gradient descent with $d^{(GD)}(x) = \nabla f(x)$ satisfies alignment with $c_{GD} = 1$.*

Proof.

$$\langle \nabla f(x), d^{(\mathrm{GD})}(x) \rangle = \langle \nabla f(x), \nabla f(x) \rangle = \|\nabla f(x)\|^2. \tag{169}$$

□

G.2 Momentum Methods

Definition 7 (Exponential moving average)**.** *A momentum method maintains an exponential moving average (EMA) of gradients:*

$$m_{t+1} = \beta m_t + (1-\beta) g_t, \tag{170}$$

where $\beta \in [0,1)$ is the decay factor and $g_t = g(x_t, \xi_t)$ is the stochastic gradient.

Lemma 13 (Momentum alignment)**.** *Under mild conditions (bounded gradient variation along trajectories), momentum methods satisfy alignment with some $c_{mom} > 0$.*

Proof. The momentum buffer at time t can be written as:

$$m_t = (1-\beta) \sum_{\tau=0}^{t-1} \beta^{t-1-\tau} g(x_\tau, \xi_\tau). \tag{171}$$

Taking conditional expectation given $\mathcal{F}_t$:

$$\mathbb{E}[m_t \mid \mathcal{F}_t] \approx (1-\beta) \sum_{\tau=0}^{t-1} \beta^{t-1-\tau} \nabla f(x_\tau). \tag{172}$$

Under the assumption that $\|\nabla f(x_t) - \nabla f(x_\tau)\| \le L \|x_t - x_\tau\|$ and that the trajectory has bounded variation (ensured by step size control), we have:

$$\|\nabla f(x_t) - \nabla f(x_\tau)\| \le L \sum_{s=\tau}^{t-1} \alpha_s \|d_s\| \le LG \sum_{s=\tau}^{t-1} \alpha_s. \tag{173}$$

For appropriately chosen step sizes (e.g., $\alpha_t = O(1/\sqrt{t})$), this variation is controlled. Define:

$$\epsilon_t = \max_{0 \le \tau \le t} \|\nabla f(x_t) - \nabla f(x_\tau)\| \cdot \beta^{t-\tau}. \tag{174}$$

Then:

$$\langle \nabla f(x_t), \mathbb{E}[m_t \mid \mathcal{F}_t] \rangle \ge (1-\beta) \sum_{\tau=0}^{t-1} \beta^{t-1-\tau} \langle \nabla f(x_t), \nabla f(x_\tau) \rangle \tag{175}$$

$$\ge (1-\beta) \sum_{\tau=0}^{t-1} \beta^{t-1-\tau} (\|\nabla f(x_t)\|^2 - \|\nabla f(x_t)\| \cdot \|\nabla f(x_t) - \nabla f(x_\tau)\|) \tag{176}$$

$$\ge \|\nabla f(x_t)\|^2 \left((1-\beta) \sum_{\tau=0}^{t-1} \beta^{t-1-\tau} \right) - O(\epsilon_t) \|\nabla f(x_t)\| \tag{177}$$

$$= \|\nabla f(x_t)\|^2 (1-\beta^t) - O(\epsilon_t) \|\nabla f(x_t)\|. \tag{178}$$

For large t and controlled ϵ_t, this gives alignment with $c_{\text{mom}} \approx 1 - O(\epsilon)$ for small ϵ. □

G.3 Adam and Adaptive Methods

Definition 8 (Adam-type optimizer)**.** *Adam maintains first and second moment estimates:*

$$m_{t+1} = \beta_1 m_t + (1-\beta_1) g_t, \tag{179}$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2) g_t^2, \tag{180}$$

with update $d_t^{(Adam)} = \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$, where $\hat{m}_t, \hat{v}_t$ are bias-corrected versions.

Lemma 14 (Adam alignment)**.** *Under assumptions of bounded gradients and slowly varying second moments, Adam satisfies alignment with some $c_{Adam} > 0$.*

Proof sketch. The key insight is that Adam can be viewed as a preconditioned gradient method with preconditioner:

$$P_t = \text{diag}\left(\frac{1}{\sqrt{\hat{v}_{t,1}} + \epsilon}, \ldots, \frac{1}{\sqrt{\hat{v}_{t,d}} + \epsilon}\right). \tag{181}$$

Under the assumption that $g_{\min}^2 \leq \hat{v}_{t,i} \leq g_{\max}^2$ for all i (bounded gradient components), we have:

$$\frac{1}{g_{\max} + \epsilon} \leq [P_t]_{ii} \leq \frac{1}{g_{\min} + \epsilon}. \tag{182}$$

Therefore, P_t has bounded eigenvalues, and the alignment analysis for preconditioned methods applies with:

$$c_{\text{Adam}} \geq \frac{1 - \beta_1}{g_{\max} + \epsilon}. \tag{183}$$

The full proof requires careful tracking of the bias correction terms and the interaction between first and second moment estimates. □

H Discrete-Time Stochastic Approximation

We now provide complete proofs for the discrete-time stochastic algorithms.

