ADAMULT: ADAPTIVE DUAL OPTIMIZATION FOR LAGRANGIAN CONSTRAINED DEEP LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce ADAMULT, a dual optimizer for Lagrangian constrained deep learning that produces feasible, well-performing solutions out of the box, at its one default hyperparameter configuration, across a wide range of tasks. This removes the costly trial-and-error tuning process of dual hyperparameters that constrained deep learning otherwise requires. ADAMULT normalizes each constraint by a running estimate of its scale, so the multipliers are driven by a dimensionless signal; updates the multipliers multiplicatively, so they traverse orders of magnitude quickly; and extrapolates each multiplier along its current violation, which, under suitable step sizes and damping, makes the primal–dual dynamics locally stable at every regular local constrained minimizer. On six tasks, ADAMULT at its default is feasible on every seed of every task and matches the performance of baselines that were tuned separately for each task. Held to a single configuration across tasks, every existing dual optimizer we test is infeasible on at least one of them, or feasible at a loss in performance. To our knowledge, ADAMULT is the first dual optimizer that transfers reliably across tasks without tuning.

1 INTRODUCTION

Constrained formulations have become a standard way to impose requirements on deep models such as fairness (Cotter et al., 2019), sparsity (Gallego-Posada et al., 2022), safety (Stooke et al., 2020), and alignment (Dai et al., 2024). These constrained deep learning problems are typically solved with Lagrangian methods (Arrow et al., 1958), which introduce one Lagrange multiplier per constraint and perform gradient descent on the model and ascent on the multipliers.

The multipliers introduce a new choice of optimizer, the *dual optimizer*, with its associated hyperparameters, which must be selected and tuned *for every* new constrained task. This involves a costly trial-and-error tuning process, where each trial requires fully training a model.

We ask whether a dual optimizer can achieve for the multipliers what Adam achieved for the model: work well out of the box, at one default configuration, across a wide range of tasks.

Existing dual optimizers fail at this. We argue in §2 that this is because each fails at least one of two requirements. *Stability*: a dual optimizer must ensure local convergence to every regular solution of the constrained problem. *Transferability*: its hyperparameters must not need retuning when the task changes. In particular, standard projected gradient ascent (Arrow et al., 1958) fails *both*: a solution of the constrained problem can be an unstable fixed point of its primal–dual dynamics (Bertsekas, 2016, §4.2.1), and its step size multiplies raw constraint values, so even a change of constraint units may require retuning it, let alone a change of task.

We introduce ADAMULT (*Adaptive Multipliers*, §3), a dual optimizer that meets *both* requirements. It extrapolates each multiplier along its current violation, making every regular local constrained minimizer locally stable (Proposition 2), and it normalizes each constraint by a running estimate of its scale, ensuring that the primal iterates of a training run are invariant to constraint rescaling (Proposition 4), so that one hyperparameter configuration serves constraints of any scale. Its single default hyperparameter configuration, Eq. (5), makes per-task tuning unnecessary (§4).

At that default, ADAMULT runs unchanged on six tasks, from a single recall constraint to $50{,}000$ per-example loss bounds and the safety-constrained fine-tuning of a 7B language model: it is feasible,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

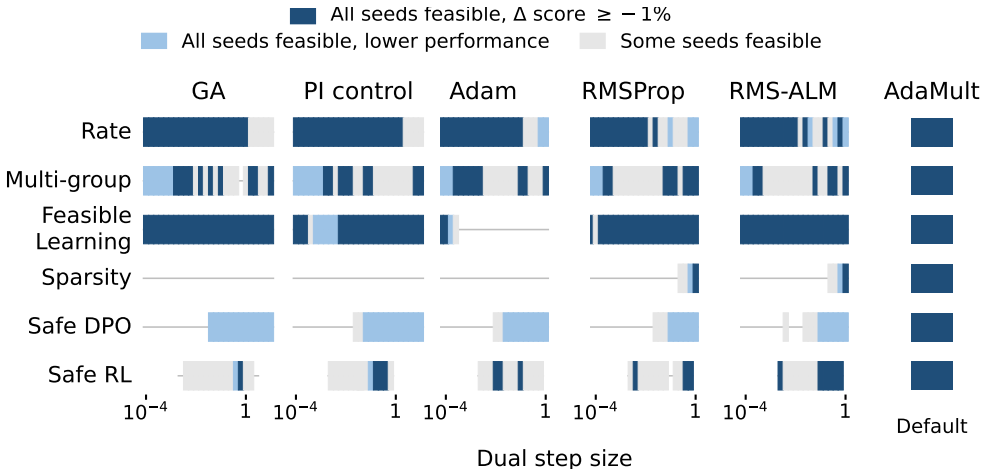

Figure 1: **The dual step size that works changes with the task; AdaMult's default works on all six.** Each row is a task and each column a baseline, and each bar spans the baseline's step-size grid. The fill says what a step achieved: *dark* where it is feasible on all five seeds with Δ score $\geq -1\%$, *light* where it is feasible on all seeds but falls outside that band, *gray* where it is feasible on some seeds but not all, and *blank* where it is infeasible on every seed. The last column is AdaMult at its default, Eq. (5), feasible on every seed of every task. Feasibility and performance as in Table 1; PI control at unit proportional gain; setup in §4.

within tolerance, on every seed of every task, with performance comparable to that of dual optimizers tuned separately for each task (Table 1). Tuned to a single configuration across all tasks, no tested baseline is feasible on every task at comparable performance (Figure 1, §4.1). To our knowledge, AdaMult is the first dual optimizer to succeed at this.

Contributions.

1. We propose AdaMult (§3), an off-the-shelf dual optimizer for constrained deep learning that works across a wide range of tasks at its default hyperparameter configuration, and provide a `torch.optim` implementation of it.
2. We prove that AdaMult is a principled method for constrained optimization: its fixed points are KKT points of the constrained problem and all regular constrained minimizers are locally stable under it (Propositions 1 and 2 in §3.1).
3. We prove that the primal iterates under AdaMult are invariant to constraint rescaling (Proposition 4 in §3.2), a necessary condition for one configuration to transfer across tasks.
4. We show that AdaMult, at its single default configuration, produces feasible, well-performing models across a wide range of machine learning tasks, matching the baselines that stay feasible when tuned separately for each task. No tested baseline held to one configuration is feasible on every task at comparable performance (§4).

Related work. Each ingredient of AdaMult exists on its own: extrapolation, in PI control and the augmented Lagrangian method (Hestenes, 1969; Powell, 1969; Rockafellar, 1973; Stooke et al., 2020), with step size and damping coefficient tuned per problem; RMS normalization, in primal optimizers (Kingma and Ba, 2015; Tieleman and Hinton, 2012), which on the dual ensures neither stability (Proposition 3) nor invariance (Proposition 5); and multiplicative multiplier updates, in the exponential method of multipliers (Kort and Bertsekas, 1976; Tseng and Bertsekas, 1993) and in ML applications with inequality constraints (Beck and Teboulle, 2003; Cotter et al., 2019; Rezende and Viola, 2018), without normalization. Parameter-free optimizers (Defazio and Mishchenko, 2023; Ivgi et al., 2023; Mishchenko and Defazio, 2024) remove step-size tuning for convex minimization, but they estimate the distance to a minimizer, which a dual objective linear in the multipliers puts on the boundary or at infinity. Closest to AdaMult, RMS-ALM (Basir and Senocak, 2023), extended as CAPU by Hu et al. (2026), stated for equality constraints only, and its relative RCML (Liu et al., 2026) normalize the dual step and extrapolate, but both apply the multiplier to the raw constraint, so neither is invariant to constraint units (§3.2). RMS-ALM's default also fails to transfer across

tasks in our single-primal-step comparisons (Table 1). AdaMult combines the three ingredients and ensures invariance to constraint units, allowing one default to transfer across many tasks.

2 Background

We consider (possibly non-convex) differentiable constrained problems

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^d} f(\boldsymbol{\theta}) \quad \text{s.t.} \quad g_i(\boldsymbol{\theta}) \le 0, \;\; i = 1,\dots,m, \quad h_j(\boldsymbol{\theta}) = 0, \;\; j = 1,\dots,n, \tag{1}$$

stacked as $\boldsymbol{g}(\boldsymbol{\theta}) \in \mathbb{R}^m$ and $\boldsymbol{h}(\boldsymbol{\theta}) \in \mathbb{R}^n$. In deep learning practice, these are generally solved with Lagrangian methods (Arrow et al., 1958), which can be applied without assumptions beyond (sub)differentiability (Gallego-Posada et al., 2025; Ramirez et al., 2025a). These involve solving

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\lambda}\ge\mathbf{0},\,\boldsymbol{\mu}} \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\lambda},\boldsymbol{\mu}) \triangleq f(\boldsymbol{\theta}) + \boldsymbol{\lambda}^\top \boldsymbol{g}(\boldsymbol{\theta}) + \boldsymbol{\mu}^\top \boldsymbol{h}(\boldsymbol{\theta}), \tag{2}$$

where $\mathcal{L}$ is the Lagrangian of Eq. (1), and $\boldsymbol{\lambda} \ge \mathbf{0}$ and $\boldsymbol{\mu} \in \mathbb{R}^n$ are the multipliers for the inequality and equality constraints, respectively. We call $\boldsymbol{\theta}$ the primal variables and $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ the dual variables.

Min-max of $\mathcal{L}$ yield solutions of Eq. (1) (Bertsekas, 2016, Props. 4.3.1 & 4.3.2). A standard algorithm for finding such min-max is alternating gradient descent–ascent (GDA),

$$\boldsymbol{\lambda}_{t+1} = \texttt{DualAscent}(\boldsymbol{\lambda}_t, \boldsymbol{g}(\boldsymbol{\theta}_t)), \quad \boldsymbol{\mu}_{t+1} = \texttt{DualAscent}(\boldsymbol{\mu}_t, \boldsymbol{h}(\boldsymbol{\theta}_t)), \tag{3a}$$

$$\boldsymbol{\theta}_{t+1} = \texttt{PrimalDescent}\big(\boldsymbol{\theta}_t,\; \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1})\big), \tag{3b}$$

for $i = 1,\dots,m$ and $j = 1,\dots,n$, where `PrimalDescent` is a step of any primal optimizer, such as Adam (Kingma and Ba, 2015), and `DualAscent` is a step of a *dual optimizer*. Because the Lagrangian's gradient with respect to a multiplier is its constraint value, ascent raises that multiplier (in magnitude) while the constraint is violated—increasing the pressure on the primal to satisfy it—and lowers it otherwise, letting the primal focus on the objective.

We argue that a dual optimizer must satisfy two requirements to serve as a dependable default, one that works robustly across tasks at a single configuration. *Stability*: it must make every regular solution of Eq. (1) (satisfying the strict-complementarity, second-order and constraint-qualification conditions of Assumptions 1 to 3 in Appendix A.1) a locally stable fixed point of Eq. (3). Otherwise, the dynamics may oscillate around the constraint boundary (Stooke et al., 2020) or fail to converge to some solutions (Bertsekas, 2016, §4.2.1). *Transferability*: much like Adam for unconstrained optimization, it must require little to no hyperparameter tuning across problems; otherwise every new task requires an expensive trial-and-error dual hyperparameter search, where each trial costs a full model training run.

The most widely adopted dual optimizer is projected gradient ascent (Arrow et al., 1958), $\lambda_{i,t+1} = [\lambda_{i,t} + \eta\, g_i(\boldsymbol{\theta}_t)]_+$ and $\mu_{j,t+1} = \mu_{j,t} + \eta\, h_j(\boldsymbol{\theta}_t)$, with dual step size $\eta > 0$ and $[\,\cdot\,]_+$ a projection onto the non-negative orthant. However, dual gradient ascent fails both requirements. It fails stability: a solution of Eq. (1), paired with its optimal multipliers, may be an *unstable* fixed point of Eq. (3) (Bertsekas, 2016, §4.2.1). It fails transferability: its step size multiplies raw constraint values, whose scale varies across constraints and tasks, so a value tuned for one need not transfer to another.

Dual gradient ascent's instability is addressed by another standard method: PI control (Stooke et al., 2020) as the dual optimizer in Eq. (3a), or equivalently (Ramirez and Lacoste-Julien, 2026) the augmented Lagrangian method (Hestenes, 1969; Powell, 1969; Rockafellar, 1973):

$$\begin{aligned} \hat\lambda_{i,t+1} &= \big[\hat\lambda_{i,t} + \eta\, g_i(\boldsymbol{\theta}_t)\big]_+, \quad \hat\mu_{j,t+1} = \hat\mu_{j,t} + \eta\, h_j(\boldsymbol{\theta}_t), \\ \lambda_{i,t+1} &= \big[\hat\lambda_{i,t+1} + \rho\, g_i(\boldsymbol{\theta}_t)\big]_+, \quad \mu_{j,t+1} = \hat\mu_{j,t+1} + \rho\, h_j(\boldsymbol{\theta}_t), \end{aligned} \tag{ALM/PI}$$

with damping coefficient $\rho > 0$. The dual iterates $\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}$ are *extrapolated* along the current violation, and the extrapolated $\boldsymbol{\lambda}, \boldsymbol{\mu}$, not the iterates, enter the primal update in Eq. (3b). For ρ large enough and small enough step sizes, Eq. (ALM/PI) ensures local stability of every regular local constrained minimizer and damps the oscillations of dual gradient ascent (Ramirez and Lacoste-Julien, 2026).

However, Eq. (ALM/PI) still fails transferability, requiring both the step size η and the damping coefficient ρ to be tuned per task. AdaMult (§3), by contrast, meets both requirements. For stability, it makes every regular local constrained minimizer a locally stable fixed point (Proposition 2). For transferability, its dual update is invariant to the scale of each constraint (Proposition 4), so one default configuration can work across a wide range of tasks, as §4 demonstrates empirically.

3 ADAMULT

ADAMULT (*Adaptive Multipliers*, Algorithm ADAMULT) is a dual optimizer: like Adam on the primal, it holds per-constraint state and, given the current constraint values, returns the multipliers $\boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1}$ of Eq. (3a), to be plugged into the plain Lagrangian of Eq. (3b) (see Algorithm 1). A practitioner can substitute dual gradient ascent with ADAMULT; we release it as a `torch.optim` optimizer.[1] ADAMULT is both stable, making every regular local constrained minimizer locally stable for large enough damping and small enough step sizes (§3.1), and transferable, being invariant to constraint rescaling (§3.2) and working out of the box across a wide range of tasks at its one default hyperparameter configuration (Table 1 in §4.1).

Without the clips, floors and bias correction that Algorithm ADAMULT adds, each inequality-constraint multiplier is updated through its log-state z_t as

$$\underbrace{\bar{g}_t = g_t / a_t}_{\text{① normalize}}, \quad \underbrace{z_t = z_{t-1} + r\,\bar{g}_t}_{\text{② multiplicative ascent}}, \quad \underbrace{\tilde{z}_t = z_t + p\,\bar{g}_t}_{\text{③ extrapolate}}, \quad \underbrace{\lambda_{t+1} = e^{\tilde{z}_t} / a_t}_{\text{④ return}}, \tag{4}$$

with a_t the running RMS scale of g_t; $r > 0$ the dual step size; and $\tilde{z}_t$ the extrapolated log-state and $p \geq 0$ the damping coefficient. ADAMULT thus does: ① normalization by a running RMS estimate of the constraint's scale (Tieleman and Hinton, 2012), so that the dual update sees approximately dimensionless signals; ② multiplicative ascent (Beck and Teboulle, 2003; Kivinen and Warmuth, 1997), allowing multipliers to quickly traverse orders of magnitude; ③ extrapolation along the current violation, as in Eq. (ALM/PI), thus stabilizing the primal-dual dynamics (Proposition 2 in §3.1). The readout in ④ reverses the normalization of ① for the output, so that the returned multiplier matches the units of the raw constraint in the subsequent primal update.

Multipliers of equality constraints follow the same procedure with $\sinh(\cdot) = (e^{\cdot} - e^{-\cdot})/2$ in place of the exponential in ④, which allows them to be signed while keeping the exponentially fast traversal.

Steps ① and ④ rest on a property of constrained problems: rescaling the objective or a constraint leaves the solution unchanged and rescales the optimal multipliers accordingly.

Lemma 1 (Equivariance of the optimal multipliers)**.** *Let $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ be a KKT point of Eq. (1) at which LICQ holds (Definition 1 and Assumption 3 in Appendix A.1), and $\varrho, \alpha_i, \gamma_j > 0$. Then $(\boldsymbol{\theta}^\star, \{\varrho\lambda_i^\star/\alpha_i\}, \{\varrho\mu_j^\star/\gamma_j\})$ is the unique KKT point at $\boldsymbol{\theta}^\star$ of the problem with objective ϱf and constraints $\alpha_i g_i \leq 0$, $\gamma_j h_j = 0$. Proof.* See proof of Lemma 1 in Appendix A.2.

ADAMULT's internal state e^{z_t} is a multiplier for the *normalized* constraint $\bar{g}_t = g_t/a_t$. ADAMULT's output, however, must multiply the *raw* g_t in the Lagrangian $\mathcal{L}$ that the primal step differentiates (line 4 of Algorithm 1), so step ④ divides the extrapolated state $e^{\tilde{z}_t}$ back by the RMS scale. The primal's term $\lambda_{t+1}\, g_t = (e^{\tilde{z}_t}/a_t)\, g_t$ then corresponds to the dual's $e^{\tilde{z}_t}\, \bar{g}_t$, the same product in normalized units for the dual and the problem's units for the primal. The constraint's units therefore never reach the dual state, and the primal's Lagrangian does not move when they change. Skipping step ④ is what makes RMSPROP and RMS-ALM fail transferability (§3.2).