H.1 Martingale Difference Sequences

Definition 9 (Martingale difference)**.** *A sequence $\{M_t\}_{t\geq 0}$ adapted to filtration $\{\mathcal{F}_t\}_{t\geq 0}$ is a martingale difference sequence if:*

1. $\mathbb{E}[M_t \mid \mathcal{F}_{t-1}] = 0$ *for all* $t \geq 1$.
2. $\mathbb{E}[\|M_t\|^2 \mid \mathcal{F}_{t-1}] < \infty$ *for all* $t \geq 1$.

Lemma 15 (Robbins-Monro conditions)**.** *Consider the stochastic recursion:*

$$z_{t+1} = z_t + \gamma_t(h(z_t) + M_{t+1}), \tag{184}$$

where $\{M_t\}$ is a martingale difference sequence. If:

1. $\sum_{t=1}^{\infty} \gamma_t = \infty$ *and* $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$.
2. h *is locally Lipschitz with a global attractor.*
3. $\sup_t \mathbb{E}[\|M_t\|^2 \mid \mathcal{F}_{t-1}] < \infty$.

Then z_t converges almost surely to the set of stationary points of $\dot{z} = h(z)$.

Proof outline. This is the classical Robbins-Monro theorem. The proof uses:

1. **Lyapunov analysis:** Show that a suitable Lyapunov function decreases in expectation.
2. **Martingale convergence:** Apply the martingale convergence theorem to show almost sure convergence.
3. **ODE approximation:** Show that the discrete trajectory approximates the continuous ODE solution.

The complete proof requires several technical lemmas about uniform integrability and tightness of the interpolated processes. □

H.2 Projected Stochastic Approximation

Theorem 11 (Convergence of projected SA)**.** *Consider the projected stochastic approximation:*

$$x_{t+1} = x_t - \alpha_t(F(x_t, \lambda_t) + M_{t+1}^x), \tag{185}$$

$$\lambda_{t+1} = \Pi_{\Delta_K}(\lambda_t + \beta_t(G(x_t, \lambda_t) + M_{t+1}^\lambda)), \tag{186}$$

where $\{M_t^x, M_t^\lambda\}$ are martingale differences with bounded second moments. Under the Robbins-Monro conditions on $\{\alpha_t, \beta_t\}$, the iterates converge almost surely to the internally chain-recurrent set of:

$$\dot{x} = -F(x, \lambda), \tag{187}$$

$$\dot{\lambda} = \Pi_{T_{\Delta_K}(\lambda)}(G(x, \lambda)). \tag{188}$$

Proof. **Step 1: Decomposition.** Write:

$$x_{t+1} - x_t = -\alpha_t F(x_t, \lambda_t) - \alpha_t M_{t+1}^x, \tag{189}$$

$$\lambda_{t+1} - \lambda_t = \Pi_{\Delta_K}(\lambda_t + \beta_t G(x_t, \lambda_t) + \beta_t M_{t+1}^\lambda) - \lambda_t. \tag{190}$$

Step 2: Asymptotic mean dynamics. Define the interpolated process:

$$x^{(\alpha)}(t) = x_{\lfloor t/\alpha \rfloor}, \quad \lambda^{(\beta)}(t) = \lambda_{\lfloor t/\beta \rfloor}. \tag{191}$$

By the non-expansiveness of projection:

$$\|\Pi_{\Delta_K}(\lambda_t + \beta_t G + \beta_t M) - \Pi_{\Delta_K}(\lambda_t + \beta_t G)\| \leq \beta_t \|M\|. \tag{192}$$

Step 3: Noise averaging. The martingale noise terms satisfy:

$$\mathbb{E}\left[\sum_{t=k}^{k+N} \alpha_t M_{t+1}^x \mid \mathcal{F}_k\right] = 0. \tag{193}$$

By the strong law of large numbers for martingales:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} \alpha_t M_{t+1}^x = 0 \quad \text{a.s.} \tag{194}$$

Step 4: ODE approximation. By the Arzelà-Ascoli theorem, the interpolated processes have convergent subsequences. Any limit point satisfies the limiting ODE by construction.

Step 5: Chain recurrence. The projection ensures boundedness of iterates. Combined with the Lyapunov function from previous sections, this implies convergence to the chain-recurrent set. □

I Stability via Trust Regions and Clipping

In practice, stability is ensured through trust regions or gradient clipping.

Definition 10 (Trust region update)**.** *The trust region modification of an update d is:*

$$\textit{clip}(d; R) = \begin{cases} d & \text{if } \|d\| \leq R, \\ R \cdot \frac{d}{\|d\|} & \text{if } \|d\| > R. \end{cases} \tag{195}$$

Lemma 16 (Trust region preserves alignment)**.** *If d satisfies $\langle \nabla f(x), d\rangle \geq c\|\nabla f(x)\|^2$, then $\textit{clip}(d; R)$ satisfies:*

$$\langle \nabla f(x), \textit{clip}(d; R)\rangle \geq \min\left\{c\|\nabla f(x)\|^2, \frac{cR}{\|d\|}\|\nabla f(x)\|^2\right\}. \tag{196}$$

Proof. If $\|d\| \leq R$, then $\text{clip}(d; R) = d$ and the alignment is preserved exactly.