Guards. The implementation of ADAMULT in Algorithm ADAMULT includes several guards against the numerical failure modes of RMS normalization and multiplicative updates. The running peak $\hat{v}_{i,t}^{\max}$ and δ and ϵ keep the normalization numerically stable when constraint values approach zero, as they do near convergence. The clip on the normalized constraint $\bar{g}_{i,t}$ bounds how far the log-state moves in one step, a move the exponential would otherwise amplify. The $z_{\max}$ clips on the dual log-states keep the multiplier magnitudes bounded.[2] For inequalities, the $z_{\min}$ floor keeps the multiplier from decaying so far that it recovers slowly if the constraint starts being violated again.[3]

Computational cost. Whatever the choice of dual optimizer in Algorithm 1, a training step evaluates the objective and the constraints once and differentiates the Lagrangian once. ADAMULT adds no additional forward or backward passes through the model, only a handful of scalar operations per

[1] See code in the supplementary material.

[2] Differentiable problems that fail the constraint qualification of Mangasarian and Fromovitz (1967) have unbounded optimal multipliers, if any (Gauvin, 1977); such problems are outside our scope.

[3] Because of $z_{\min}$, inequality multipliers never reach exactly zero, unlike under additive updates with a projection; at a small enough $z_{\min}$, however, this is numerically negligible.

Algorithm ADAMULT The ADAMULT dual optimizer, applied independently to each inequality (left) and equality (right), with $g_{i,t} = g_i(\boldsymbol{\theta}_t)$ and $h_{j,t} = h_j(\boldsymbol{\theta}_t)$; $\pm z_{\max}$ abbreviates $[-z_{\max}, z_{\max}]$; `sg` is stop-gradient.

`AdaMult.init`: per-constraint states $v_{i,0} = \widehat{v}^{\max}_{i,0} = z_{i,0} = 0$, $u_{j,0} = \widehat{u}^{\max}_{j,0} = w_{j,0} = 0$; hyperparameters: RMS decay $\beta \in [0, 1)$, floors $\epsilon > 0$ (absolute) and $\delta > 0$ (relative), clip radius $c > 0$, dual step size $r > 0$, damping coefficient $p \geq 0$, log-state bounds $z_{\min} \leq 0 \leq z_{\max}$ (Eq. (5)).

`AdaMult.step`: constraint values $\boldsymbol{g}(\boldsymbol{\theta}_t), \boldsymbol{h}(\boldsymbol{\theta}_t) \mapsto$ multipliers $\boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1}$

Inequality g_i, $\lambda_i \geq 0$	Equality h_j, $\mu_j \in \mathbb{R}$
$v_{i,t} = \beta v_{i,t-1} + (1-\beta)\, g_{i,t}^2$	$u_{j,t} = \beta u_{j,t-1} + (1-\beta)\, h_{j,t}^2$
$\widehat{v}_{i,t} = v_{i,t}/(1-\beta^t)$	$\widehat{u}_{j,t} = u_{j,t}/(1-\beta^t)$
$\widehat{v}^{\max}_{i,t} = \max\big(\widehat{v}^{\max}_{i,t-1},\, \widehat{v}_{i,t}\big)$	$\widehat{u}^{\max}_{j,t} = \max\big(\widehat{u}^{\max}_{j,t-1},\, \widehat{u}_{j,t}\big)$
$a_{i,t} = \mathtt{sg}\left(\max(\sqrt{\widehat{v}_{i,t}},\, \delta\sqrt{\widehat{v}^{\max}_{i,t}}) + \epsilon\right)$	$b_{j,t} = \mathtt{sg}\left(\max(\sqrt{\widehat{u}_{j,t}},\, \delta\sqrt{\widehat{u}^{\max}_{j,t}}) + \epsilon\right)$
$\bar{g}_{i,t} = \mathtt{clip}\left(g_{i,t}/a_{i,t},\, -c, c\right)$	$\bar{h}_{j,t} = \mathtt{clip}\left(h_{j,t}/b_{j,t},\, -c, c\right)$
$z_{i,t} = \mathtt{clip}\left(z_{i,t-1} + r\,\bar{g}_{i,t},\, z_{\min}, z_{\max}\right)$	$w_{j,t} = \mathtt{clip}\left(w_{j,t-1} + r\,\bar{h}_{j,t},\, \pm z_{\max}\right)$
$\tilde{z}_{i,t} = \mathtt{clip}\left(z_{i,t} + p\,\bar{g}_{i,t},\, z_{\min}, z_{\max}\right)$	$\tilde{w}_{j,t} = \mathtt{clip}\left(w_{j,t} + p\,\bar{h}_{j,t},\, \pm z_{\max}\right)$
$\lambda_{i,t+1} = \exp\big(\tilde{z}_{i,t}\big)/a_{i,t}$	$\mu_{j,t+1} = \sinh\big(\tilde{w}_{j,t}\big)/b_{j,t}$

Guard: until a constraint's first nonzero observation ($\widehat{v}^{\max}_{i,t} = 0$), its scale is $a_{i,t} = 1$ (likewise $b_{j,t}$).

return $\boldsymbol{\lambda}_{t+1}$, $\boldsymbol{\mu}_{t+1}$ (detached), for the Lagrangian Eq. (2) of the primal step Eq. (3b).

Algorithm 1 A `PyTorch`-style constrained training loop, Eq. (3) with ADAMULT as the dual optimizer.

Input: model parameters $\boldsymbol{\theta}$; `primal_opt` on $\boldsymbol{\theta}$ (e.g. `Adam`); `dual_opt` = ADAMULT (Algorithm ADAMULT, Eq. (5))

for each training step **do**
  $\ell,\ \boldsymbol{g},\ \boldsymbol{h} \leftarrow f(\boldsymbol{\theta}),\ \boldsymbol{g}(\boldsymbol{\theta}),\ \boldsymbol{h}(\boldsymbol{\theta})$ ▷ *forward pass*
  $\boldsymbol{\lambda}, \boldsymbol{\mu} \leftarrow$ `dual_opt.step`$(\boldsymbol{g}, \boldsymbol{h})$ ▷ *detached (Algorithm ADAMULT)*
  $\mathcal{L} \leftarrow \ell + \langle \boldsymbol{\lambda}, \boldsymbol{g} \rangle + \langle \boldsymbol{\mu}, \boldsymbol{h} \rangle$ ▷ *plain Lagrangian (Eq. (2))*
  `primal_opt.zero_grad()`
  $\mathcal{L}$`.backward()` ▷ $\nabla_{\boldsymbol{\theta}} \mathcal{L}$
  `primal_opt.step()` ▷ *primal step*
end for

constraint, as dual gradient ascent and PI control also do. It carries three state variables per constraint, $v_{i,t}$, $\widehat{v}^{\max}_{i,t}$ and $z_{i,t}$, against gradient ascent's one and dual PI control's two.

Defaults. ADAMULT ships a single default configuration. The multiplier log-states are initialized to zero, corresponding to multipliers $\lambda_0 = e^{z_0} = 1$ and $\mu_0 = \sinh(w_0) = 0$ for inequalities and equalities, respectively. The RMS memory β and the floor ϵ match `PyTorch`'s RMSPROP defaults (Paszke et al., 2019); the dual step size r, the damping coefficient p and the remaining entries were chosen during development (see Appendix B.2):

$$r = 0.01,\ \ p = 0.1,\ \ \beta = 0.99,\ \ \epsilon = 10^{-8},\ \ \delta = 10^{-2},\ \ c = 10,\ \ z_{\min} = -10,\ \ z_{\max} = 10. \quad (5)$$

3.1 STABILITY

We show that ADAMULT is a principled dual optimizer: the fixed points of Algorithm 1 correspond to the KKT points of Eq. (1), up to the $z_{\min}$ floor on the inactive-inequality multipliers (Proposition 1), and every regular local constrained minimizer is locally stable under it, up to the same floor (Proposition 2). ADAMULT is thus principled and meets the stability requirement of §2.

Setting. We mark quantities of a KKT point with $\star$, and those at a fixed point of Algorithm 1 with $\circ$. Let $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ be a KKT point of Eq. (1) (Definition 1 in Appendix A.1). We call $\boldsymbol{\theta}^\star$ a *regular local constrained minimizer* if $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ satisfies strict complementarity, SOSC, and LICQ (Assumptions 1 to 3 in Appendix A.1). Propositions 1 and 2 assume a plain gradient descent

primal step and simplify Algorithm AdaMult in three ways, none of which changes fixed points with inactive upper clips. The RMS scales are held at constants a_i, b_j, the values they converge to near a fixed point; the bias correction is dropped ($\widehat{v}_i = v_i$), since β^t is negligible after the first few hundred steps; and the upper clip on the dual log-states is removed ($z_{\max} = \infty$), since choosing $z_{\max}$ above every z_i° and $|w_j^\circ|$ keeps the clip inactive near the fixed point being analyzed.

Proposition 1 (KKT consistency of AdaMult)**.** *The fixed points of Algorithm 1 correspond to the triples $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ that satisfy the KKT conditions of Eq. (1) with $e^{z_{\min}}/a_i$ in place of 0 in dual feasibility and complementary slackness (Eq. (10) in Appendix A.3), where $\mu_j^\circ = \sinh(w_j^\circ)/b_j$, $\lambda_i^\circ = e^{z_i^\circ}/a_i$ if $g_i(\boldsymbol{\theta}^\circ) = 0$, and $\lambda_i^\circ = e^{z_{\min}}/a_i$ if $g_i(\boldsymbol{\theta}^\circ) < 0$. As $z_{\min} \to -\infty$, every limit point of $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ is a KKT point of Eq. (1). Proof.* We adapt the fixed-point argument of the exponential method of multipliers (Bertsekas, 1982, Ch. 5); see proof of Proposition 1 in Appendix A.3.

Proposition 1 extends to any primal optimizer whose fixed points are those where $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \mathbf{0}$, such as momentum SGD or Adam without weight decay or bias correction (Remark 1 in Appendix A.3).

Proposition 2 (Local stability of AdaMult)**.** *Under the preceding simplifications, for every regular local constrained minimizer $\boldsymbol{\theta}^\star$, Algorithm 1 has a locally stable fixed point (Definition 2 in Appendix A.1) that corresponds to a triple $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ near $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$, where the multipliers of the inequalities inactive at $\boldsymbol{\theta}^\star$ rest on the floor $e^{z_{\min}}/a_i$ instead of 0. This holds for all damping values p beyond some finite threshold $\bar{p} \geq 0$, sufficiently small primal and dual step sizes, and every sufficiently negative $z_{\min}$. The $z_{\min}$ floor also shifts the rest of the triple, and both the floor and its induced shift vanish as $z_{\min} \to -\infty$, where $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) \to (\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$. Proof.* Adapts the classical local analysis of multiplier methods (Bertsekas, 1982, Ch. 2); see proof of Proposition 2 in Appendix A.4.

The extrapolation of AdaMult stabilizes the dynamics because, within each primal step, it acts as the augmented-Lagrangian penalty (Ramirez and Lacoste-Julien, 2026), which convexifies the Lagrangian in $\boldsymbol{\theta}$ around a regular local constrained minimizer (Bertsekas, 1982, Ch. 2). Proposition 2, however, holds *only* for back-to-back primal and dual updates: if the Lagrangian is fully or approximately minimized with respect to $\boldsymbol{\theta}$ before each dual update, any form of dual extrapolation, and with it AdaMult, is no longer guaranteed to stabilize the dynamics. Extending AdaMult's stabilization and scale invariance to methods with inner primal solves remains future work.

Why not RMSProp or Adam? RMS normalization makes the dual state approximately unit-free, easing step-size tuning in practice (Table 1). It does not, however, ensure stability in general (Proposition 3). Momentum, as in Adam, works *against* stability: it keeps pushing a multiplier after the constraint becomes feasible, exacerbating dual oscillations and potentially de-stabilizing *otherwise* stable solutions (Sohrabi et al., 2024).

Proposition 3 (Dual RMS normalization does not ensure stability)**.** *Consider the problem*

$$\min_{x,y} \quad \frac{x^4}{4} - \frac{\kappa x^2}{2} + \frac{y^2}{2} \qquad \text{subject to } \gamma x = 0, \tag{6}$$

where $\kappa, \gamma > 0$, whose unique minimizer $(x, y) = (0, 0)$ is regular, satisfying LICQ and SOSC with multiplier $\mu^\star = 0$. Under plain primal gradient descent paired with dual RMSProp *ascent without extrapolation (Eq. (34) in Appendix A.5), this solution is not locally stable at any curvature κ or constraint scale γ, whatever the choice of primal and dual step sizes, RMS decay $\beta \in [0, 1)$ and relative floor $\delta \in (0, 1)$. Proof.* See proof of Proposition 3 in Appendix A.5.

AdaMult, by contrast, makes the same minimizer locally stable whenever its (normalized) damping exceeds the curvature κ and its primal step size is small enough, two conditions that *do not involve* the constraint's scale γ (Proposition 6 in §3.2).

3.2 Transferability via Scale Invariance

For methods that do not normalize the constraints, such as dual gradient ascent and Eq. (ALM/PI), the dual step size must account both for the constraints' scale and for the problem's dynamics. Rescaling a constraint, let alone changing the problem, may require retuning the dual step size and damping coefficient. Thus, invariance to constraint rescaling is a necessary, although not sufficient, condition

for hyperparameters to transfer across problems. Proposition 4 shows that AdaMult is invariant to constraint units, producing the same primal trajectory under any constraint rescaling:

Proposition 4 (Scale invariance of AdaMult)**.** *Run Algorithm 1 on Eq. (1) and, from the same primal and dual initializations, on the problem with every constraint rescaled, $g'_i = \alpha_i g_i$ and $h'_j = \gamma_j h_j$ with $\alpha_i, \gamma_j > 0$ for all i, j. Then both runs produce the same primal iterates, $\boldsymbol{\theta}'_t = \boldsymbol{\theta}_t$, and the multiplier iterates of the second run are $\lambda'_{i,t+1} = \lambda_{i,t+1}/\alpha_i$ and $\mu'_{j,t+1} = \mu_{j,t+1}/\gamma_j$ for all $t \geq 1$ and for any choice of deterministic primal optimizer, provided every constraint is nonzero at initialization and $\epsilon = 0$. Proof.* See proof of Proposition 4 in Appendix A.6.

In particular, if AdaMult converges at a given configuration, it converges at the same configuration under any constraint rescaling, whichever deterministic primal optimizer it is paired with.

Why not RMSProp? Normalizing the dual step, as vanilla RMSProp does, produces a multiplier in normalized units, the same whether the constraint is g_i or $\alpha_i g_i$. Applied to a *raw*, rescaled constraint $\alpha_i g_i$, however, it contributes $\alpha_i \lambda_i g_i$ to the Lagrangian where the unscaled run had $\lambda_i g_i$. The primal trajectory under dual RMSProp therefore depends on the constraint's units. In contrast, AdaMult's step ④ (Eq. (4)) divides the output multiplier by the RMS scale, which rescales it by $1/\alpha_i$, making the Lagrangian term invariant and allowing Proposition 4 to hold.

RMS-ALM (Basir and Senocak, 2023), which stacks vanilla RMSProp with extrapolation, does not buy invariance either. Proposition 5 shows that, on the two-dimensional problem of Eq. (6), the hyperparameters that allow it to converge change as the constraint is rescaled:

Proposition 5 (Dual RMS normalization with extrapolation does not ensure scale invariance)**.** *Pair one plain primal gradient descent step of size $\eta_\theta \in (0, 2)$ with* RMS-ALM*'s dual update (Eq. (35) in Appendix A.5) on the two-dimensional problem of Eq. (6), with dual step size $\eta > 0$ and damping coefficient $\rho > 0$. At a constraint scale γ, with its RMS floor settled at γA, the minimizer is locally stable if and only if $\rho > \kappa A/\gamma$ and $\eta + 2\rho < (4/\eta_\theta + 2\kappa)A/\gamma$. From any initialization whose first primal step leaves $x \neq 0$, any fixed choice of η and ρ is unstable at every small enough constraint scale. Proof.* See proof of Proposition 5 in Appendix A.5.

RMS-ALM's (η, ρ) that converge exist at every γ; however, they move with the constraint's scale. In contrast, AdaMult is both stable and transferable on Eq. (6):

Proposition 6 (AdaMult on the two-dimensional problem)**.** *Pair one plain primal gradient descent step of size $\eta_\theta \in (0, 2)$ with* AdaMult*'s dual update (Algorithm AdaMult) on the two-dimensional problem of Eq. (6), with $\epsilon = 0$, and let the relative floor of its RMS scale settle at γA, with $A > 0$ independent of γ. The minimizer is locally stable if and only if $p > \kappa A^2$ and $\eta_\theta\big((r+2p)/A^2 - 2\kappa\big) < 4$, two conditions that do not involve γ. Proof.* See proof of Proposition 6 in Appendix A.5.

A single configuration of RMS-ALM therefore cannot transfer across constraint scales, while one configuration of AdaMult transfers across all of them.

AdaMult still faces two obstacles to a transferable default. First, the damping that stability requires (Proposition 2) depends on the *curvature* of the Lagrangian, which constraint values by themselves do not reveal, so no default p is guaranteed to suffice in general. Even so, tuning AdaMult's p is easier than tuning the raw ρ of Eq. (ALM/PI), which must track the curvature *and* the constraint's scale, while p tracks only the curvature. Second, rescaling the objective rescales the optimal multipliers (Lemma 1), possibly by orders of magnitude. However, AdaMult's multiplicative updates traverse orders of magnitude quickly, mitigating the cost of such rescalings. Whether AdaMult's default transfers in practice is thus an empirical question, and §4 shows it does across many tasks.