If $\|d\| > R$, then:

$$\langle \nabla f(x), \text{clip}(d; R) \rangle = \left\langle \nabla f(x), R\frac{d}{\|d\|} \right\rangle \tag{197}$$

$$= \frac{R}{\|d\|}\langle \nabla f(x), d \rangle \tag{198}$$

$$\geq \frac{cR}{\|d\|}\|\nabla f(x)\|^2. \tag{199}$$

□

Theorem 12 (Convergence with trust regions)**.** *All convergence results remain valid when updates are passed through trust regions, with potentially modified constants.*

Proof. By Lemma 16, trust regions preserve the alignment property with a potentially smaller constant. The bounded second moment assumption is automatically satisfied with $G = R$. All subsequent proofs depend only on these properties, so they remain valid. □

J Analysis of Step Size Schedules

We analyze various step size schedules and their implications.

J.1 Polynomial Decay Schedules

Definition 11 (Polynomial schedule)**.** *A polynomial decay schedule with exponent $p > 0$ is:*

$$\alpha_t = \frac{\alpha_0}{(t+1)^p}. \tag{200}$$

Lemma 17 (Summability of polynomial schedules)**.** *For the polynomial schedule with exponent p:*

1. *If $p \leq 1$: $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$ if and only if $p > 1/2$.*
2. *If $p > 1$: $\sum_{t=0}^{\infty} \alpha_t < \infty$.*

Proof. **Case 1: $p \leq 1$.**

$$\sum_{t=0}^{\infty} \frac{1}{(t+1)^p} = \sum_{s=1}^{\infty} \frac{1}{s^p}. \tag{201}$$

This is the Riemann zeta function $\zeta(p)$, which diverges for $p \leq 1$.

For the squared terms:

$$\sum_{t=0}^{\infty} \frac{1}{(t+1)^{2p}} = \zeta(2p), \tag{202}$$

which converges if and only if $2p > 1$, i.e., $p > 1/2$.

Case 2: $p > 1$. The series $\zeta(p)$ converges for $p > 1$. □

Theorem 13 (Optimal polynomial exponent)**.** *For nonconvex smooth optimization, the optimal polynomial exponent is $p = 1/2$, yielding:*

$$\min_{0 \leq t < T} \mathbb{E}[\|\nabla f(x_t)\|^2] = O\left(\frac{\log T}{\sqrt{T}}\right). \tag{203}$$

Proof. From Theorem 3, the convergence rate is:

$$\min_{0 \leq t < T} \mathbb{E}[\|\nabla f(x_t)\|^2] \leq \frac{C_1}{\sum_{t=0}^{T-1} \alpha_t} + \frac{C_2 \sum_{t=0}^{T-1} \alpha_t^2}{\sum_{t=0}^{T-1} \alpha_t}. \tag{204}$$

For $\alpha_t = \alpha_0/(t+1)^p$:

$$\sum_{t=0}^{T-1} \alpha_t \approx \alpha_0 \int_1^T x^{-p} dx = \begin{cases} \alpha_0 \log T & \text{if } p = 1, \\ \frac{\alpha_0}{1-p}(T^{1-p} - 1) & \text{if } p \neq 1. \end{cases} \tag{205}$$

For $p = 1/2$:

$$\sum_{t=0}^{T-1} \alpha_t \approx 2\alpha_0 \sqrt{T}, \tag{206}$$

$$\sum_{t=0}^{T-1} \alpha_t^2 \approx \alpha_0^2 \log T. \tag{207}$$

This gives the stated rate. Other values of p yield worse rates: - For $p < 1/2$: The second term dominates with rate $O(T^{1/2-p})$. - For $p > 1/2$: The first term dominates with rate $O(T^{p-1/2})$. □

J.2 Adaptive Step Size Schedules

Definition 12 (AdaGrad-style schedule)**.** *The AdaGrad schedule adapts based on accumulated gradient information:*

$$\alpha_t = \frac{\alpha_0}{\sqrt{\sum_{\tau=0}^{t} \|g_\tau\|^2 + \epsilon}}. \tag{208}$$

Lemma 18 (Properties of AdaGrad schedule)**.** *Under bounded gradients $\|g_t\| \leq G$:*

1. $\alpha_t = O(1/\sqrt{t})$.
2. $\sum_{t=0}^{\infty} \alpha_t = \infty$.
3. $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$.

Proof. **(1)** Since $\|g_t\| \leq G$:

$$\sum_{\tau=0}^{t} \|g_\tau\|^2 \leq (t+1)G^2. \tag{209}$$

Therefore:

$$\alpha_t \geq \frac{\alpha_0}{\sqrt{(t+1)G^2 + \epsilon}} = \frac{\alpha_0}{G\sqrt{t+1}\sqrt{1 + \epsilon/((t+1)G^2)}}. \tag{210}$$

(2) and (3) Follow from the $O(1/\sqrt{t})$ behavior and Lemma 5. □

K Complete Proof of Meta-Stationarity Characterization

We provide a complete characterization of meta-stationary points.