4 Experiments

AdaMult's single default (Eq. (5)) finds feasible, well-performing solutions across six machine learning tasks (§4.1). Against dual optimizers tuned separately for *each* task, it performs on par. Against the same optimizers restricted to *one* configuration across tasks, it comes out ahead: each of them is infeasible on at least one task, or feasible only at a loss in performance relative to AdaMult. Ablating any one of AdaMult's three ingredients hurts transferability (§4.2).

Table 1: **AdaMult's default against tuned baselines on six tasks.** *Top*: each baseline at its best configuration *for each* task. *Middle*: each baseline at its best single configuration *across all* tasks. *Bottom*: methods at their fixed defaults. *Across six tasks* (left): feasibility and Δ score averaged over all tasks. *Worst task* (right): feasibility and Δ score on each method's worst task. *Best configuration*: most feasible seeds, with ties broken by mean score within a task or average within-task score rank across tasks (Appendix B.2). *Worst task*: feasible on the fewest seeds, with ties broken by the lowest Δ score. *Feasible*: proportion of feasible seeds. *Default*: a method's reported default configuration. Δ *score*: mean score difference from AdaMult, as a percentage of its score for accuracy or its gain over the initial policy for reward (Appendix C.1). All five seeds count. The per-task results behind every entry are in Tables 10 and 11 of Appendix C. Both AdaMult variants use the defaults in Eq. (5).

		Across six tasks		Worst task		
	Method	Feasible	Δ score	Task	Feasible	Δ score
Tuned Per-Task	GA	83%	−3.51%	Sparsity	0/5	+0.66%
	PI control	100%	−6.93%	Safe DPO	5/5	−43.52%
	Adam	83%	−4.60%	Sparsity	0/5	+0.43%
	RMSProp	100%	−0.41%	Safe DPO	5/5	−4.14%
	RMS-ALM	100%	−1.70%	Safe DPO	5/5	−11.89%
Tuned Across Tasks	GA	80%	−4.41%	Sparsity	0/5	−1.22%
	PI control	83%	−11.35%	Sparsity	0/5	−2.35%
	Adam	67%	−13.01%	Sparsity	0/5	−5.78%
	RMSProp	100%	−6.41%	Safe DPO	5/5	−38.09%
	RMS-ALM	97%	−7.49%	Multi-group	4/5	+0.49%
Fixed Defaults	RMS-ALM default	53%	+2.13%	Sparsity	0/5	+0.21%
	Additive AdaMult default	100%	−3.14%	Safe DPO	5/5	−10.36%
	AdaMult default	100%	+0.00%	None	5/5	—

Tasks. Appendix B specifies each task's training pipeline and primal hyperparameters. We keep each pipeline fixed and vary only the dual optimizer. **Rate**: classification on Adult (Kohavi, 1996) under training recall ≥ 0.9 (one constraint; Cotter et al., 2019). **Multi-group**: classification on Adult under a training recall floor per race group, $\text{recall}_k \geq 0.7$ (five constraints; Agarwal et al., 2018). **Feasible Learning**: classification on CIFAR-10 (Krizhevsky, 2009) under a loss bound per training example, $\ell_i(\boldsymbol{\theta}) \leq 0.3$, with bounded multipliers (50,000 constraints; Ramirez et al., 2025b). **Sparsity**: classification on CIFAR-10 with a ResNet-18 (He et al., 2016) under a density of $\tau = 0.3$ in all sparsifiable layers (21 equality constraints; Gallego-Posada et al., 2022). **Safe DPO**: helpfulness fine-tuning of Alpaca-7B (Taori et al., 2023) on PKU-SafeRLHF (Dai et al., 2024), evaluated against $\mathbb{E}[\text{cost}] \leq 0$ (one constraint; Wachi et al., 2024). **Safe RL**: policy optimization on the SafetyHalfCheetahVelocity environment of Safety-Gymnasium (Ji et al., 2023b) under a cost limit of 25 (one constraint; Ji et al., 2024).

Baselines. We compare against GA, projected gradient ascent (Arrow et al., 1958); PI control (Sohrabi et al., 2024), ascent with extrapolation as in Eq. (ALM/PI); Adam (Kingma and Ba, 2015); RMSProp (Tieleman and Hinton, 2012); and RMS-ALM (Basir and Senocak, 2023).

Protocol. We tune each baseline's dual step size over a logarithmic grid (Table 3 in Appendix B).[4] AdaMult is not tuned, but run at its default configuration of Eq. (5) on every task. Every configuration runs on five seeds. Appendix C gives one results table per task. The code for all experiments is in the supplementary material.

Feasibility. We consider a run feasible if its end-of-training constraint violation, normalized as follows, is at most 1%; for Safe DPO, we use zero tolerance.[5] **One constraint** (Rate, Safe RL): we divide the violation by the constraint level. **Several constraints** (Multi-group, Sparsity): we divide the largest violation by the constraint level. **Feasible Learning**: we take the percentage of examples whose loss exceeds the bound. **Safe DPO**: we divide mean cost by the base model's mean cost.

[4] PI control crosses 16 step sizes with five proportional gains on every task, giving 80 configurations.

[5] Requiring exact satisfaction would count even tiny constraint violations as failures; we therefore allow a small 1% tolerance. Safe DPO's zero mean-cost bound provides no nonzero reference level for a relative tolerance, so we use zero tolerance. Per-task absolute tolerances are in Appendix B; Tables 10 and 11 in Appendix C report signed, normalized constraint violations for each method before thresholding.

Table 2: **ADAMULT's ingredients ablated on six tasks.** Each row removes one ingredient from ADAMULT and runs it at its best single configuration across all tasks, selected as in Table 1; columns as there. − ① *normalization*: the multiplicative and extrapolated update on raw constraints. − ② *multiplicative*: normalized additive ascent with damping (the additive ADAMULT variant in Table 1). − ③ *extrapolation*: full ADAMULT with $p = 0$. All three keep the readout division by the RMS scale, ④, which for − ① is the identity. The per-task results behind every entry are in Table 12 of Appendix C.

		Across six tasks		Worst task		
	Method	Feasible	Δ score	Task	Feasible	Δ score
	ADAMULT default	100%	+0.00%	None	5/5	—
Tuned Across Tasks	− ① normalization	73%	−2.33%	Sparsity	0/5	+0.08%
	− ② multiplicative	100%	−3.14%	Safe DPO	5/5	−10.36%
	− ③ extrapolation	83%	−4.46%	Sparsity	0/5	−7.92%

4.1 ADAMULT MAKES DUAL TUNING UNNECESSARY

Table 1 tests ADAMULT on two counts. Does its default match baselines tuned separately for *each* task? If so, ADAMULT makes per-task tuning unnecessary. And can any other method work across tasks at *one* configuration? If none can, ADAMULT alone meets the transferability requirement.

On these six tasks, ADAMULT makes per-task dual tuning unnecessary: its default is feasible on every seed and performs comparably, on average, to RMSPROP and RMS-ALM tuned separately for each task. Tuned PI control also achieves feasibility on every seed, but performs worse on average. Even with per-task tuning, GA and ADAM fail to reach feasibility on Sparsity.

Tuned to one default configuration for all tasks, no baseline matches ADAMULT: each is either infeasible sometimes or achieves full feasibility only with lower performance. GA, PI control and ADAM fail to produce feasible solutions on every Sparsity seed, and ADAM on other tasks too. RMS-ALM performs worse than default ADAMULT on average and falls short of full feasibility. RMSPROP is feasible on every seed, but performs worse than default ADAMULT on average.

RMSPROP's success on these tasks shows the value of normalization. However, RMSPROP can remain locally unstable and fail to converge even with tuned step sizes (Proposition 3). Its primal dynamics also remain sensitive to constraint rescaling (§3.2). ADAMULT addresses both limitations.

Across these six tasks, ADAMULT combines feasibility on every seed with performance comparable to baselines tuned separately for each task—all at one fixed default.

4.2 ADAMULT'S INGREDIENTS ALL EARN THEIR PLACE

Table 2 removes one ingredient of ADAMULT at a time and asks whether the resulting algorithm still works across tasks at *one* configuration. If it does not, the ingredient is needed for transferability.

Normalization supports transferability, while extrapolation supplies the damping needed for stability. Removing either leads to infeasible runs even after tuning each variant to a single configuration shared across tasks. Additive ADAMULT remains feasible throughout; multiplicative updates provide a modest performance gain across our tasks.

5 CONCLUSION

We propose ADAMULT, an adaptive dual optimizer designed to work across constrained learning tasks at one default configuration. We prove that ADAMULT has a locally stable fixed point near every regular constrained minimizer and that its primal trajectory is invariant to constraint rescaling. At its default, ADAMULT is feasible on every seed of six tasks and matches the performance of dual optimizers tuned separately for each task. To our knowledge, ADAMULT is the first dual optimizer for constrained deep learning whose default transfers across tasks and constraint scales. ADAMULT does for the multipliers what Adam did for the model: it works out of the box.

ADAMULT needs essentially no tuning of its step size, but it may still need its damping tuned: the damping a problem requires depends on the curvature of its Lagrangian, so ADAMULT's default can under- or over-damp on some problems. Setting the damping automatically, from the dual side alone or by coordinating the primal and dual optimizers, or by a heuristic, remains important future work.

Reproducibility Statement

Proofs are in Appendix A. Appendix B gives the setup of every task and baseline, including how the default of Eq. (5) was fixed, and complete results are in Appendix C. The provided code includes the optimizer, the scripts that run every experiment, and a README with instructions for reproducing the tables and figures. All datasets and models are public.

Statement on the Use of Large Language Models

Large language models (LLMs) helped develop research ideas, theoretical examples, mathematical claims, and proofs. They generated much of the code, including code for method implementations, data preprocessing, and synthetic-data generation, helped design, run and interpret experiments, searched the literature, and assisted with manuscript writing. The authors reviewed and verified the AI-assisted work and take full responsibility for the paper.

References

A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford, and H. Wallach. A Reductions Approach to Fair Classification. In *ICML*, 2018. (Cit. on p. 8, 30)

K. J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-Linear Programming*. Stanford University Press, 1958. (Cit. on p. 1, 3, 8)

S. Basir and I. Senocak. An adaptive augmented Lagrangian method for training physics and equality constrained artificial neural networks. *arXiv preprint arXiv:2306.04904*, 2023. (Cit. on p. 2, 7, 8, 24, 29)

A. Beck and M. Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 2003. (Cit. on p. 2, 4)

M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 2005. (Cit. on p. 21)

D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, 1982. (Cit. on p. 6, 14, 15, 20)

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, third edition, 2016. (Cit. on p. 1, 3, 14)

I. Boero, I. Hounie, L. F. O. Chamon, and A. Ribeiro. Everywhere Learning: Artificial Intelligence with Pointwise Constraints. *arXiv preprint arXiv:2606.01557*, 2026. (Cit. on p. 31)

A. Cotter, H. Jiang, M. Gupta, S. Wang, T. Narayan, S. You, and K. Sridharan. Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals. *JMLR*, 2019. (Cit. on p. 1, 2, 8, 30, 32)

J. Dai, X. Pan, R. Sun, J. Ji, X. Xu, M. Liu, Y. Wang, and Y. Yang. Safe RLHF: Safe Reinforcement Learning from Human Feedback. In *ICLR*, 2024. (Cit. on p. 1, 8, 33)

A. Defazio and K. Mishchenko. Learning-Rate-Free Learning by D-Adaptation. In *ICML*, 2023. (Cit. on p. 2)

T. Dettmers, M. Lewis, S. Shleifer, and L. Zettlemoyer. 8-bit Optimizers via Block-wise Quantization. In *ICLR*, 2022. (Cit. on p. 33)

S. Elaydi. *An Introduction to Difference Equations*. Springer, third edition, 2005. (Cit. on p. 25)

P. Finsler. Über das Vorkommen definiter und semidefiniter Formen in Scharen quadratischer Formen. *Commentarii Mathematici Helvetici*, 1937. (Cit. on p. 21)

J. Gallego-Posada, J. Ramirez, A. Erraqabi, Y. Bengio, and S. Lacoste-Julien. Controlled Sparsity via Constrained Optimization or: *How I Learned to Stop Tuning Penalties and Love Constraints*. In *NeurIPS*, 2022. (Cit. on p. 1, 8, 32)

J. Gallego-Posada, J. Ramirez, M. Hashemizadeh, and S. Lacoste-Julien. Cooper: A Library for Constrained Optimization in Deep Learning. *arXiv preprint arXiv:2504.01212*, 2025. (Cit. on p. 3, 28)

J. Gauvin. A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming. *Mathematical Programming*, 1977. (Cit. on p. 4)

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. (Cit. on p. 8, 31)

M. R. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 1969. (Cit. on p. 2, 3)

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, second edition, 2012. (Cit. on p. 19, 22)

E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. (Cit. on p. 32)

Q. Hu, S. Basir, and I. Senocak. Conditionally adaptive augmented Lagrangian method for physics-informed learning of forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 2026. (Cit. on p. 2)

M. Ivgi, O. Hinder, and Y. Carmon. DoG is SGD's Best Friend: A Parameter-Free Dynamic Step Size Schedule. In *ICML*, 2023. (Cit. on p. 2)

J. Ji, M. Liu, J. Dai, X. Pan, C. Zhang, C. Bian, B. Chen, R. Sun, Y. Wang, and Y. Yang. BeaverTails: Towards Improved Safety Alignment of LLM via a Human-Preference Dataset. In *Advances in Neural Information Processing Systems*, volume 36, pages 24678–24704, 2023a. doi: 10.52202/075280-1072. (Cit. on p. 32)

J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang. Safety-Gymnasium: A Unified Safe Reinforcement Learning Benchmark. In *NeurIPS Datasets and Benchmarks Track*, 2023b. (Cit. on p. 8, 33)

J. Ji, J. Zhou, B. Zhang, J. Dai, X. Pan, R. Sun, W. Huang, Y. Geng, M. Liu, and Y. Yang. OmniSafe: An Infrastructure for Accelerating Safe Reinforcement Learning Research. *JMLR*, 2024. (Cit. on p. 8, 33)

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015. (Cit. on p. 2, 3, 8, 28)

J. Kivinen and M. K. Warmuth. Exponentiated Gradient versus Gradient Descent for Linear Predictors. *Information and Computation*, 1997. (Cit. on p. 4)

R. Kohavi. Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid. In *KDD*, 1996. (Cit. on p. 8, 30)

B. W. Kort and D. P. Bertsekas. Combined Primal–Dual and Penalty Methods for Convex Programming. *SIAM Journal on Control and Optimization*, 1976. (Cit. on p. 2, 15)

A. Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, 2009. (Cit. on p. 8, 31)

K. Liu, J. Hu, Z. Qu, E. H. Yan, L. Yang, and M. Zhang. Residual-Controlled Multiplier Learning for Stochastic Constrained Decision-Making. *arXiv preprint arXiv:2606.07088*, 2026. (Cit. on p. 2)

I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. (Cit. on p. 33)

C. Louizos, M. Welling, and D. P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. In *ICLR*, 2018. (Cit. on p. 32)

O. L. Mangasarian and S. Fromovitz. The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, 1967. (Cit. on p. 4)

K. Mishchenko and A. Defazio. Prodigy: An Expeditiously Adaptive Parameter-Free Learner. In *ICML*, 2024. (Cit. on p. 2)

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019. (Cit. on p. 5, 28)

M. J. D. Powell. A method for nonlinear constraints in minimization problems. In R. Fletcher, editor, *Optimization*. Academic Press, 1969. (Cit. on p. 2, 3)

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*, 2023. (Cit. on p. 32)

J. Ramirez and S. Lacoste-Julien. Dual Optimistic Ascent (PI Control) is the Augmented Lagrangian Method in Disguise. In *ICLR*, 2026. (Cit. on p. 3, 6)

J. Ramirez, M. Hashemizadeh, and S. Lacoste-Julien. Position: Adopt Constraints Over Fixed Penalties in Deep Learning. *arXiv preprint arXiv:2505.20628*, 2025a. (Cit. on p. 3)

J. Ramirez, I. Hounie, J. Elenter, J. Gallego-Posada, M. Hashemizadeh, A. Ribeiro, and S. Lacoste-Julien. Feasible Learning. In *AISTATS*, 2025b. (Cit. on p. 8, 31)

D. J. Rezende and F. Viola. Taming VAEs. *arXiv preprint arXiv:1810.00597*, 2018. (Cit. on p. 2)

R. T. Rockafellar. A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical Programming*, 1973. (Cit. on p. 2, 3)

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017. (Cit. on p. 33)

M. Sohrabi, J. Ramirez, T. H. Zhang, S. Lacoste-Julien, and J. Gallego-Posada. On PI Controllers for Updating Lagrange Multipliers in Constrained Optimization. In *ICML*, 2024. (Cit. on p. 6, 8, 28)

A. Stooke, J. Achiam, and P. Abbeel. Responsive Safety in Reinforcement Learning by PID Lagrangian Methods. In *ICML*, 2020. (Cit. on p. 1, 2, 3)

R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA Model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023. (Cit. on p. 8, 33)

T. Tieleman and G. Hinton. Lecture 6.5—RMSProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. (Cit. on p. 2, 4, 8, 28)

P. Tseng and D. P. Bertsekas. On the convergence of the exponential multiplier method for convex programming. *Mathematical Programming*, 1993. (Cit. on p. 2)

J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. OpenML: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 2014. (Cit. on p. 30)

A. Wachi, T. Q. Tran, R. Sato, T. Tanabe, and Y. Akimoto. Stepwise Alignment for Constrained Language Model Policy Optimization. In *NeurIPS*, 2024. (Cit. on p. 8, 32)

Appendix

A Proofs and Theoretical Results **14**

A.1 Definitions and Assumptions 14

A.2 Equivariance of the Optimal Multipliers (Lemma 1) 14

A.3 KKT Consistency of AdaMult (Proposition 1) 15

A.4 Local Stability of AdaMult (Proposition 2) 17

A.5 Stability and Scale Sensitivity of RMSProp and RMS-ALM (Propositions 3, 5 and 6) 23

A.6 Scale Invariance of AdaMult (Proposition 4) 27

B Experimental Setup **28**

B.1 Baseline optimizers 28

B.2 Hyperparameter choices 29

B.3 Rate-constrained classification 30

B.4 Multi-group rate constraints 30

B.5 Feasible Learning 31

B.6 Controlled sparsity 32

B.7 Safety-constrained DPO 32

B.8 Safe RL 33

C Complete Results per Task **34**

C.1 Aggregation 34

C.2 Detailed comparisons 35

C.3 Rate-constrained classification 38

C.4 Multi-group rate constraints 39

C.5 Feasible Learning 40

C.6 Controlled sparsity 41

C.7 Safety-constrained DPO 42

C.8 Safe RL 43

C.9 Ingredient sweeps 44

A Proofs and Theoretical Results

A.1 Definitions and Assumptions

These definitions match the standard ones of nonlinear programming (Bertsekas, 2016, Ch. 3) and augmented-Lagrangian theory (Bertsekas, 1982, Ch. 2). We assume $f, \boldsymbol{g}, \boldsymbol{h}$ twice continuously differentiable. At a KKT point $\boldsymbol{\theta}^\star$, $\mathcal{A} = \{i : g_i(\boldsymbol{\theta}^\star) = 0\}$ is the set of inequalities active at $\boldsymbol{\theta}^\star$, and a subscript $\mathcal{A}$ restricts a vector to those entries, or a matrix to those rows, as in $\boldsymbol{g}_\mathcal{A}$ and $\nabla_{\boldsymbol{\theta}} \boldsymbol{g}_\mathcal{A}$. The norm $\|\cdot\|$ is the Euclidean norm on vectors, and $\|\cdot\|_{\mathrm{op}}$ the operator norm on matrices.