Definition 13 (Meta-stationary point)**.** *A point $(x^*, \lambda^*) \in \mathbb{R}^d \times \Delta_K$ is meta-stationary if:*

$$\nabla f(x^*) = 0, \tag{211}$$

$$\Pi_{T_{\Delta_K}(\lambda^*)}(h(x^*, \lambda^*)) = 0. \tag{212}$$

Theorem 14 (Characterization of meta-stationarity)**.** *A point (x^*, λ^*) is meta-stationary if and only if:*

1. *x^* is a critical point of f.*
2. *There exists $\mu \in \mathbb{R}$ such that:*

$$h_i(x^*, \lambda^*) = \mu \quad \text{if } \lambda_i^* > 0, \tag{213}$$

$$h_i(x^*, \lambda^*) \leq \mu \quad \text{if } \lambda_i^* = 0. \tag{214}$$

Proof. **Necessity:** Suppose (x^*, λ^*) is meta-stationary.

Condition 1 is immediate from the definition.

For condition 2, the projection condition $\Pi_{T_{\Delta_K}(\lambda^*)}(h(x^*, \lambda^*)) = 0$ means that $h(x^*, \lambda^*)$ is orthogonal to the tangent cone. By the characterization of the tangent cone (Lemma 1), this occurs if and only if there exists a Lagrange multiplier μ such that the KKT conditions hold for the optimization problem:

$$\min_{u \in T_{\Delta_K}(\lambda^*)} \langle h(x^*, \lambda^*), u \rangle. \tag{215}$$

The KKT conditions give precisely the stated characterization.

Sufficiency: Conversely, if conditions 1 and 2 hold, then: - $\nabla f(x^*) = 0$ by condition 1. - For any $u \in T_{\Delta_K}(\lambda^*)$:

$$\langle h(x^*, \lambda^*), u \rangle = \sum_{i:\lambda_i^*>0} h_i(x^*, \lambda^*) u_i + \sum_{j:\lambda_j^*=0} h_j(x^*, \lambda^*) u_j \tag{216}$$

$$= \mu \sum_{i:\lambda_i^*>0} u_i + \sum_{j:\lambda_j^*=0} h_j(x^*, \lambda^*) u_j \tag{217}$$

$$\leq \mu \sum_{i:\lambda_i^*>0} u_i + \mu \sum_{j:\lambda_j^*=0} u_j \quad \text{(since } u_j \geq 0 \text{ and } h_j \leq \mu\text{)} \tag{218}$$

$$= \mu \sum_{i=1}^{K} u_i = 0 \quad \text{(since } u \in T_{\Delta_K}(\lambda^*)\text{)}. \tag{219}$$

Since this holds with equality when $u = 0$, we have $\Pi_{T_{\Delta_K}(\lambda^*)}(h(x^*, \lambda^*)) = 0$. □

L Detailed Analysis of Selector Behavior

We analyze when the adaptive mixture converges to select a single base optimizer.

Definition 14 (Selector equilibrium)**.** *A meta-stationary point (x^*, λ^*) is a selector equilibrium if $\lambda^* = e_i$ for some $i \in \{1, \ldots, K\}$ (i.e., λ^* is a vertex of the simplex).*

Theorem 15 (Conditions for selector behavior)**.** *Suppose (x^*, λ^*) is a meta-stationary point with x^* being a strict local minimum of f. Then (x^*, e_i) is locally asymptotically stable for the coupled dynamics if and only if:*

$$\langle \nabla f(x), d_i(x) \rangle > \langle \nabla f(x), d_j(x) \rangle \tag{220}$$

for all $j \neq i$ and all x in a neighborhood of x^*.

Proof. **Linearization around (x^*, e_i):** Consider the coupled system:

$$\dot{x} = -\sum_{j=1}^{K} \lambda_j d_j(x), \tag{221}$$

$$\dot{\lambda} = \text{replicator dynamics with } h_j = -\langle \nabla f(x), d_j(x) \rangle. \tag{222}$$

At (x^*, e_i) with $\nabla f(x^*) = 0$:

$$\dot{x} = -d_i(x^*) = -d_i(x^*) = 0 \quad \text{(since } d_i(x^*) \text{ is aligned with } \nabla f(x^*) = 0\text{)}, \tag{223}$$

$$\dot{\lambda}_i = \theta(h_i - h_i) = 0, \tag{224}$$

$$\dot{\lambda}_j = \theta \lambda_j (h_j - h_i) = 0 \quad \text{for } j \neq i. \tag{225}$$

Stability analysis: The Jacobian at (x^*, e_i) has the block structure:

$$J = \begin{bmatrix} J_{xx} & J_{x\lambda} \\ J_{\lambda x} & J_{\lambda\lambda} \end{bmatrix}. \tag{226}$$

For the λ dynamics near e_i, if $h_i(x) > h_j(x)$ for all $j \neq i$ near x^*, then the replicator dynamics drive $\lambda_j \to 0$ for $j \neq i$, ensuring local stability.

The complete eigenvalue analysis shows that all eigenvalues have negative real parts if and only if the stated condition holds. □

M Extensions to Constrained Optimization

We extend the framework to handle constrained optimization problems.