Definition 1 (KKT point)**.** $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ is a *KKT point* of Eq. (1) if it satisfies:

- *primal feasibility*: $\boldsymbol{g}(\boldsymbol{\theta}^\star) \le \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{\theta}^\star) = \mathbf{0}$;
- *dual feasibility*: $\boldsymbol{\lambda}^\star \ge \mathbf{0}$;
- *stationarity*: $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^\star) + \sum_i \lambda_i^\star \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}^\star) + \sum_j \mu_j^\star \nabla_{\boldsymbol{\theta}} h_j(\boldsymbol{\theta}^\star) = \mathbf{0}$;
- *complementary slackness*: $\lambda_i^\star g_i(\boldsymbol{\theta}^\star) = 0$ for all i.

Note that KKT points correspond exactly to the fixed points of projected gradient descent–ascent on the Lagrangian $\mathcal{L}$.

Definition 2 (Locally stable fixed point)**.** Let $\boldsymbol{s}_{t+1} = F(\boldsymbol{s}_t)$ be the update map of an algorithm on its state $\boldsymbol{s}$, and suppose F is continuously differentiable near a fixed point $\boldsymbol{s} = F(\boldsymbol{s})$. This fixed point is a *locally stable fixed point* (LSFP) if the Jacobian of F at $\boldsymbol{s}$ has spectral radius strictly less than one.

For the local-stability analysis, the state $\boldsymbol{s}$ is $(\boldsymbol{\theta}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ for algorithms such as gradient descent–ascent, and $(\boldsymbol{\theta}, \boldsymbol{z}, \boldsymbol{w})$ for AdaMult, where $\boldsymbol{z}, \boldsymbol{w}$ are the dual log-states and the RMS scales are held fixed (§3.1). From any starting point close enough to an LSFP, the iterates converge to it at a linear rate (Bertsekas, 2016, Prop. 5.4.1).

Assumption 1 (Strict complementary slackness)**.** A KKT point $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ satisfies *strict complementary slackness* if

$$\lambda_i^\star > 0 \ \text{ and } \ g_i(\boldsymbol{\theta}^\star) = 0 \quad \text{or} \quad \lambda_i^\star = 0 \ \text{ and } \ g_i(\boldsymbol{\theta}^\star) < 0, \qquad \forall i = 1, \dots, m.$$

Assumption 2 (Second-order sufficiency condition)**.** A KKT point $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ satisfies the *second-order sufficiency condition* (SOSC) if the Hessian of the Lagrangian is positive definite on the tangent space of the active constraints:

$$\boldsymbol{d}^\top \nabla_{\boldsymbol{\theta}}^2 \mathcal{L}(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)\, \boldsymbol{d} > 0$$
$$\text{for all } \boldsymbol{d} \in \mathbb{R}^d \setminus \{\mathbf{0}\} \text{ with } \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_\mathcal{A}(\boldsymbol{\theta}^\star)\, \boldsymbol{d} = \mathbf{0} \text{ and } \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}^\star)\, \boldsymbol{d} = \mathbf{0},$$

i.e., along the directions that, to first order, neither leave the feasible set nor change the active set.

Assumption 3 (Linear independence constraint qualification)**.** A point $\boldsymbol{\theta}^\star \in \mathbb{R}^d$ satisfies the *linear independence constraint qualification* (LICQ) if the matrix $\left[\nabla_{\boldsymbol{\theta}} \boldsymbol{g}_\mathcal{A}(\boldsymbol{\theta}^\star)^\top, \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}^\star)^\top\right]^\top$ has full row rank, i.e., the gradients of the active constraints are linearly independent.

A.2 Equivariance of the Optimal Multipliers (Lemma 1)

Proof of Lemma 1. Let $(\boldsymbol{\lambda}', \boldsymbol{\mu}')$ be candidate multipliers at $\boldsymbol{\theta}^\star$ for the rescaled problem. Since $\varrho, \alpha_i, \gamma_j > 0$, each KKT condition of the rescaled problem is the corresponding condition of Eq. (1) with multipliers $(\alpha_i \lambda_i'/\varrho, \gamma_j \mu_j'/\varrho)$: $\boldsymbol{\theta}^\star$ is feasible for one problem iff for the other; dual feasibility $\lambda_i' \ge 0$ holds iff $\alpha_i \lambda_i'/\varrho \ge 0$; complementary slackness $\lambda_i' \cdot \alpha_i g_i(\boldsymbol{\theta}^\star) = 0$ holds iff $(\alpha_i \lambda_i'/\varrho)\, g_i(\boldsymbol{\theta}^\star) = 0$; and stationarity of the rescaled problem,

$$\varrho\, \nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}^\star) + \sum_i \alpha_i \lambda_i'\, \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}^\star) + \sum_j \gamma_j \mu_j'\, \nabla_{\boldsymbol{\theta}} h_j(\boldsymbol{\theta}^\star) = \mathbf{0}, \tag{7}$$

divided by ϱ is stationarity of Eq. (1) with multipliers $(\alpha_i \lambda_i'/\varrho, \gamma_j \mu_j'/\varrho)$. Hence $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}', \boldsymbol{\mu}')$ is a KKT point of the rescaled problem iff $(\boldsymbol{\theta}^\star, \{\alpha_i \lambda_i'/\varrho\}, \{\gamma_j \mu_j'/\varrho\})$ is a KKT point of Eq. (1).

Inactive multipliers vanish by complementary slackness; LICQ makes the remaining multipliers at $\boldsymbol{\theta}^\star$ unique through stationarity. So $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}', \boldsymbol{\mu}')$ is a KKT point of the rescaled problem iff $\alpha_i \lambda_i'/\varrho = \lambda_i^\star$ and $\gamma_j \mu_j'/\varrho = \mu_j^\star$, i.e. iff $(\boldsymbol{\lambda}', \boldsymbol{\mu}') = (\{\varrho\lambda_i^\star/\alpha_i\}, \{\varrho\mu_j^\star/\gamma_j\})$. □

A.3 KKT Consistency of AdaMult (Proposition 1)

We prove that the fixed points of Algorithm 1 correspond to the KKT points of Eq. (1), up to the $z_{\min}$ floor on the multipliers of inactive inequalities. We proceed in two stages. Lemma 2 characterizes the fixed points of a step projected onto $[z_{\min}, \infty)$. The proof of Proposition 1 then writes out the fixed-point equations of the whole AdaMult state and, applying the lemma to each inequality's log-state, shows that they are equivalent to the KKT conditions adjusted for the multiplier floor.

Lemma 2 (Fixed points of a projected step)**.** *Let $z_{\min} \in \mathbb{R}$, $z \geq z_{\min}$, $r > 0$, $p \geq 0$ and $s \in \mathbb{R}$. Then*

$$z = \max(z + r\,s,\ z_{\min}) \iff s = 0 \ \text{ or } \ \big(z = z_{\min},\ s \leq 0\big), \tag{8}$$

and in that case $\max(z + p\,s,\ z_{\min}) = z$.

Proof. "⇒." Suppose $z = \max(z + rs, z_{\min})$. If $z > z_{\min}$, the maximum is not $z_{\min}$, hence equals its first argument: $z + rs = z$, and $s = 0$ since $r > 0$. If $z = z_{\min}$, the equation reads $\max(z_{\min} + rs, z_{\min}) = z_{\min}$, i.e. $z_{\min} + rs \leq z_{\min}$, hence $s \leq 0$.

"⇐." If $s = 0$, then $\max(z + rs, z_{\min}) = \max(z, z_{\min}) = z$ since $z \geq z_{\min}$. If $z = z_{\min}$ and $s \leq 0$, then $z + rs \leq z_{\min}$ since $r > 0$, so $\max(z + rs, z_{\min}) = z_{\min} = z$.

For the last claim: if $s = 0$, then $\max(z + ps, z_{\min}) = \max(z, z_{\min}) = z$; if $z = z_{\min}$ and $s \leq 0$, then $ps \leq 0$ since $p \geq 0$, so $\max(z + ps, z_{\min}) = z_{\min} = z$. □

Proof of Proposition 1. The argument adapts the fixed-point analysis of the exponential method of multipliers (Kort and Bertsekas, 1976; Bertsekas, 1982, Ch. 5) to Algorithm 1.

Let $\boldsymbol{s} = (\boldsymbol{\theta}; z_i, v_i, \widehat{v}_i^{\max}; w_j, u_j, \widehat{u}_j^{\max})$ collect the primal iterate and the state of Algorithm AdaMult, and let F be one iteration of Algorithm 1 on $\boldsymbol{s}$, with a plain gradient primal step of size $\eta_\theta > 0$, $z_{\max} = \infty$, and no bias correction, $\widehat{v}_i = v_i$ and $\widehat{u}_j = u_j$.

Fix a candidate triple $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ and peak values $\widehat{v}_i^{\max} \geq g_i(\boldsymbol{\theta}^\circ)^2$, $\widehat{u}_j^{\max} \geq h_j(\boldsymbol{\theta}^\circ)^2$, and let

$$a_i = \max\Big(|g_i(\boldsymbol{\theta}^\circ)|,\ \delta\sqrt{\widehat{v}_i^{\max}}\Big) + \epsilon, \qquad b_j = \max\Big(|h_j(\boldsymbol{\theta}^\circ)|,\ \delta\sqrt{\widehat{u}_j^{\max}}\Big) + \epsilon \tag{9}$$

be the scales of Algorithm AdaMult at second-moment estimates $g_i(\boldsymbol{\theta}^\circ)^2$, $h_j(\boldsymbol{\theta}^\circ)^2$. Both scales are positive because $\epsilon > 0$. Each also bounds the corresponding constraint magnitude: $a_i \geq |g_i(\boldsymbol{\theta}^\circ)|$ and $b_j \geq |h_j(\boldsymbol{\theta}^\circ)|$, since these magnitudes appear in their respective maxima.[6] Write the clipped, normalized constraint values as $\bar{g}_i = \texttt{clip}(g_i(\boldsymbol{\theta}^\circ)/a_i, -c, c)$ and $\bar{h}_j = \texttt{clip}(h_j(\boldsymbol{\theta}^\circ)/b_j, -c, c)$; since $a_i, b_j, c > 0$, these share the signs of $g_i(\boldsymbol{\theta}^\circ)$ and $h_j(\boldsymbol{\theta}^\circ)$.

We show that $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ corresponds to a fixed point with these peaks if and only if

$$\begin{aligned}
&\textit{primal feasibility}: && \boldsymbol{g}(\boldsymbol{\theta}^\circ) \leq \mathbf{0} \ \text{ and } \ \boldsymbol{h}(\boldsymbol{\theta}^\circ) = \mathbf{0};\\
&\textit{dual feasibility}: && \lambda_i^\circ \geq e^{z_{\min}}/a_i \ \text{ for all } i;\\
&\textit{stationarity}: && \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \mathbf{0};\\
&\textit{complementary slackness}: && \big(\lambda_i^\circ - e^{z_{\min}}/a_i\big)\, g_i(\boldsymbol{\theta}^\circ) = 0 \ \text{ for all } i.
\end{aligned} \tag{10}$$

This is the KKT system of Definition 1 with the floor $e^{z_{\min}}/a_i$ in place of 0 in dual feasibility and complementary slackness.

Fixed point ⇒ Eq. (10). Suppose $F(\boldsymbol{s}^\circ) = \boldsymbol{s}^\circ$, with the peak values used in Eq. (9). We prove that the corresponding triple $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ satisfies Eq. (10).

The second-moment equation, $\beta v_i^\circ + (1 - \beta)\, g_i(\boldsymbol{\theta}^\circ)^2 = v_i^\circ$, forces $v_i^\circ = g_i(\boldsymbol{\theta}^\circ)^2$ since $\beta < 1$; likewise $u_j^\circ = h_j(\boldsymbol{\theta}^\circ)^2$. The peak equation $\widehat{v}_i^{\max} = \max(\widehat{v}_i^{\max}, v_i^\circ)$ then gives $\widehat{v}_i^{\max} \geq g_i(\boldsymbol{\theta}^\circ)^2$;

[6] If $\widehat{v}_i^{\max} = 0$, Algorithm AdaMult sets $a_i = 1$ in place of Eq. (9). Then $a_i > 0$, and $a_i \geq |g_i(\boldsymbol{\theta}^\circ)| = 0$ because $g_i(\boldsymbol{\theta}^\circ)^2 \leq \widehat{v}_i^{\max} = 0$. The same holds for b_j.

likewise $\widehat{u}_j^{\max} \geq h_j(\boldsymbol{\theta}^\circ)^2$. Thus every fixed point satisfies the peak bounds used in Eq. (9), and its constraint scales are those given there.

For each inequality, the z_i° equation reads $z_i^\circ = \max(z_i^\circ + r\bar{g}_i,\ z_{\min})$, so $z_i^\circ \geq z_{\min}$, and Lemma 2 with $s = \bar{g}_i$ gives two cases:

$$\text{(a)} \quad \bar{g}_i = 0, \qquad \text{or} \qquad \text{(b)} \quad z_i^\circ = z_{\min} \ \text{ and } \ \bar{g}_i \leq 0.$$

In both cases, the extrapolated log-state is $\max(z_i^\circ + p\bar{g}_i,\ z_{\min}) = z_i^\circ$, so the corresponding multiplier is $\lambda_i^\circ = e^{z_i^\circ}/a_i$.

For each equality, the w_j° equation reads $w_j^\circ = w_j^\circ + r\bar{h}_j$, so $\bar{h}_j = 0$ as $r > 0$; the extrapolated log-state is then $w_j^\circ + p\bar{h}_j = w_j^\circ$, so the corresponding multiplier is $\mu_j^\circ = \sinh(w_j^\circ)/b_j$.

These fixed-point equations imply the four conditions in Eq. (10):

Primal feasibility. As $\bar{h}_j$ and $\bar{g}_i$ share the signs of $h_j(\boldsymbol{\theta}^\circ)$ and $g_i(\boldsymbol{\theta}^\circ)$, $\bar{h}_j = 0$ gives $h_j(\boldsymbol{\theta}^\circ) = 0$, and $\bar{g}_i \leq 0$, which holds in both inequality cases above, gives $g_i(\boldsymbol{\theta}^\circ) \leq 0$.

Dual feasibility. $\lambda_i^\circ = e^{z_i^\circ}/a_i \geq e^{z_{\min}}/a_i$, since $z_i^\circ \geq z_{\min}$.

Stationarity. The $\boldsymbol{\theta}^\circ$ equation reads

$$\boldsymbol{\theta}^\circ = \boldsymbol{\theta}^\circ - \eta_\theta \, \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ), \tag{11}$$

so $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \mathbf{0}$.

Complementary slackness. In inequality case (a), $g_i(\boldsymbol{\theta}^\circ) = 0$; in case (b), $\lambda_i^\circ = e^{z_{\min}}/a_i$. Either way $(\lambda_i^\circ - e^{z_{\min}}/a_i)\, g_i(\boldsymbol{\theta}^\circ) = 0$. Hence Eq. (10) holds.

Equation (10) $\Rightarrow$ fixed point. Suppose $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ satisfies Eq. (10), with the scales and peak values of Eq. (9). We construct a corresponding state $\boldsymbol{s}^\circ$ and prove $F(\boldsymbol{s}^\circ) = \boldsymbol{s}^\circ$. Set $\boldsymbol{s}^\circ = (\boldsymbol{\theta}^\circ;\ z_i^\circ, v_i^\circ, \widehat{v}_i^{\max};\ w_j^\circ, u_j^\circ, \widehat{u}_j^{\max})$, where

$$\begin{aligned} z_i^\circ &= \log(a_i \lambda_i^\circ), \qquad & w_j^\circ &= \operatorname{asinh}(b_j \mu_j^\circ), \\ v_i^\circ &= g_i(\boldsymbol{\theta}^\circ)^2, \qquad & u_j^\circ &= h_j(\boldsymbol{\theta}^\circ)^2. \end{aligned} \tag{12}$$

Dual feasibility in Eq. (10) gives $a_i \lambda_i^\circ \geq e^{z_{\min}} > 0$, so the logarithm exists and $z_i^\circ \geq z_{\min}$. Each component remains fixed:

Second moments and peaks. Each second-moment estimate equals the squared constraint value, so its update leaves it unchanged. The peaks also stay fixed because $\widehat{v}_i^{\max} \geq v_i^\circ$ and $\widehat{u}_j^{\max} \geq u_j^\circ$. The constraint scales therefore remain those of Eq. (9).

Inequality log-states z_i°. For each i, dual feasibility in Eq. (10) leaves two cases. Either $\lambda_i^\circ > e^{z_{\min}}/a_i$, so $g_i(\boldsymbol{\theta}^\circ) = 0$ by complementary slackness in Eq. (10) and thus $\bar{g}_i = 0$; or $\lambda_i^\circ = e^{z_{\min}}/a_i$, so $z_i^\circ = z_{\min}$, while $g_i(\boldsymbol{\theta}^\circ) \leq 0$ by primal feasibility in Eq. (10) gives $\bar{g}_i \leq 0$. In both cases the right-hand side of Eq. (8) holds with $s = \bar{g}_i$, so by Lemma 2 the update gives $\max(z_i^\circ + r\bar{g}_i,\ z_{\min}) = z_i^\circ$ and the extrapolated log-state is $\max(z_i^\circ + p\bar{g}_i,\ z_{\min}) = z_i^\circ$. The corresponding multiplier is therefore $e^{z_i^\circ}/a_i = \lambda_i^\circ$.

Equality log-states w_j°. By primal feasibility in Eq. (10), $h_j(\boldsymbol{\theta}^\circ) = 0$, so $\bar{h}_j = 0$; the update is $w_j^\circ + r\bar{h}_j = w_j^\circ$, and the corresponding multiplier is $\sinh(w_j^\circ + p\bar{h}_j)/b_j = \sinh(w_j^\circ)/b_j = \mu_j^\circ$.

Primal iterate $\boldsymbol{\theta}^\circ$. The corresponding multipliers are therefore $(\boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$, and by stationarity in Eq. (10), $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \mathbf{0}$, so $\boldsymbol{\theta}^\circ - \eta_\theta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \boldsymbol{\theta}^\circ$. Hence $F(\boldsymbol{s}^\circ) = \boldsymbol{s}^\circ$.

Vanishing multiplier floor. At any fixed point, Eq. (10) gives:

- *Primal feasibility, dual feasibility and stationarity* hold exactly, since $\boldsymbol{g}(\boldsymbol{\theta}^\circ) \leq \mathbf{0}$, $\boldsymbol{h}(\boldsymbol{\theta}^\circ) = \mathbf{0}$, $\boldsymbol{\lambda}^\circ > \mathbf{0}$ and $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \mathbf{0}$.
- *Complementary slackness* holds up to an error bounded by the floor:

$$|\lambda_i^\circ \, g_i(\boldsymbol{\theta}^\circ)| = \frac{e^{z_{\min}}}{a_i} \, |g_i(\boldsymbol{\theta}^\circ)| \leq e^{z_{\min}},$$

since $a_i \geq |g_i(\boldsymbol{\theta}^\circ)|$.

Thus, along any sequence of fixed points with $z_{\min} \to -\infty$, the complementarity error tends to zero. Every finite limit point of their triples $(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ is therefore a KKT point of Eq. (1), by continuity of $\boldsymbol{g}$, $\boldsymbol{h}$ and $\nabla_{\boldsymbol{\theta}}\mathcal{L}$. □

Remark 1 (Other primal optimizers)**.** The proof of Proposition 1 uses the primal update only through the equivalence between its fixed-point equations and $\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbf{0}$. It therefore extends to any primal optimizer with this property, including momentum gradient descent and ADAM without weight decay or bias correction. For these optimizers, the additional moment states vanish at a fixed point.

A.4 LOCAL STABILITY OF ADAMULT (PROPOSITION 2)

Let $\boldsymbol{\theta}^\star$ be a local minimizer of Eq. (1) satisfying LICQ (Assumption 3). Its KKT multipliers $(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ therefore exist and are unique. Assume that this KKT triple also satisfies strict complementarity and SOSC (Assumptions 1 and 2), and let $\mathcal{A}$ be its active set.

Proof idea. First consider a modified version of ADAMULT (Eqs. (13) to (15) below) in which the multipliers of inequalities inactive at $\boldsymbol{\theta}^\star$ are held at zero instead of updated. By Lemma 3, the KKT triple $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ corresponds to a fixed point of this modified algorithm.

At this fixed point, SOSC (Assumption 2) gives the Lagrangian positive curvature along directions tangent to the active inequalities and equalities. Dual extrapolation adds curvature in the normal directions, so sufficiently large damping makes the effective primal curvature positive in every direction. With LICQ and sufficiently small primal and dual steps, nearby iterates of modified ADAMULT converge to this fixed point.

In ADAMULT, the multipliers of those inactive inequalities sit at a positive floor at a fixed point. Taking $z_{\min}$ sufficiently negative makes their contribution to the primal update arbitrarily small. The implicit function theorem then gives a fixed point of ADAMULT near that of the modified iteration. The update Jacobian of ADAMULT at this nearby point remains stable by continuity.

Modified ADAMULT. As in §3.1, assume the RMS scales have settled to constants $a_i, b_j > 0$ and take $z_{\max} = \infty$. The normalized active constraints and equalities vanish at $\boldsymbol{\theta}^\star$, so their $[-c, c]$ clip does not bind nearby; neither does the active log-state floor for sufficiently negative $z_{\min}$.

Keep $\lambda_i = 0$ for $i \notin \mathcal{A}$ and update only the active inequality log-states $\boldsymbol{z}_t = (z_{i,t})_{i\in\mathcal{A}}$ and the equality log-states $\boldsymbol{w}_t$. Write $\bar{g}_i(\boldsymbol{\theta}) = g_i(\boldsymbol{\theta})/a_i$ and $\bar{h}_j(\boldsymbol{\theta}) = h_j(\boldsymbol{\theta})/b_j$. With dual step size $r > 0$ and damping $p \geq 0$, modified ADAMULT has the local updates

$$z_{i,t} = z_{i,t-1} + r\,\bar{g}_i(\boldsymbol{\theta}_t), \qquad \lambda_{i,t+1} = \exp\big(z_{i,t} + p\,\bar{g}_i(\boldsymbol{\theta}_t)\big)/a_i, \qquad i \in \mathcal{A}, \tag{13}$$

$$w_{j,t} = w_{j,t-1} + r\,\bar{h}_j(\boldsymbol{\theta}_t), \qquad \mu_{j,t+1} = \sinh\big(w_{j,t} + p\,\bar{h}_j(\boldsymbol{\theta}_t)\big)/b_j, \tag{14}$$

followed by a plain gradient primal step of size $\eta_\theta > 0$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_\theta\,\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1}), \tag{15}$$

where $\boldsymbol{\lambda}_{t+1}$ has the entries $\lambda_{i,t+1}$ of Eq. (13) for $i \in \mathcal{A}$ and 0 elsewhere. The iteration maps $(\boldsymbol{\theta}_t; \boldsymbol{z}_{t-1}; \boldsymbol{w}_{t-1})$ to $(\boldsymbol{\theta}_{t+1}; \boldsymbol{z}_t; \boldsymbol{w}_t)$.

Lemma 3 (Fixed point at a KKT point)**.** *Let $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ be a KKT point of Eq. (1) satisfying Assumption 1, with active set $\mathcal{A}$, so that $\lambda_i^\star > 0$ for $i \in \mathcal{A}$ and $\lambda_i^\star = 0$ otherwise. Then, with*

$$z_i^\star = \log(a_i\lambda_i^\star)\ (i \in \mathcal{A}), \qquad w_j^\star = \operatorname{asinh}(b_j\mu_j^\star), \tag{16}$$

the state $(\boldsymbol{\theta}^\star; \boldsymbol{z}_{\mathcal{A}}^\star; \boldsymbol{w}^\star)$ is a fixed point of the modified ADAMULT iteration (Eqs. (13) to (15)), and the multipliers it feeds to the primal step there are $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}^\star$ and $\boldsymbol{\mu}_{t+1} = \boldsymbol{\mu}^\star$.

Proof. By primal feasibility in Definition 1, $\boldsymbol{h}(\boldsymbol{\theta}^\star) = \mathbf{0}$, and $g_i(\boldsymbol{\theta}^\star) = 0$ for $i \in \mathcal{A}$ by the definition of the active set, so $\bar{g}_i(\boldsymbol{\theta}^\star) = 0$ for $i \in \mathcal{A}$ and $\bar{h}_j(\boldsymbol{\theta}^\star) = 0$ for all j. At $(\boldsymbol{\theta}^\star; \boldsymbol{z}_{\mathcal{A}}^\star; \boldsymbol{w}^\star)$ the dual step Eqs. (13) and (14) therefore gives $z_{i,t} = z_i^\star$ and $w_{j,t} = w_j^\star$, and, as the extrapolation terms vanish, $\lambda_{i,t+1} = e^{z_i^\star}/a_i = \lambda_i^\star$ for $i \in \mathcal{A}$ and $\mu_{j,t+1} = \sinh(w_j^\star)/b_j = \mu_j^\star$ by Eq. (16). With the entries outside $\mathcal{A}$ held at $0 = \lambda_i^\star$, $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}^\star$. The primal step Eq. (15) then gives $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}^\star$, as $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \mathbf{0}$ by stationarity in Definition 1. □

Lemma 4 (Linearization at a fixed point)**.** *Let $(\boldsymbol{\theta}^\circ; \boldsymbol{z}^\circ_{\mathcal{A}}; \boldsymbol{w}^\circ)$ be a fixed point of the modified* AdaMult *iteration (Eqs. (13) to (15)) at which $f, \boldsymbol{g}, \boldsymbol{h}$ are twice continuously differentiable, and write $\lambda_i^\circ = e^{z_i^\circ}/a_i$ for $i \in \mathcal{A}$, $\lambda_i^\circ = 0$ otherwise, and $\mu_j^\circ = \sinh(w_j^\circ)/b_j$. Then the Jacobian of the iteration at $(\boldsymbol{\theta}^\circ; \boldsymbol{z}^\circ_{\mathcal{A}}; \boldsymbol{w}^\circ)$ is*

$$\mathcal{J}_{\eta_\theta} = \begin{pmatrix} \mathbf{I} - \eta_\theta \big(\boldsymbol{H} + (r+p)\,\mathbf{M}\big) & -\eta_\theta\, \mathbf{N}_g^\top \boldsymbol{\Phi}_g & -\eta_\theta\, \mathbf{N}_h^\top \boldsymbol{\Phi}_h \\ r\,\mathbf{N}_g & \mathbf{I} & \mathbf{0} \\ r\,\mathbf{N}_h & \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{17}$$

where $\boldsymbol{H} = \nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ)$ is the Hessian of the Lagrangian. The matrices $\mathbf{N}_g = \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}^\circ)$ and $\mathbf{N}_h = \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{h}}(\boldsymbol{\theta}^\circ)$ are the Jacobians of the normalized constraints, with rows $a_i^{-1} \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}^\circ)$ and $b_j^{-1} \nabla_{\boldsymbol{\theta}} h_j(\boldsymbol{\theta}^\circ)$. The remaining matrices are

$$\boldsymbol{\Phi}_g = \operatorname{diag}(a_i \lambda_i^\circ)_{i \in \mathcal{A}}, \qquad \boldsymbol{\Phi}_h = \operatorname{diag}(\cosh w_j^\circ), \qquad \mathbf{M} = \mathbf{N}_g^\top \boldsymbol{\Phi}_g \mathbf{N}_g + \mathbf{N}_h^\top \boldsymbol{\Phi}_h \mathbf{N}_h. \tag{18}$$

Since $a_i \lambda_i^\circ = e^{z_i^\circ}$, the diagonal entries of $\boldsymbol{\Phi}_g$ and $\boldsymbol{\Phi}_h$ are the derivatives of exp *and* sinh *at the corresponding log-states.*

Proof. The map is the dual step $(\boldsymbol{\theta}_t; \boldsymbol{z}_{t-1}; \boldsymbol{w}_{t-1}) \mapsto (\boldsymbol{\theta}_t; \boldsymbol{z}_t; \boldsymbol{w}_t)$ of Eqs. (13) and (14) followed by the primal step $(\boldsymbol{\theta}_t; \boldsymbol{z}_t; \boldsymbol{w}_t) \mapsto (\boldsymbol{\theta}_{t+1}; \boldsymbol{z}_t; \boldsymbol{w}_t)$ of Eq. (15).

Dual step. The dual step adds $r\, \bar{g}_i(\boldsymbol{\theta}_t)$ to $z_{i,t-1}$ and $r\, \bar{h}_j(\boldsymbol{\theta}_t)$ to $w_{j,t-1}$, increments that depend on $\boldsymbol{\theta}_t$ alone, and acts as the identity on $\boldsymbol{\theta}_t$, so

$$\mathcal{J}^{\text{dual}} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ r\, \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}_t) & \mathbf{I} & \mathbf{0} \\ r\, \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{h}}(\boldsymbol{\theta}_t) & \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{19}$$

Primal step. The primal step moves $\boldsymbol{\theta}_t$ by $-\eta_\theta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1})$, whose multipliers are those of Eqs. (13) and (14). With $\nabla_{\boldsymbol{\theta}} \mathcal{L} = \nabla_{\boldsymbol{\theta}} f + \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}^\top \boldsymbol{\lambda}_{\mathcal{A}} + \nabla_{\boldsymbol{\theta}} \boldsymbol{h}^\top \boldsymbol{\mu}$ and the multipliers outside $\mathcal{A}$ held at 0 contributing nothing, the chain rule gives the Jacobian of the primal step, with $\boldsymbol{z}_t$ and $\boldsymbol{w}_t$ held fixed,

$$\begin{aligned} \frac{\partial \boldsymbol{\theta}_{t+1}}{\partial \boldsymbol{\theta}_t} &= \mathbf{I} - \eta_\theta \left(\nabla^2_{\boldsymbol{\theta}} \mathcal{L} + \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}_t)^\top \frac{\partial \boldsymbol{\lambda}_{\mathcal{A}}}{\partial \boldsymbol{\theta}} + \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}_t)^\top \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}} \right), \\ \frac{\partial \boldsymbol{\theta}_{t+1}}{\partial \boldsymbol{z}_t} &= -\eta_\theta\, \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}_t)^\top \frac{\partial \boldsymbol{\lambda}_{\mathcal{A}}}{\partial \boldsymbol{z}}, \\ \frac{\partial \boldsymbol{\theta}_{t+1}}{\partial \boldsymbol{w}_t} &= -\eta_\theta\, \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}_t)^\top \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{w}}, \end{aligned} \tag{20}$$

with $\nabla^2_{\boldsymbol{\theta}} \mathcal{L}$ at $(\boldsymbol{\theta}_t, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1})$.