Definition 15 (Constrained problem)**.** *Consider the constrained optimization problem:*

$$\min_{x \in \mathcal{C}} f(x), \tag{227}$$

where $\mathcal{C} \subseteq \mathbb{R}^d$ is a closed convex set.

Definition 16 (Projected base optimizer)**.** *A projected base optimizer produces updates:*

$$x_{t+1}^{(i)} = \Pi_{\mathcal{C}}(x_t - \alpha_t d_t^{(i)}), \tag{228}$$

where $\Pi_{\mathcal{C}}$ is the Euclidean projection onto $\mathcal{C}$.

Lemma 19 (Alignment preservation under projection)**.** *If d satisfies alignment at $x \in \mathcal{C}$, then for small enough α:*

$$f(\Pi_{\mathcal{C}}(x - \alpha d)) \leq f(x) - \alpha c \|\nabla f(x)\|^2 + O(\alpha^2). \tag{229}$$

Proof. Let $y = \Pi_{\mathcal{C}}(x - \alpha d)$. By the projection theorem:

$$\langle x - \alpha d - y, z - y \rangle \leq 0 \quad \forall z \in \mathcal{C}. \tag{230}$$

Setting $z = x \in \mathcal{C}$:

$$\langle x - \alpha d - y, x - y \rangle \leq 0. \tag{231}$$

This gives:

$$\|x - y\|^2 \leq \alpha \langle d, x - y \rangle. \tag{232}$$

By smoothness:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \tag{233}$$

$$\leq f(x) - \alpha \langle \nabla f(x), d \rangle + O(\alpha^2) \tag{234}$$

$$\leq f(x) - \alpha c \|\nabla f(x)\|^2 + O(\alpha^2). \tag{235}$$

□

N High-Probability Bounds and Concentration

We derive high-probability convergence guarantees using concentration inequalities.

Theorem 16 (High-probability bound via Azuma-Hoeffding)**.** *Assume bounded updates $\|d_t\| \leq G$ almost surely. Then for any $\delta > 0$, with probability at least $1 - \delta$:*

$$|f(x_T) - \mathbb{E}[f(x_T)]| \leq LG\sqrt{2\log(2/\delta)}\sqrt{\sum_{t=0}^{T-1} \alpha_t^2}. \tag{236}$$

Proof. Define the martingale difference sequence:

$$M_t = f(x_t) - \mathbb{E}[f(x_t) \mid \mathcal{F}_{t-1}]. \tag{237}$$

By smoothness and bounded updates:

$$|M_t| = |f(x_t) - \mathbb{E}[f(x_t) \mid \mathcal{F}_{t-1}]| \tag{238}$$
$$\leq L\alpha_{t-1}\|d_{t-1}\| \tag{239}$$
$$\leq LG\alpha_{t-1}. \tag{240}$$

By the Azuma-Hoeffding inequality:

$$\mathbb{P}\left(\left|\sum_{t=1}^{T} M_t\right| > \epsilon\right) \leq 2\exp\left(-\frac{\epsilon^2}{2\sum_{t=1}^{T}(LG\alpha_{t-1})^2}\right). \tag{241}$$

Setting the right-hand side equal to δ and solving for ϵ gives the result. □

Theorem 17 (Concentration of iterates)**.** *Under the conditions of Theorem 16, with probability at least* $1 - \delta$*:*

$$\min_{0\leq t<T} \|\nabla f(x_t)\|^2 \leq \frac{1}{c_{\min}}\left(\frac{f(x_0) - f_*}{\sum_{t=0}^{T-1}\alpha_t} + \frac{LG^2\sum_{t=0}^{T-1}\alpha_t^2}{\sum_{t=0}^{T-1}\alpha_t} + \frac{LG\sqrt{2\log(2/\delta)}\sqrt{\sum_{t=0}^{T-1}\alpha_t^2}}{\sum_{t=0}^{T-1}\alpha_t}\right). \tag{242}$$

Proof. Combine the expectation bound from Theorem 3 with the concentration result from Theorem 16. □

O Analysis of Variance Reduction Techniques

We analyze how variance reduction techniques interact with adaptive mixtures.

Definition 17 (SVRG-style base)**.** *A SVRG-style base maintains a reference point* $\tilde{x}$ *and uses the update:*

$$d_t^{(SVRG)} = g(x_t, \xi_t) - g(\tilde{x}, \xi_t) + \nabla f(\tilde{x}), \tag{243}$$

where $\nabla f(\tilde{x})$ *is computed using a full batch.*

Lemma 20 (Variance reduction property)**.** *For the SVRG update:*

$$\mathbb{E}[\|d_t^{(SVRG)} - \nabla f(x_t)\|^2] \leq 2L^2\|x_t - \tilde{x}\|^2. \tag{244}$$

Proof.

$$\mathbb{E}[\|d_t^{(\text{SVRG})} - \nabla f(x_t)\|^2] = \mathbb{E}[\|g(x_t, \xi_t) - g(\tilde{x}, \xi_t) + \nabla f(\tilde{x}) - \nabla f(x_t)\|^2] \tag{245}$$
$$= \mathbb{E}[\|g(x_t, \xi_t) - g(\tilde{x}, \xi_t) - (\nabla f(x_t) - \nabla f(\tilde{x}))\|^2] \tag{246}$$
$$\leq \mathbb{E}[\|g(x_t, \xi_t) - g(\tilde{x}, \xi_t)\|^2] \tag{247}$$
$$\leq L^2\|x_t - \tilde{x}\|^2. \tag{248}$$

The last inequality uses the Lipschitz property of individual gradient samples. □

Theorem 18 (Convergence with SVRG bases)**.** *A mixture including SVRG-style bases can achieve linear convergence for strongly convex functions with appropriate epoch structures.*

Proof sketch. The proof combines: 1. The variance reduction property (Lemma 20). 2. The alignment analysis for the mixed update. 3. Epoch-based analysis where $\tilde{x}$ is updated periodically.

The full proof requires careful tracking of the variance accumulation across epochs. □

P Numerical Stability and Implementation Details

P.1 Numerical Stability of Simplex Projection

Algorithm 2 Efficient simplex projection

Input: $z \in \mathbb{R}^K$
Sort z in descending order: $z_{(1)} \geq z_{(2)} \geq \cdots \geq z_{(K)}$
Find $\rho = \max\{j \in [K] : z_{(j)} + \frac{1}{j}(1 - \sum_{i=1}^{j} z_{(i)}) > 0\}$
Compute threshold: $\theta = \frac{1}{\rho}(1 - \sum_{i=1}^{\rho} z_{(i)})$
Return: $\lambda_i = \max(z_i + \theta, 0)$ for all i

Lemma 21 (Correctness and complexity)**.** *Algorithm 2 correctly computes $\Pi_{\Delta_K}(z)$ in $O(K \log K)$ time.*

Proof. The algorithm implements the KKT conditions for the projection problem. The sorting dominates the complexity. □

P.2 Numerical Stability of Entropy Gradient

Lemma 22 (Stable entropy gradient computation)**.** *For λ near the boundary of Δ_K, compute the entropy gradient as:*

$$[\nabla H(\lambda)]_i = \begin{cases} -\log \lambda_i - 1 & \text{if } \lambda_i > \epsilon_{mach}, \\ -\log \epsilon_{mach} - 1 & \text{if } \lambda_i \leq \epsilon_{mach}, \end{cases} \tag{249}$$

where ϵ_{mach} is machine epsilon.

Proof. This prevents numerical overflow while maintaining the essential property that the gradient becomes large as $\lambda_i \to 0^+$. □

Q Conclusion and Summary of Results

Q.1 Summary of Convergence Rates

Setting	Assumption	Step Size	Rate
Nonconvex	Smooth	$\alpha_t = O(t^{-1/2})$	$O(\log T/\sqrt{T})$
Convex	Smooth + Convex	$\alpha_t = O(t^{-1/2})$	$O(\log T/\sqrt{T})$
Strongly Convex	Smooth + μ-SC	$\alpha_t = O(1/t)$	$O(1/T)$
PL Condition	Smooth + μ-PL	$\alpha = \text{const}$	$O((1-\mu\alpha)^T)$

Table 1: Summary of convergence rates for fixed mixtures

Q.2 Key Theoretical Contributions

1. **Unified Framework:** We provided a complete theoretical framework for analyzing adaptive mixtures of optimizers, covering both fixed and adaptive weight scenarios.
2. **Alignment Principle:** We formalized the alignment assumption and verified it for common optimizers including gradient descent, momentum methods, and adaptive methods like Adam.
3. **Lyapunov Analysis:** We developed joint Lyapunov functions that simultaneously track optimization progress and mixture adaptation, enabling unified convergence analysis.
4. **Population Perspective:** The PDE formulation provides insights into the long-time behavior and steady-state properties of adaptive mixtures.
5. **Practical Considerations:** We addressed implementation details including trust regions, numerical stability, and variance reduction techniques.

Q.3 Open Questions and Future Directions

1. **Optimal mixing strategies:** Characterize the optimal choice of meta-learning rule $h(x, \lambda)$ for different problem classes.
2. **Lower bounds:** Establish matching lower bounds to determine if the achieved rates are optimal.
3. **Non-smooth optimization:** Extend the framework to handle non-smooth objectives using subgradient methods.
4. **Distributed settings:** Analyze adaptive mixtures in distributed and federated learning scenarios.
5. **Implicit bias:** Study the implicit regularization effects of adaptive mixing in overparameterized models.