With $\boldsymbol{z}_t$ and $\boldsymbol{w}_t$ held fixed, the multipliers depend on $\boldsymbol{\theta}_t$ only through the extrapolation terms $p\, \bar{g}_i(\boldsymbol{\theta}_t)$ and $p\, \bar{h}_j(\boldsymbol{\theta}_t)$, so

$$\frac{\partial \boldsymbol{\lambda}_{\mathcal{A}}}{\partial \boldsymbol{\theta}} = p\, \mathbf{D}_g\, \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}_t), \qquad \frac{\partial \boldsymbol{\mu}}{\partial \boldsymbol{\theta}} = p\, \mathbf{D}_h\, \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{h}}(\boldsymbol{\theta}_t), \tag{21}$$

where $\mathbf{D}_g = \partial \boldsymbol{\lambda}_{\mathcal{A}} / \partial \boldsymbol{z}$ and $\mathbf{D}_h = \partial \boldsymbol{\mu} / \partial \boldsymbol{w}$ are the Jacobians of the multipliers with respect to the log-states. Each $\lambda_{i,t+1}$ depends on the log-states only through $z_{i,t}$, and each $\mu_{j,t+1}$ only through $w_{j,t}$, so both are diagonal:

$$\mathbf{D}_g = \operatorname{diag}\big(\exp(z_{i,t} + p\, \bar{g}_i(\boldsymbol{\theta}_t))/a_i\big), \qquad \mathbf{D}_h = \operatorname{diag}\big(\cosh(w_{j,t} + p\, \bar{h}_j(\boldsymbol{\theta}_t))/b_j\big). \tag{22}$$

Substituting Eqs. (21) and (22) into Eq. (20) gives the primal-step Jacobian, whose lower blocks pass $\boldsymbol{z}_t$ and $\boldsymbol{w}_t$ through unchanged:

$$\mathcal{J}^{\text{primal}} = \begin{pmatrix} \mathbf{I} - \eta_\theta \big(\nabla^2_{\boldsymbol{\theta}} \mathcal{L} + p\, \mathbf{M}_t\big) & -\eta_\theta\, \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}_t)^\top \mathbf{D}_g & -\eta_\theta\, \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}_t)^\top \mathbf{D}_h \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{pmatrix}, \tag{23}$$

where

$$\mathbf{M}_t = \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}_t)^\top \mathbf{D}_g \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}_t) + \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}_t)^\top \mathbf{D}_h \nabla_{\boldsymbol{\theta}} \bar{\boldsymbol{h}}(\boldsymbol{\theta}_t). \tag{24}$$

Composition. The Jacobian of the iteration is $\mathcal{J}_t = \mathcal{J}^{\text{primal}}\mathcal{J}^{\text{dual}}$. Multiplying, the r contribution of $\mathcal{J}^{\text{dual}}$ joins the p contribution in the leading block, by Eq. (24):

$$\mathcal{J}_t = \begin{pmatrix} \mathbf{I} - \eta_\theta\big(\nabla^2_{\boldsymbol{\theta}}\mathcal{L} + (r+p)\,\mathbf{M}_t\big) & -\eta_\theta\,\nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}_t)^\top\mathbf{D}_g & -\eta_\theta\,\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{\theta}_t)^\top\mathbf{D}_h \\ r\,\nabla_{\boldsymbol{\theta}}\bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}_t) & \mathbf{I} & \mathbf{0} \\ r\,\nabla_{\boldsymbol{\theta}}\bar{\boldsymbol{h}}(\boldsymbol{\theta}_t) & \mathbf{0} & \mathbf{I} \end{pmatrix}. \tag{25}$$

At the fixed point. The log-states satisfy $z_{i,t} = z_{i,t-1}$ and $w_{j,t} = w_{j,t-1}$. Since $r > 0$, Eqs. (13) and (14) give $\bar{g}_i(\boldsymbol{\theta}^\circ) = \bar{h}_j(\boldsymbol{\theta}^\circ) = 0$. The multipliers are therefore $e^{z_i^\circ}/a_i = \lambda_i^\circ$ and $\sinh(w_j^\circ)/b_j = \mu_j^\circ$. The entries of Eq. (25) evaluate to

$$\begin{gathered} \nabla_{\boldsymbol{\theta}}\bar{\boldsymbol{g}}_{\mathcal{A}}(\boldsymbol{\theta}^\circ) = \mathbf{N}_g, \qquad \nabla_{\boldsymbol{\theta}}\bar{\boldsymbol{h}}(\boldsymbol{\theta}^\circ) = \mathbf{N}_h, \qquad \nabla^2_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}^\circ, \boldsymbol{\lambda}^\circ, \boldsymbol{\mu}^\circ) = \boldsymbol{H}, \\ \mathbf{D}_g = \operatorname{diag}(\lambda_i^\circ), \qquad \mathbf{D}_h = \operatorname{diag}\big(\cosh(w_j^\circ)/b_j\big). \end{gathered} \tag{26}$$

Since $\mathbf{N}_g$ has rows $a_i^{-1}\nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}^\circ)$, $\nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}^\circ)^\top = \mathbf{N}_g^\top \operatorname{diag}(a_i)$, and $\operatorname{diag}(a_i)\mathbf{D}_g = \operatorname{diag}(a_i\lambda_i^\circ) = \boldsymbol{\Phi}_g$; likewise $\nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{\theta}^\circ)^\top = \mathbf{N}_h^\top \operatorname{diag}(b_j)$ and $\operatorname{diag}(b_j)\mathbf{D}_h = \operatorname{diag}(\cosh w_j^\circ) = \boldsymbol{\Phi}_h$. Hence

$$\nabla_{\boldsymbol{\theta}}\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}^\circ)^\top\mathbf{D}_g = \mathbf{N}_g^\top\boldsymbol{\Phi}_g, \qquad \nabla_{\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{\theta}^\circ)^\top\mathbf{D}_h = \mathbf{N}_h^\top\boldsymbol{\Phi}_h. \tag{27}$$

So $\mathbf{M}_t = \mathbf{M}$ by Eq. (24), and Eq. (25) becomes Eq. (17). □

Lemma 5 (Spectral-radius bound near the identity)**.** *Let $\mathbf{Q}$ be a square matrix all of whose eigenvalues have negative real part, and let $(\mathbf{Q}_\eta)_{\eta>0}$ be matrices of the same size with $\|\mathbf{Q}_\eta - \mathbf{Q}\|_{\text{op}} \to 0$ as $\eta \to 0$. Then $\mathbf{I} + \eta\,\mathbf{Q}_\eta$ has spectral radius below one for all sufficiently small $\eta > 0$.*

Proof. We show that all eigenvalues of $\mathbf{I} + \eta\mathbf{Q}_\eta$ have absolute value below one. They have the form $1 + \eta\sigma'$, where σ' is an eigenvalue of $\mathbf{Q}_\eta$, so we need $|1 + \eta\sigma'| < 1$ for every such σ'. Expanding gives

$$\begin{aligned} |1 + \eta\sigma'|^2 &= (1 + \eta\,\operatorname{Re}\sigma')^2 + \eta^2(\operatorname{Im}\sigma')^2 \\ &= 1 + 2\eta\,\operatorname{Re}\sigma' + \eta^2|\sigma'|^2. \end{aligned} \tag{28}$$

We bound $\operatorname{Re}\sigma'$ and $|\sigma'|$ independently of η.

Since $\mathbf{Q}$ has finitely many eigenvalues, all with negative real part, there is $\varsigma > 0$ such that $\operatorname{Re}\sigma \le -\varsigma$ for every eigenvalue σ of $\mathbf{Q}$. Eigenvalues vary continuously with the matrix entries (Horn and Johnson, 2012, App. D). Hence there is $\delta > 0$ such that any matrix within δ of $\mathbf{Q}$ in operator norm has every eigenvalue within $\varsigma/2$ of some eigenvalue of $\mathbf{Q}$.

By the assumed convergence, choose $\eta_0 > 0$ so that $\|\mathbf{Q}_\eta - \mathbf{Q}\|_{\text{op}} < \min(\delta, 1)$ whenever $0 < \eta < \eta_0$. For each such η and each eigenvalue σ' of $\mathbf{Q}_\eta$, the nearby eigenvalue σ of $\mathbf{Q}$ gives

$$\operatorname{Re}\sigma' \le \operatorname{Re}\sigma + |\sigma' - \sigma| \le -\varsigma + \varsigma/2 = -\varsigma/2.$$

For the magnitude, take an eigenvector $\boldsymbol{x}$ for σ' with $\|\boldsymbol{x}\| = 1$. Since $\mathbf{Q}_\eta\boldsymbol{x} = \sigma'\boldsymbol{x}$, the definition of the operator norm and the triangle inequality give

$$\begin{aligned} |\sigma'| = \|\mathbf{Q}_\eta\boldsymbol{x}\| &\le \|\mathbf{Q}_\eta\|_{\text{op}} \\ &\le \|\mathbf{Q}\|_{\text{op}} + \|\mathbf{Q}_\eta - \mathbf{Q}\|_{\text{op}} \le \|\mathbf{Q}\|_{\text{op}} + 1 =: K. \end{aligned}$$

The constants ς and K are independent of η and of the chosen eigenvalue.

Substituting these bounds in Eq. (28) gives

$$|1 + \eta\sigma'|^2 \le 1 - \eta\varsigma + \eta^2K^2.$$

If

$$0 < \eta < \min\{\eta_0, \varsigma/K^2\},$$

then $-\eta\varsigma + \eta^2K^2 = -\eta(\varsigma - \eta K^2) < 0$. Hence every eigenvalue of $\mathbf{I} + \eta\mathbf{Q}_\eta$ has absolute value below one, so its spectral radius is below one. □

Proof of Proposition 2, inactive multipliers held at zero. The argument adapts the local analysis of the quadratic method of multipliers (Bertsekas, 1982, Ch. 2) to Algorithm 1.

Let $\boldsymbol{\theta}^\star$ be a local minimizer of Eq. (1) satisfying Assumptions 1 to 3, and let $\mathcal{A}$ be its active set. LICQ gives unique KKT multipliers $(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$. Let F be one iteration of Algorithm 1 on the state $(\boldsymbol{\theta}; \boldsymbol{z}; \boldsymbol{w})$, with a plain gradient primal step of size $\eta_\theta > 0$. Hold the constraint scales fixed at positive values a_i, b_j and take $z_{\max} = \infty$, as in §3.1.

Fix a ratio $\zeta > 0$ and tie the dual step size to the primal one:

$$r = \zeta \eta_\theta.$$

For the active inequality log-states, suppose

$$z_{\min} < \log(a_i \lambda_i^\star), \qquad i \in \mathcal{A}.$$

Reduction to the active set. Let $F_\mathcal{A}$ be F with the inactive-inequality multipliers held at zero and their log-states omitted. Where no clip binds, it is the modified AdaMult iteration of Eqs. (13) to (15) on $(\boldsymbol{\theta}; \boldsymbol{z}_\mathcal{A}; \boldsymbol{w})$. The second part considers the actual iteration F, with inequality log-states projected onto $[z_{\min}, \infty)$.

Fixed point. Let $z_i^\star, w_j^\star$ be as in Eq. (16), and write

$$\boldsymbol{s}_0^\star = (\boldsymbol{\theta}^\star; \boldsymbol{z}_\mathcal{A}^\star; \boldsymbol{w}^\star).$$

We use Lemmas 3 and 4 to establish that $\boldsymbol{s}_0^\star$ is a fixed point of $F_\mathcal{A}$ and compute its Jacobian there. To apply these lemmas, we first show that no clip binds near $\boldsymbol{s}_0^\star$. At the KKT point, primal feasibility and the definition of $\mathcal{A}$ give

$$\bar{g}_i(\boldsymbol{\theta}^\star) = 0 \quad (i \in \mathcal{A}), \qquad \bar{h}_j(\boldsymbol{\theta}^\star) = 0 \quad \text{for every } j.$$

These values are strictly inside $[-c, c]$. The updated and extrapolated active log-states both equal $z_i^\star > z_{\min}$ there. By continuity, these strict inequalities persist in a neighborhood of $\boldsymbol{s}_0^\star$, so no clip binds on that neighborhood.

Thus, by Lemma 3, $\boldsymbol{s}_0^\star$ is a fixed point of $F_\mathcal{A}$, and the multipliers it supplies to the primal step are $(\boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$.

Jacobian. By Definition 2, local stability requires the Jacobian of $F_\mathcal{A}$ at $\boldsymbol{s}_0^\star$ to have spectral radius below one. By Lemma 5, it suffices to write it as $\mathbf{I} + \eta_\theta \mathbf{Q}_{\eta_\theta}$, where $\mathbf{Q}_{\eta_\theta}$ converges to a matrix $\mathbf{Q}$ whose eigenvalues have negative real part.

Lemma 4 gives this Jacobian as $\mathcal{J}_{\eta_\theta}$ in Eq. (17). All the matrices below are evaluated at the KKT triple $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$. Write

$$\boldsymbol{P} \triangleq \boldsymbol{H} + p\,\mathbf{M}.$$

With $r = \zeta \eta_\theta$, the decomposition is

$$\begin{aligned} \mathcal{J}_{\eta_\theta} &= \mathbf{I} + \eta_\theta \mathbf{Q}_{\eta_\theta}, \\ \mathbf{Q}_{\eta_\theta} &= \mathbf{Q} - \eta_\theta \begin{pmatrix} \zeta\mathbf{M} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \\ \mathbf{Q} &= \begin{pmatrix} -\boldsymbol{P} & -\mathbf{N}_g^\top \boldsymbol{\Phi}_g & -\mathbf{N}_h^\top \boldsymbol{\Phi}_h \\ \zeta \mathbf{N}_g & \mathbf{0} & \mathbf{0} \\ \zeta \mathbf{N}_h & \mathbf{0} & \mathbf{0} \end{pmatrix}. \end{aligned} \tag{29}$$

For fixed p, the matrices on the right are independent of η_θ, so

$$\|\mathbf{Q}_{\eta_\theta} - \mathbf{Q}\|_{\mathrm{op}} = \eta_\theta \zeta \|\mathbf{M}\|_{\mathrm{op}} \to 0.$$

To apply Lemma 5, it therefore remains to choose p so that every eigenvalue of $\mathbf{Q}$ has negative real part.

Eigenvalue condition. We determine the eigenvalues of $\mathbf{Q}$ through a similar matrix, obtained by rescaling the dual coordinates so that the two off-diagonal blocks become opposite transposes. Strict complementarity and positive scales give

$$\boldsymbol{\Phi}_g = \mathrm{diag}(a_i \lambda_i^\star)_{i \in \mathcal{A}} \succ \mathbf{0}, \qquad \boldsymbol{\Phi}_h = \mathrm{diag}(\cosh w_j^\star) \succeq \mathbf{I}.$$

Collect the weighted constraint Jacobians in

$$\mathbf{B} \triangleq \begin{pmatrix} \mathbf{\Phi}_g^{1/2}\mathbf{N}_g \\ \mathbf{\Phi}_h^{1/2}\mathbf{N}_h \end{pmatrix},$$

and use the invertible change of coordinates

$$\mathbf{S} = \operatorname{diag}\big(\mathbf{I},\, \zeta^{-1/2}\mathbf{\Phi}_g^{1/2},\, \zeta^{-1/2}\mathbf{\Phi}_h^{1/2}\big).$$

After stacking the two dual blocks, this gives

$$\mathbf{S}\mathbf{Q}\mathbf{S}^{-1} = \tilde{\mathbf{Q}} = \begin{pmatrix} -\boldsymbol{P} & -\mathbf{C}^\top \\ \mathbf{C} & \mathbf{0} \end{pmatrix}, \qquad \mathbf{C} = \sqrt{\zeta}\,\mathbf{B}. \tag{30}$$

For matrices of this form, $\boldsymbol{P} \succ \mathbf{0}$ and full row rank of $\mathbf{C}$ ensure that every eigenvalue has negative real part. This follows from Benzi et al. (2005, Thm. 3.6(iii)), applied to $-\tilde{\mathbf{Q}}$.

Damping threshold. We now verify the two sufficient conditions: $\mathbf{C}$ has full row rank, and

$$\boldsymbol{P} = \boldsymbol{H} + p\,\mathbf{M} \succ \mathbf{0} \qquad \text{for every } p > \bar{p} \tag{31}$$

for some finite $\bar{p} \geq 0$. The effective primal curvature $\boldsymbol{P}$ is the Hessian at $\boldsymbol{\theta}^\star$ of the weighted quadratic augmented Lagrangian

$$\mathcal{L}(\cdot, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) + \tfrac{p}{2}\big(\bar{\boldsymbol{g}}_{\mathcal{A}}^\top \mathbf{\Phi}_g \bar{\boldsymbol{g}}_{\mathcal{A}} + \bar{\boldsymbol{h}}^\top \mathbf{\Phi}_h \bar{\boldsymbol{h}}\big),$$

since the active constraints and equalities vanish there.

To verify full row rank of $\mathbf{C} = \sqrt{\zeta}\,\mathbf{B}$, write $\mathbf{B}$ in terms of the active constraint gradients. By Eq. (18),

$$\begin{aligned} \mathbf{M} &= \mathbf{B}^\top \mathbf{B}, \\ \mathbf{B} &= \begin{pmatrix} \mathbf{\Phi}_g^{1/2}\operatorname{diag}(a_i)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Phi}_h^{1/2}\operatorname{diag}(b_j)^{-1} \end{pmatrix} \begin{pmatrix} \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}^\star) \\ \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}^\star) \end{pmatrix}. \end{aligned} \tag{32}$$

The first factor in $\mathbf{B}$ is invertible, and the second has full row rank by LICQ. Thus $\mathbf{B}$ and $\mathbf{C} = \sqrt{\zeta}\,\mathbf{B}$ have full row rank.

Also, $\mathbf{M} \succeq \mathbf{0}$, and $\ker \mathbf{B}$ is the tangent space at $\boldsymbol{\theta}^\star$ to the manifold defined by $\boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}) = \mathbf{0}$ and $\boldsymbol{h}(\boldsymbol{\theta}) = \mathbf{0}$:

$$\mathbf{B}\boldsymbol{x} = \mathbf{0} \quad \Longleftrightarrow \quad \nabla_{\boldsymbol{\theta}} \boldsymbol{g}_{\mathcal{A}}(\boldsymbol{\theta}^\star)\boldsymbol{x} = \mathbf{0} \quad \text{and} \quad \nabla_{\boldsymbol{\theta}} \boldsymbol{h}(\boldsymbol{\theta}^\star)\boldsymbol{x} = \mathbf{0}.$$

SOSC therefore gives

$$\boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{x} > 0 \qquad \text{for every nonzero } \boldsymbol{x} \in \ker \mathbf{B}.$$

Finsler's lemma (Finsler, 1937) then gives $\boldsymbol{H} + t\mathbf{B}^\top\mathbf{B} \succ \mathbf{0}$ for some finite $t \in \mathbb{R}$. Because $\mathbf{M} = \mathbf{B}^\top\mathbf{B} \succeq \mathbf{0}$, increasing t preserves positive definiteness. Hence the threshold

$$\bar{p} \triangleq \inf\{p \geq 0 : \boldsymbol{H} + p\,\mathbf{M} \succ \mathbf{0}\}$$

is finite and satisfies Eq. (31).

Fix any $p > \bar{p}$. Both conditions above now hold, so every eigenvalue of $\tilde{\mathbf{Q}}$, and hence of $\mathbf{Q}$, has negative real part. Since $\mathbf{Q}_{\eta_\theta} \to \mathbf{Q}$, Lemma 5 gives spectral radius below one for $\mathcal{J}_{\eta_\theta}$ whenever $\eta_\theta > 0$ is sufficiently small, with $r = \zeta\eta_\theta$. Thus $\boldsymbol{s}_0^\star$ is a locally stable fixed point of $F_{\mathcal{A}}$, corresponding to $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$. □

Proof of Proposition 2, inactive multipliers at a positive floor. Keep the choices of p, η_θ and $r = \zeta\eta_\theta$ for which we proved local stability of $F_{\mathcal{A}}$ at $\boldsymbol{s}_0^\star$. We now use AdaMult's iteration F under the same assumptions, but also update the inactive inequality multipliers. We show that, for sufficiently negative $z_{\min}$, F has a locally stable fixed point whose primal variables and output multipliers approach the KKT triple $(\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star)$ as $z_{\min} \to -\infty$.

Effect of the projection. For the active inequalities and equalities, the constraint values vanish at $\boldsymbol{\theta}^\star$. By continuity, we can choose a fixed neighborhood U of $\boldsymbol{s}_0^\star$ where their normalized values lie in $(-c, c)$ and the updated and extrapolated active inequality log-states are bounded below. For all

sufficiently negative $z_{\min}$, no active clip binds on U. The restriction of $F_{\mathcal{A}}$ to U is therefore the same smooth map Eqs. (13) to (15) for all these floors.

For the inactive inequalities, $g_i(\boldsymbol{\theta}^\star) < 0$, and clipping preserves the negative sign. By shrinking U, we can choose $\omega > 0$ such that

$$\check{g}_i(\boldsymbol{\theta}) \triangleq \texttt{clip}(g_i(\boldsymbol{\theta})/a_i, -c, c) \le -\omega, \qquad i \notin \mathcal{A},$$

throughout U. For inactive log-states with $|z_i - z_{\min}| < r\omega$, the dual update and extrapolation give

$$\max\{z_i + r\check{g}_i(\boldsymbol{\theta}), z_{\min}\} = z_{\min}, \qquad \max\{z_{\min} + p\check{g}_i(\boldsymbol{\theta}), z_{\min}\} = z_{\min}.$$

Thus F maps these log-states to $z_{\min}$ and supplies the constant inactive inequality multipliers $e^{z_{\min}}/a_i$ to the primal step.

Perturbed iteration and fixed point. Set $\xi \triangleq e^{z_{\min}}$, so the inactive inequality multipliers are ξ/a_i. The primal gradient in the full iteration F is

$$\underbrace{\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) + \sum_{i\in\mathcal{A}} \lambda_i \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}) + \sum_j \mu_j \nabla_{\boldsymbol{\theta}} h_j(\boldsymbol{\theta})}_{\text{primal gradient in } F_{\mathcal{A}}} + \underbrace{\xi \sum_{i\notin\mathcal{A}} a_i^{-1} \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta})}_{\text{inactive inequalities}},$$

where λ_i, μ_j are the output multipliers of Eqs. (13) and (14), held fixed when differentiating the Lagrangian.

Let G_ξ denote the resulting update on the reduced state $\boldsymbol{s} = (\boldsymbol{\theta}; \boldsymbol{z}_{\mathcal{A}}; \boldsymbol{w})$. The inactive-inequality term changes only the primal step of Eq. (15), giving, on U,

$$G_\xi(\boldsymbol{s}) = F_{\mathcal{A}}(\boldsymbol{s}) - \eta_\theta \xi \begin{pmatrix} \sum_{i\notin\mathcal{A}} a_i^{-1} \nabla_{\boldsymbol{\theta}} g_i(\boldsymbol{\theta}) \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}. \tag{33}$$

The two zero blocks correspond to the unchanged updates of the active inequality log-states $\boldsymbol{z}_{\mathcal{A}}$ and equality log-states $\boldsymbol{w}$. Ordering the reduced coordinates first and the inactive inequality log-states $\boldsymbol{z}_{\mathcal{A}^c}$ last, the full update in this neighborhood is

$$F(\boldsymbol{s}, \boldsymbol{z}_{\mathcal{A}^c}) = \big(G_\xi(\boldsymbol{s}),\ z_{\min}\mathbf{1}\big).$$

The formula in Eq. (33) defines a C^1 map in $(\boldsymbol{s}, \xi)$ for real ξ near zero, with $G_0 = F_{\mathcal{A}}$ on U. The extra primal-gradient term generally shifts the fixed point. To show that a fixed point still exists near $\boldsymbol{s}_0^\star$, we apply the implicit function theorem to $G_\xi(\boldsymbol{s}) - \boldsymbol{s} = \mathbf{0}$ at $(\boldsymbol{s}_0^\star, 0)$. Its derivative with respect to $\boldsymbol{s}$ is $\mathcal{J}_{\eta_\theta} - \mathbf{I}$, which is invertible because $\mathcal{J}_{\eta_\theta}$ has spectral radius below one. Thus, for every sufficiently small $\xi > 0$, there is a fixed point $\boldsymbol{s}_\xi^\circ = (\boldsymbol{\theta}_\xi^\circ; \boldsymbol{z}_{\mathcal{A},\xi}^\circ; \boldsymbol{w}_\xi^\circ)$ satisfying

$$G_\xi(\boldsymbol{s}_\xi^\circ) = \boldsymbol{s}_\xi^\circ, \qquad \boldsymbol{s}_\xi^\circ \to \boldsymbol{s}_0^\star \quad \text{as } \xi \to 0.$$

For small ξ, $\boldsymbol{s}_\xi^\circ$ lies in U. Adjoining $z_i = z_{\min}$ for each inactive inequality then gives a fixed point of the full iteration F: the reduced coordinates are fixed by G_ξ, and the inactive log-states remain at their floor.

Stability. The reduced fixed point $\boldsymbol{s}_\xi^\circ$ is near $\boldsymbol{s}_0^\star$. We now check local stability of the corresponding full fixed point of F. In the same coordinate order, the Jacobian of F at this full fixed point is

$$\begin{pmatrix} D_{\boldsymbol{s}} G_\xi(\boldsymbol{s}_\xi^\circ) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}.$$

The zero blocks follow because the inactive log-states and their output multipliers are locally constant after the update. Since G_ξ is C^1 and $\boldsymbol{s}_\xi^\circ \to \boldsymbol{s}_0^\star$,

$$D_{\boldsymbol{s}} G_\xi(\boldsymbol{s}_\xi^\circ) \to D_{\boldsymbol{s}} F_{\mathcal{A}}(\boldsymbol{s}_0^\star) = \mathcal{J}_{\eta_\theta} \quad \text{as } \xi \to 0.$$

By continuity of the spectral radius (Horn and Johnson, 2012, App. D), the reduced block retains spectral radius below one for sufficiently small ξ. The remaining eigenvalues are zero, so the full fixed point is locally stable.

Let $(\boldsymbol{\lambda}_\xi^\circ, \boldsymbol{\mu}_\xi^\circ)$ be its output multipliers. The active inequality and equality outputs in Eqs. (13) and (14) are continuous functions of the reduced state, so they converge to their KKT values as $\boldsymbol{s}_\xi^\circ \to \boldsymbol{s}_0^\star$. The inactive outputs satisfy $\xi/a_i \to 0 = \lambda_i^\star$. Hence

$$(\boldsymbol{\theta}_\xi^\circ, \boldsymbol{\lambda}_\xi^\circ, \boldsymbol{\mu}_\xi^\circ) \to (\boldsymbol{\theta}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) \qquad \text{as } z_{\min} \to -\infty.$$

□

A.5 Stability and Scale Sensitivity of RMSProp and RMS-ALM (Propositions 3, 5 and 6)

We compare dual RMSProp, RMS-ALM and AdaMult, each paired with one primal gradient step, on the problem of Eq. (6):

$$\min_{x,y} \quad f(x,y) = \frac{x^4}{4} - \frac{\kappa x^2}{2} + \frac{y^2}{2} \qquad \text{subject to } h(x) = \gamma x = 0, \qquad \kappa, \gamma > 0.$$

The constraint fixes $x = 0$, leaving the objective $y^2/2$ on the feasible set. Thus $(0, 0)$ is the unique constrained minimizer, even though the objective curves downward in the x direction there. The x^4 term keeps the objective bounded below without changing this local curvature.

The constraint gradient at the solution is $\nabla h(0, 0) = (\gamma, 0) \neq 0$, so LICQ holds (Assumption 3). Stationarity gives $\mu^\star = 0$, and

$$\nabla^2_{(x,y)} \mathcal{L}(0, 0, \mu^\star) = \begin{pmatrix} -\kappa & 0 \\ 0 & 1 \end{pmatrix}.$$

Every nonzero feasible tangent direction is $(0, d_y)$ and has positive curvature d_y^2, so SOSC holds (Assumption 2). We will show that stabilizing this minimizer requires damping, supplied here by dual extrapolation, to counter the negative curvature normal to the constraint. RMSProp has no such damping. RMS-ALM and AdaMult do, but only AdaMult's stability conditions are unchanged when we rescale the constraint by changing γ.

For this theoretical comparison, all three methods use decay $\beta \in [0, 1)$, $\epsilon = 0$ and a relative RMS floor with $\delta \in (0, 1)$. Only AdaMult uses bias correction, as in Algorithm AdaMult. The relative floor modifies the RMS baselines, whose usual updates use an absolute ϵ. The Lagrangian of Eq. (6) is

$$\mathcal{L}(x, y, \mu) = \frac{x^4}{4} - \frac{\kappa x^2}{2} + \frac{y^2}{2} + \mu\gamma x.$$

Since $\partial_y \mathcal{L} = y$, every method's primal step gives $y_{t+1} = y_t - \eta_\theta y_t$.

*Proof of Proposition 3 (*RMSProp*).* RMSProp updates its second-moment estimate and multiplier from $h(x_t)$, then takes the primal step using the new multiplier. With $v_0 = v_0^{\max} = 0$ and $\mu_1 = 0$, the updates for $t \geq 1$ are

$$v_t = \beta v_{t-1} + (1 - \beta)(\gamma x_t)^2, \tag{34a}$$

$$v_t^{\max} = \max(v_{t-1}^{\max}, v_t), \tag{34b}$$

$$a_t = \begin{cases} 1, & v_t^{\max} = 0, \\ \max(\sqrt{v_t}, \delta\sqrt{v_t^{\max}}), & v_t^{\max} > 0, \end{cases} \tag{34c}$$

$$\mu_{t+1} = \mu_t + \eta\, \gamma x_t / a_t, \tag{34d}$$

$$x_{t+1} = x_t - \eta_\theta(x_t^3 - \kappa x_t + \gamma\mu_{t+1}), \tag{34e}$$

$$y_{t+1} = y_t - \eta_\theta y_t. \tag{34f}$$

At the constrained minimizer $(x^\star, y^\star) = (0, 0)$, setting $\mu = v = 0$ and the stored peak $v^{\max}$ to any $m > 0$ gives a fixed point of Eq. (34), with RMS scale

$$a = \delta\sqrt{m} > 0.$$

Near this fixed point, the relative floor keeps the scale fixed at a. Indeed, in a sufficiently small neighborhood of $(x, v_{t-1}) = (0, 0)$,

$$v_t = \beta v_{t-1} + (1 - \beta)(\gamma x_t)^2 < \delta^2 m < m.$$

Thus the peak remains m and $a_t = a$. As in §3.1, we assess local stability with the scale held fixed: here this means holding the stored peak at m.

With the peak held at m, the remaining state is

$$\boldsymbol{s}_t = (x_t, \mu_t, y_t, v_{t-1}).$$

To calculate its Jacobian at $\boldsymbol{s}^\star = (0,0,0,0)$, substitute the dual step into the primal step of Eq. (34):

$$x_{t+1} = x_t - \eta_\theta \left(x_t^3 - \kappa x_t + \gamma \mu_t + \frac{\eta\gamma^2}{a} x_t \right).$$

Differentiating the four state updates gives the full Jacobian

$$\mathbf{J}_{\mathrm{RMSProp}} = \begin{pmatrix} 1 + \eta_\theta\kappa - \eta_\theta\eta\gamma^2/a & -\eta_\theta\gamma & 0 & 0 \\ \eta\gamma/a & 1 & 0 & 0 \\ 0 & 0 & 1-\eta_\theta & 0 \\ 0 & 0 & 0 & \beta \end{pmatrix}.$$

Its upper-left 2×2 block, denoted $\mathbf{B}_{\mathrm{RMSProp}}$, has determinant

$$\det \mathbf{B}_{\mathrm{RMSProp}} = 1 + \eta_\theta\kappa > 1.$$

At least one of this block's eigenvalues therefore has absolute value greater than one. It is also an eigenvalue of $\mathbf{J}_{\mathrm{RMSProp}}$, so the fixed point is unstable for every positive primal step size η_θ, dual step size η, and RMS scale a. □

*Proof of Proposition 5 (*RMS-ALM*).* RMS-ALM takes the primal step first, using its stored multiplier $\hat{\mu}_{t-1}$ and penalty ρ_{t-1}. It then updates the second-moment estimate, penalty and multiplier from the new residual $h(x_{t+1})$. We allow the dual step size $\eta > 0$ and damping coefficient $\rho > 0$ to be chosen independently. This encompasses the tied choice $\eta = \rho$ of Basir and Senocak (2023, Algorithm 3). For this analysis, initialize $v_0 = v_0^{\max} = \hat{\mu}_0 = \rho_0 = 0$. The updates for $t \geq 1$ are

$$x_{t+1} = x_t - \eta_\theta\big(x_t^3 - \kappa x_t + \gamma(\hat{\mu}_{t-1} + \rho_{t-1}\gamma x_t)\big), \tag{35a}$$

$$y_{t+1} = y_t - \eta_\theta y_t, \tag{35b}$$

$$v_t = \beta v_{t-1} + (1-\beta)(\gamma x_{t+1})^2, \tag{35c}$$

$$v_t^{\max} = \max(v_{t-1}^{\max}, v_t), \tag{35d}$$

$$a_t = \begin{cases} 1, & v_t^{\max} = 0, \\ \max(\sqrt{v_t}, \delta\sqrt{v_t^{\max}}), & v_t^{\max} > 0, \end{cases} \tag{35e}$$

$$\rho_t = \rho / a_t, \tag{35f}$$

$$\hat{\mu}_t = \hat{\mu}_{t-1} + \eta\,\gamma x_{t+1}/a_t. \tag{35g}$$

At the constrained minimizer $(x^\star, y^\star) = (0,0)$, fix a stored peak $m > 0$ and write its RMS scale as

$$a = \delta\sqrt{m} = \gamma A, \qquad A = \frac{\delta\sqrt{m}}{\gamma} > 0.$$

Setting $\hat{\mu} = v = 0$ and the stored penalty to ρ/a gives a fixed point of Eq. (35). With the peak held at m, the scale is constant in a neighborhood of the fixed point: the same bound used in the RMSPROP proof applies with the observed residual γx_{t+1}, which is continuous in the current state.

We calculate the full Jacobian with the peak held at m, in the state order

$$\boldsymbol{s}_t = (x_t, \hat{\mu}_{t-1}, y_t, v_{t-1}, \rho_{t-1}).$$

First differentiate the primal step Eq. (35a) at the fixed point $\boldsymbol{s}^\star = (0,0,0,0,\rho/a)$, writing d for its derivative with respect to x_t:

$$\frac{\partial x_{t+1}}{\partial x_t} = 1 + \eta_\theta\kappa - \eta_\theta\frac{\rho\gamma^2}{a} = d,$$

$$\frac{\partial x_{t+1}}{\partial \hat{\mu}_{t-1}} = -\eta_\theta\gamma,$$

$$\frac{\partial x_{t+1}}{\partial \rho_{t-1}} = -\eta_\theta\gamma^2 x^\star = 0.$$

The dual step Eq. (35g) uses x_{t+1}, so the chain rule gives

$$\frac{\partial \hat{\mu}_t}{\partial x_t} = \frac{\eta\gamma}{a}\frac{\partial x_{t+1}}{\partial x_t} = \frac{\eta\gamma}{a} d,$$

$$\frac{\partial \hat{\mu}_t}{\partial \hat{\mu}_{t-1}} = 1 + \frac{\eta\gamma}{a}\frac{\partial x_{t+1}}{\partial \hat{\mu}_{t-1}} = 1 - \eta_\theta\frac{\eta\gamma^2}{a}.$$

The penalty update sets $\rho_t = \rho/a$ locally, so all of its derivatives are zero. Including the y and second-moment estimate updates, the full Jacobian is

$$\mathbf{J}_{\text{RMS-ALM}} = \begin{pmatrix} d & -\eta_\theta\gamma & 0 & 0 & 0 \\ (\eta\gamma/a)d & 1-\eta_\theta\eta\gamma^2/a & 0 & 0 & 0 \\ 0 & 0 & 1-\eta_\theta & 0 & 0 \\ 0 & 0 & 0 & \beta & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Denote its upper-left 2×2 block by $\mathbf{B}_{\text{RMS-ALM}}$. The other eigenvalues, $1-\eta_\theta$, β and 0, have absolute value less than one for $0 < \eta_\theta < 2$ and $\beta \in [0, 1)$. Local stability therefore requires precisely that both eigenvalues of $\mathbf{B}_{\text{RMS-ALM}}$ have absolute value less than one.