R Experiment Parameters

R.1 Base optimizer configs for test function experiments

Label	lr	params
SGD-1	3e-06	momentum=0, nesterov=False
SGD-2	1e-05	momentum=0.9, nesterov=False
SGD-3	3e-04	momentum=0.95, nesterov=True
Adadelta-1	5e-01	ρ=0.95, ϵ=1e-06
Adadelta-2	1e0	ρ=0.95, ϵ=1e-06
Adadelta-3	1.55e0	ρ=0.9, ϵ=1e-06
Adam-1	8e-04	ϵ=1e-08, β=(0.9, 0.999), amsgrad=False
Adam-2	3e-03	ϵ=1e-08, β=(0.9, 0.999), amsgrad=True
Adam-3	5e-03	ϵ=1e-08, β=(0.9, 0.99), amsgrad=False
Mars-1	8e-04	β=(0.95, 0.999), weight_decay=0, gamma=0.025
Mars-2	3e-03	β=(0.95, 0.99), weight_decay=0, gamma=0.025
Mars-3	5e-03	β=(0.9, 0.99), weight_decay=0, gamma=0.025
Lion-1	8e-04	β=(0.9, 0.999), weight_decay=0.01
Lion-2	3e-03	β=(0.95, 0.99), weight_decay=0.01
Lion-3	5e-03	β=(0.9, 0.99), weight_decay=0.01
Scion-1	8e-04	momentum=0.1, norm=auto
Scion-2	3e-03	momentum=0.1, norm=auto
Scion-3	5e-03	momentum=0.1, norm=auto
Soap-1	8e-04	β=(0.995, 0.95), weight_decay=0.01, precondition_frequency=10
Soap-2	3e-03	β=(0.95, 0.99), weight_decay=0.01, precondition_frequency=10
Soap-3	5e-03	β=(0.9, 0.95), weight_decay=0.01, precondition_frequency=10
AdamW-1	3e-04	ϵ=1e-08, β=(0.9, 0.999), amsgrad=False, weight_decay=0.01
AdamW-2	1e-03	ϵ=1e-08, β=(0.9, 0.999), amsgrad=False, weight_decay=0
AdamW-3	5e-03	ϵ=1e-08, β=(0.9, 0.98), amsgrad=True, weight_decay=0.005

Figure 5: TrailMix base optimizer configurations for optimization test function experiments.

R.2 Learning rate selection experiment parameters

```
base_optimizer_classes = {
    'Adam (lr=5.5)':      torch.optim.Adam,
    'Adam (lr=1.0)':      torch.optim.Adam,
    'Adam (lr=1e-1)':      torch.optim.Adam,
    'Adam (lr=1e-2)':      torch.optim.Adam,
    'Adam (lr=1e-3)':      torch.optim.Adam,
}
base_optimizer_configs = {
    'Adam (lr=5.5)':     {'lr': 5.5e0,  'betas': (0.9, 0.999),  'eps': 1e-8,  'amsgrad': False},
    'Adam (lr=1.0)':     {'lr': 1.e0,  'betas': (0.9, 0.999),  'eps': 1e-8,  'amsgrad': False},
```

```
    'Adam (lr=1e-1)':     {'lr': 1e-1,  'betas': (0.9, 0.999),  'eps': 1e-8,  'amsgrad': True},
    'Adam (lr=1e-2)':     {'lr': 1e-2,  'betas': (0.9, 0.999),  'eps': 1e-8,  'amsgrad': False},
    'Adam (lr=1e-3)':     {'lr': 1e-3,  'betas': (0.9, 0.999),  'eps': 1e-8,  'amsgrad': False},
}
base_scheduler_specs = {key: (InverseLogTimeLR, {}) for key in base_optimizer_configs.keys()}

adaptive_optimizer_args = {
    'lr_meta': 1e0,
    'curvature_bonus': 0.8,
    'meta_temperature_switch_step': 100,
}
meta_scheduler_spec = None
```

R.3 Common loss surface experiments parameters

```
base_optimizer_classes = {
    'SGD-1':       torch.optim.SGD,
    'SGD-2':       torch.optim.SGD,
    'SGD-3':       torch.optim.SGD,

    'RMSprop-1':   torch.optim.RMSprop,
    'RMSprop-2':   torch.optim.RMSprop,
    'RMSprop-3':   torch.optim.RMSprop,

    'Adagrad-1':   torch.optim.Adagrad,
    'Adagrad-2':   torch.optim.Adagrad,
    'Adagrad-3':   torch.optim.Adagrad,

    'Adadelta-1':  torch.optim.Adadelta,
    'Adadelta-2':  torch.optim.Adadelta,
    'Adadelta-3':  torch.optim.Adadelta,

    'Adam-1':      torch.optim.Adam,
    'Adam-2':      torch.optim.Adam,
    'Adam-3':      torch.optim.Adam,
}
base_optimizer_configs = {
    'SGD-1':      {'lr': 3e-6,  'momentum': 0.0,  'nesterov': False},
    'SGD-2':      {'lr': 1e-5,  'momentum': 0.9,  'nesterov': False},
    'SGD-3':      {'lr': 3e-4,  'momentum': 0.95, 'nesterov': True},

    'RMSprop-1':  {'lr': 3e-3,  'alpha': 0.99, 'eps': 1e-8, 'momentum': 0.0, 'centered': False},
    'RMSprop-2':  {'lr': 1e-2,  'alpha': 0.99, 'eps': 1e-8, 'momentum': 0.2, 'centered': True},
    'RMSprop-3':  {'lr': 2e-2,  'alpha': 0.95, 'eps': 1e-8, 'momentum': 0.5,  'centered': False},

    'Adagrad-1':  {'lr': 5e-2,   'eps': 1e-10, 'lr_decay': 0.0},
    'Adagrad-2':  {'lr': 1.5e-1, 'eps': 1e-10, 'lr_decay': 0.0},
    'Adagrad-3':  {'lr': 5e-1,   'eps': 1e-10, 'lr_decay': 0.0},

    'Adadelta-1': {'lr': 0.50, 'rho': 0.95, 'eps': 1e-6},
    'Adadelta-2': {'lr': 1.00, 'rho': 0.95, 'eps': 1e-6},
    'Adadelta-3': {'lr': 1.55, 'rho': 0.90, 'eps': 1e-6},

    'Adam-1':     {'lr': 8e-4,  'betas': (0.9, 0.999), 'eps': 1e-8, 'amsgrad': False},
    'Adam-2':     {'lr': 3e-3,  'betas': (0.9, 0.999), 'eps': 1e-8, 'amsgrad': True},
```

```
    'Adam-3':     {'lr': 5e-3,  'betas': (0.9, 0.99),  'eps': 1e-8,
    'amsgrad': False},
}
base_scheduler_specs = {key: (InverseLogTimeLR, {}) for key in
    base_optimizer_configs.keys()}

adaptive_optimizer_args = {
    'lr_meta': 2e1,
    'curvature_bonus': .2,
    'meta_temperature_switch_step': 10,
}
meta_scheduler_spec = None
```

R.4 Distribution shift experiments

```
base_optimizer_classes = {
    'SGD':                  torch.optim.SGD,
    "Lion": pytorch_optimizer.Lion,
    'Scion':    pytorch_optimizer.SCION,
    "Soap": pytorch_optimizer.SOAP,
}

base_optimizer_configs = {
    'SGD':                {'lr': 5e-4,  'momentum': 0.0,  'nesterov':
    False},
    'Lion':       {'lr': 5e-4, 'betas': (0.9, 0.99), 'weight_decay':
    0.01},
    'Scion':      {'lr': 5e-4, 'momentum': 0.1, 'weight_decay':5e-4,
    'norm':1},
    'Soap':       {'lr': 5e-4, 'betas': (0.9, 0.99), 'weight_decay':
    0.01, 'precondition_frequency': 10},
}
base_scheduler_specs = {key: (InverseLogTimeLR, {}) for key in
    base_optimizer_configs.keys()}

adaptive_optimizer_args = {
    'lr_meta': 5e-2,
    'curvature_bonus': 0.5,
    'meta_temperature_switch_step': 100,

}
meta_scheduler_spec = None
```

R.5 Loss surface definitions

```
class RosenbrockModel2D(torch.nn.Module):
    def __init__(self, start_point, a=1, b=100):
        super().__init__()
        self.params = torch.nn.Parameter(torch.tensor(start_point,
    dtype=torch.float32))
        self.a = a
        self.b = b

    def forward(self):
        x, y = self.params[0], self.params[1]
        loss = (self.a - x)**2 + self.b * (y - x**2)**2
        return loss

class BoothModel2D(_TestFunction2D):
    def minimizer(self): return (1.0, 3.0)
    def canonical(self, x, y):
        return (x + 2*y - 7)**2 + (2*x + y - 5)**2

```

```python
class GriewankModel2D(_TestFunction2D):
    def minimizer(self): return (0.0, 0.0)
    def canonical(self, x, y):
        return (x*x + y*y) / 4000.0 - torch.cos(x) * torch.cos(y / math.sqrt(2.0)) + 1.0

class HimmelblauModel2D(_TestFunction2D):
    def minimizer(self): return (3.0, 2.0)
    def canonical(self, x, y):
        return (x*x + y - 11.0)**2 + (x + y*y - 7.0)**2

class ThreeHumpCamelModel2D(_TestFunction2D):
    def minimizer(self): return (0.0, 0.0)
    def canonical(self, x, y):
        return 2*x*x - 1.05*x**4 + (x**6)/6.0 + x*y + y*y
```

R.6 Learning rate scheduler definitions

```python
class InverseTimeLR(_LRScheduler):
    def __init__(self, optimizer, decay_rate=1.0, last_epoch=-1):
        self.decay_rate = decay_rate
        super(InverseTimeLR, self).__init__(optimizer, last_epoch)

    def get_lr(self):
        return [base_lr / (1 + self.decay_rate * self._step_count)
                for base_lr in self.base_lrs]

class InverseSqrtTimeLR(_LRScheduler):
    def __init__(self, optimizer, decay_rate=1.0, last_epoch=-1):
        self.decay_rate = decay_rate
        super(InverseSqrtTimeLR, self).__init__(optimizer, last_epoch)

    def get_lr(self):
        return [base_lr / math.sqrt(1 + self.decay_rate * self._step_count)
                for base_lr in self.base_lrs]


class InverseLogTimeLR(_LRScheduler):
    def __init__(self, optimizer, log_offset=math.e, decay_rate=1.0, last_epoch=-1):
        if log_offset <= 1.0:
            raise ValueError("log_offset must be > 1.0 to ensure a positive denominator.")

        self.log_offset = log_offset
        self.decay_rate = decay_rate
        super(InverseLogTimeLR, self).__init__(optimizer, last_epoch)

    def get_lr(self):
        t = self.last_epoch

        return [base_lr / math.log(self.log_offset + self.decay_rate * t)
                for base_lr in self.base_lrs]
```