For a real 2×2 matrix with trace T and determinant D, this holds if and only if (Elaydi, 2005, eq. (4.3.9))

$$1 - D > 0, \qquad 1 - T + D > 0, \qquad 1 + T + D > 0. \tag{36}$$

For $\mathbf{B}_{\text{RMS-ALM}}$,

$$T = 2 + \eta_\theta\kappa - \eta_\theta\frac{(\eta+\rho)\gamma^2}{a},$$
$$D = 1 + \eta_\theta\kappa - \eta_\theta\frac{\rho\gamma^2}{a}.$$

Substituting into Eq. (36) gives

$$\eta_\theta\left(\frac{\rho\gamma^2}{a} - \kappa\right) > 0,$$
$$\eta_\theta\frac{\eta\gamma^2}{a} > 0,$$
$$4 - \eta_\theta\left(\frac{(\eta+2\rho)\gamma^2}{a} - 2\kappa\right) > 0.$$

The middle inequality holds because $\eta > 0$. Substituting $a = \gamma A$ into the other two gives precisely

$$\rho > \frac{\kappa A}{\gamma}, \qquad \eta + 2\rho < \left(\frac{4}{\eta_\theta} + 2\kappa\right)\frac{A}{\gamma}.$$

If the first inequality is strictly reversed, $D > 1$, so an eigenvalue has absolute value above one. If the second is strictly reversed, the characteristic polynomial $\chi(z) = z^2 - Tz + D$ satisfies $\chi(-1) < 0$ and therefore has a real root below -1.

To establish instability at small constraint scales, consider runs from the same initialization. Since $\hat{\mu}_0 = \rho_0 = 0$, their first primal step is

$$\bar{x} = x_1 - \eta_\theta(x_1^3 - \kappa x_1),$$

independent of γ. Suppose $\bar{x} \neq 0$. We must show that no fixed pair (η, ρ) stabilizes the minimizer for all sufficiently small γ. The condition $\rho > \kappa A/\gamma$ alone does not establish this: A also depends on the run, and could in principle shrink with γ. We therefore use the first residual to bound A away from zero.

The first dual observation gives

$$v_1 = (1-\beta)\gamma^2\bar{x}^2.$$

Since the peak never decreases, any fixed point approached from this initialization must have

$$m \geq (1-\beta)\gamma^2\bar{x}^2, \qquad A = \frac{\delta\sqrt{m}}{\gamma} \geq \delta\sqrt{1-\beta}\,|\bar{x}|.$$

Consequently, at every scale

$$\gamma < \frac{\kappa\delta\sqrt{1-\beta}\,|\bar{x}|}{\rho}$$

we have $\rho\gamma/A < \kappa$, hence strict instability.

Thus, at a settled RMS scale γA, the locally stable damping range is

$$\frac{\kappa A}{\gamma} < \rho < \left(\frac{2}{\eta_\theta} + \kappa\right)\frac{A}{\gamma} - \frac{\eta}{2}.$$

Damping strictly below the lower bound or above the upper bound makes the fixed point unstable. Both bounds depend on A/γ. For the common initialization above, A stays bounded away from zero, so any fixed damping ρ eventually falls below the stable range as γ decreases. □

*Proof of Proposition 6 (*AdaMult*).* AdaMult updates its log-state from $h(x_t)$ and extrapolates it before taking the primal step. We use the equality updates of Algorithm AdaMult with positive clipping bounds $c, z_{\max} > 0$ and initialize $u_0 = \widehat{u}_0^{\max} = w_0 = 0$. For $t \geq 1$,

$$u_t = \beta u_{t-1} + (1-\beta)(\gamma x_t)^2, \tag{37a}$$
$$\widehat{u}_t = u_t/(1-\beta^t), \tag{37b}$$
$$\widehat{u}_t^{\max} = \max(\widehat{u}_{t-1}^{\max}, \widehat{u}_t), \tag{37c}$$
$$b_t = \begin{cases} 1, & \widehat{u}_t^{\max} = 0, \\ \max(\sqrt{\widehat{u}_t}, \delta\sqrt{\widehat{u}_t^{\max}}), & \widehat{u}_t^{\max} > 0, \end{cases} \tag{37d}$$
$$\bar{h}_t = \texttt{clip}(\gamma x_t/b_t, -c, c), \tag{37e}$$
$$w_t = \texttt{clip}(w_{t-1} + r\bar{h}_t, -z_{\max}, z_{\max}), \tag{37f}$$
$$\widetilde{w}_t = \texttt{clip}(w_t + p\bar{h}_t, -z_{\max}, z_{\max}), \tag{37g}$$
$$\mu_{t+1} = \sinh(\widetilde{w}_t)/b_t, \tag{37h}$$
$$x_{t+1} = x_t - \eta_\theta(x_t^3 - \kappa x_t + \gamma\mu_{t+1}), \tag{37i}$$
$$y_{t+1} = y_t - \eta_\theta y_t. \tag{37j}$$

At the constrained minimizer $(x^\star, y^\star) = (0,0)$, setting $w = u = 0$ and the stored peak to any $m > 0$ gives a fixed point of Eq. (37), with RMS scale

$$b = \delta\sqrt{m} = \gamma A, \qquad A = \frac{\delta\sqrt{m}}{\gamma} > 0.$$

With the peak held at m, the relative floor keeps $b_t = b$ locally. Indeed, $1 - \beta^t \geq 1 - \beta > 0$ for $t \geq 1$, so near $(x, u_{t-1}) = (0,0)$,

$$\widehat{u}_t \leq \frac{\beta u_{t-1} + (1-\beta)(\gamma x_t)^2}{1-\beta} < \delta^2 m < m.$$

The clips are also inactive near $x = w = 0$, since $c, z_{\max} > 0$. The local log-state and multiplier updates therefore reduce to

$$w_t = w_{t-1} + r\gamma x_t/b,$$
$$\mu_{t+1} = \frac{\sinh(w_{t-1} + (r+p)\gamma x_t/b)}{b}.$$

We calculate the full Jacobian with the peak held at m, in the state order

$$\boldsymbol{s}_t = (x_t, w_{t-1}, y_t, u_{t-1}).$$

At the fixed point $\boldsymbol{s}^\star = (0,0,0,0)$, the output multiplier has derivatives

$$\frac{\partial \mu_{t+1}}{\partial x_t} = \frac{(r+p)\gamma}{b^2},$$
$$\frac{\partial \mu_{t+1}}{\partial w_{t-1}} = \frac{1}{b}.$$

These use $\cosh(0) = 1$. Together with $\partial w_t/\partial x_t = r\gamma/b$ and the y and second-moment estimate updates, they give

$$\mathbf{J}_{\text{AdaMult}} = \begin{pmatrix} 1 + \eta_\theta\kappa - \eta_\theta(r+p)\gamma^2/b^2 & -\eta_\theta\gamma/b & 0 & 0 \\ r\gamma/b & 1 & 0 & 0 \\ 0 & 0 & 1-\eta_\theta & 0 \\ 0 & 0 & 0 & \beta \end{pmatrix}.$$

Denote its upper-left 2×2 block by $\mathbf{B}_{\mathrm{AdaMult}}$. The other eigenvalues, $1 - \eta_\theta$ and β, have absolute value less than one for $0 < \eta_\theta < 2$ and $\beta \in [0, 1)$. Thus local stability is determined by $\mathbf{B}_{\mathrm{AdaMult}}$.

Substituting $b = \gamma A$ into this block's trace and determinant gives

$$T = 2 + \eta_\theta \kappa - \eta_\theta \frac{r + p}{A^2},$$
$$D = 1 + \eta_\theta \kappa - \eta_\theta \frac{p}{A^2}.$$

The three conditions in Eq. (36) become

$$\eta_\theta \left(\frac{p}{A^2} - \kappa \right) > 0,$$
$$\eta_\theta \frac{r}{A^2} > 0,$$
$$4 - \eta_\theta \left(\frac{r + 2p}{A^2} - 2\kappa \right) > 0.$$

The middle inequality holds for $r > 0$, leaving

$$p > \kappa A^2, \qquad \eta_\theta \left(\frac{r + 2p}{A^2} - 2\kappa \right) < 4.$$

As for RMS-ALM, strictly reversing either inequality gives an eigenvalue with absolute value above one. Unlike RMS-ALM's conditions, these contain no constraint scale γ.

From a common initialization with $h(x_1) \neq 0$, Proposition 4 makes the primal iterates identical and the peaks proportional to γ^2, so $A = \delta\sqrt{m}/\gamma$ is independent of γ. □

A.6 Scale Invariance of AdaMult (Proposition 4)

Rescaling a constraint changes its RMS scale by the same factor, leaving the normalized constraint values inside AdaMult and its dual log-states unchanged. Dividing the extrapolated state by the RMS scale then cancels the change in the raw constraint, so the primal's Lagrangian is unchanged.

Proof of Proposition 4. The dual updates act independently on each constraint. We write the calculation for one inequality g, rescaled to αg with $\alpha > 0$, and omit its index. Let $\boldsymbol{\omega}_t$ denote the state of the primal optimizer. At the start of step t, the primal state is $(\boldsymbol{\theta}_t, \boldsymbol{\omega}_t)$, while the stored dual state $(z_{t-1}, v_{t-1}, \widehat{v}_{t-1}^{\max})$ comes from the dual update at $t - 1$ (or initialization when $t = 1$).

We show by induction that, at the start of each step $t \geq 1$,

$$\begin{aligned} (\boldsymbol{\theta}'_t, \boldsymbol{\omega}'_t) &= (\boldsymbol{\theta}_t, \boldsymbol{\omega}_t), \\ z'_{t-1} &= z_{t-1}, \\ v'_{t-1} &= \alpha^2 v_{t-1}, \\ \widehat{v}'^{\max}_{t-1} &= \alpha^2 \widehat{v}^{\max}_{t-1}, \end{aligned} \tag{38}$$

for each inequality, using its own rescaling factor α, with the corresponding relations for equalities. Step t will then return inversely rescaled multipliers.

Base case ($t = 1$). The runs share the same primal initialization, $(\boldsymbol{\theta}'_1, \boldsymbol{\omega}'_1) = (\boldsymbol{\theta}_1, \boldsymbol{\omega}_1)$. `AdaMult.init` sets $z'_0 = z_0 = 0$, $v'_0 = v_0 = 0$ and $\widehat{v}'^{\max}_0 = \widehat{v}^{\max}_0 = 0$, so Eq. (38) holds at $t = 1$. The equality states also start at zero.

Inductive step. Assume Eq. (38) and its equality counterpart hold at t. Since $\boldsymbol{\theta}'_t = \boldsymbol{\theta}_t$, the rescaled run observes $g'_t = \alpha g_t$. Its second moment, bias-corrected estimate and running peak satisfy

$$v'_t = \beta v'_{t-1} + (1 - \beta)\, g'^2_t = \alpha^2 \big(\beta v_{t-1} + (1 - \beta)\, g_t^2 \big) = \alpha^2 v_t, \tag{39}$$
$$\widehat{v}'_t = v'_t / (1 - \beta^t) = \alpha^2 \widehat{v}_t,$$
$$\widehat{v}'^{\max}_t = \max\big(\widehat{v}'^{\max}_{t-1}, \widehat{v}'_t \big) = \alpha^2 \widehat{v}^{\max}_t. \tag{40}$$

The nonzero constraint value at initialization gives $\widehat{v}_t^{\max} \geq \widehat{v}_1 = g_1^2 > 0$. By Eq. (40), the rescaled peak is also positive, so neither run uses the guard that sets the scale to 1 before the first nonzero observation. With $\epsilon = 0$ and $\delta > 0$, the scales satisfy

$$\begin{aligned} a_t &= \max\big(\sqrt{\widehat{v}_t},\, \delta\sqrt{\widehat{v}_t^{\max}}\big) > 0, \\ a'_t &= \max\big(\sqrt{\widehat{v}'_t},\, \delta\sqrt{\widehat{v}_t'^{\max}}\big) = \alpha\, a_t. \end{aligned} \tag{41}$$

The normalized constraint value is therefore unchanged:

$$\bar{g}'_t = \texttt{clip}\big(g'_t/a'_t,\, -c, c\big) = \texttt{clip}\big(g_t/a_t,\, -c, c\big) = \bar{g}_t. \tag{42}$$

With the same previous log-state, step size, damping coefficient and clip bounds, both runs produce identical updated and extrapolated log-states:

$$\begin{aligned} z'_t &= \texttt{clip}\big(z'_{t-1} + r\,\bar{g}'_t,\, z_{\min}, z_{\max}\big) = z_t, \\ \tilde{z}'_t &= \texttt{clip}\big(z'_t + p\,\bar{g}'_t,\, z_{\min}, z_{\max}\big) = \tilde{z}_t. \end{aligned} \tag{43}$$

The output multiplier consequently scales inversely:

$$\lambda'_{t+1} = e^{\tilde{z}'_t}/a'_t = e^{\tilde{z}_t}/(\alpha a_t) = \lambda_{t+1}/\alpha. \tag{44}$$

For equalities, the same calculations apply with $(h, \gamma, u, b, w, \sinh)$ in place of $(g, \alpha, v, a, z, \exp)$ and log-state bounds $[-z_{\max}, z_{\max}]$. In particular, $\mu'_{t+1} = \mu_{t+1}/\gamma$.

Let $\mathcal{L}'$ denote the Lagrangian of the rescaled problem. Holding the output multipliers fixed, the two Lagrangians agree for every $\boldsymbol{\theta}$:

$$\begin{aligned} \mathcal{L}'(\boldsymbol{\theta}, \boldsymbol{\lambda}'_{t+1}, \boldsymbol{\mu}'_{t+1}) &= f(\boldsymbol{\theta}) + \sum_i \lambda'_{i,t+1}\alpha_i g_i(\boldsymbol{\theta}) + \sum_j \mu'_{j,t+1}\gamma_j h_j(\boldsymbol{\theta}) \\ &= f(\boldsymbol{\theta}) + \sum_i \lambda_{i,t+1} g_i(\boldsymbol{\theta}) + \sum_j \mu_{j,t+1} h_j(\boldsymbol{\theta}) \\ &= \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\lambda}_{t+1}, \boldsymbol{\mu}_{t+1}). \end{aligned} \tag{45}$$

Their primal gradients at $\boldsymbol{\theta}_t$ therefore also agree. The deterministic primal optimizer receives the same gradient at the same state $(\boldsymbol{\theta}_t, \boldsymbol{\omega}_t)$ and therefore returns the same $(\boldsymbol{\theta}_{t+1}, \boldsymbol{\omega}_{t+1})$. Together with the moment and log-state relations, this establishes Eq. (38) and its equality counterpart at $t+1$, completing the induction. Hence $\boldsymbol{\theta}'_t = \boldsymbol{\theta}_t$ for every $t \geq 1$, and each constraint's output multiplier scales inversely with its rescaling factor. □

B Experimental Setup

Appendices B.1 and B.2 specify the baseline optimizers and hyperparameter choices, respectively. Appendices B.3 to B.8 describe each task's training and evaluation setup.

B.1 Baseline optimizers

GA uses the same ascent update as `PyTorch`'s SGD (Paszke et al., 2019) without momentum, projected for inequalities. PI control uses νPI (Sohrabi et al., 2024), as implemented in `Cooper` (v1.0.1; Gallego-Posada et al., 2025), with $\nu = 0$, $K_I = 1$ and proportional gain K_p. Adam (Kingma and Ba, 2015) and RMSProp (Tieleman and Hinton, 2012) use `PyTorch`'s default ascent updates:[7]

$$\text{GA:} \quad \lambda_{t+1} = \big[\lambda_t + \eta g_t\big]_+, \tag{46}$$

$$\text{PI control:} \quad \lambda_{t+1} = \big[\lambda_t + \eta g_t + \eta K_p(g_t - g_{t-1})\big]_+, \tag{47}$$

$$\text{Adam:} \quad \lambda_{t+1} = \big[\lambda_t + \eta \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)\big]_+, \tag{48}$$

$$\text{RMSProp:} \quad \lambda_{t+1} = \big[\lambda_t + \eta g_t/(\sqrt{v_t} + \epsilon)\big]_+, \tag{49}$$

[7] If a constraint is absent from a batch, every method freezes its multiplier and running estimates.

where g_t is the constraint residual, $\hat{m}_t, \hat{v}_t$ are Adam's bias-corrected moments, and $v_t = \beta v_{t-1} + (1-\beta)g_t^2$ is RMSProp's moving average of g_t^2. Equalities follow the same update rules but omit the projection. To match AdaMult's initialization, these baseline multipliers start at $\lambda_0 = 1$ for inequalities and $\mu_0 = 0$ for equalities.

RMS-ALM. Algorithm 3 of Basir and Senocak (2023) alternates minimization of an *augmented* Lagrangian with RMS-normalized dual ascent for equality constraints. We instead take a single primal optimizer step per dual update, written on the *plain* Lagrangian with an *extrapolated* multiplier, as in Eq. (ALM/PI) and AdaMult (Eq. (4)). For one inequality constraint, we add the two projections below:

$$\lambda_t = \left[\hat{\lambda}_t + \rho_t g_t\right]_+, \tag{50}$$

$$\boldsymbol{\theta}_{t+1} = \texttt{PrimalDescent}\big(\boldsymbol{\theta}_t, \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}_t, \lambda_t)\big), \tag{51}$$

$$g_{t+1} = g(\boldsymbol{\theta}_{t+1}), \tag{52}$$

$$v_{t+1} = \beta v_t + (1-\beta)g_{t+1}^2, \tag{53}$$

$$\eta_{t+1} = \rho_{t+1} = \eta/(\sqrt{v_{t+1}} + \epsilon), \tag{54}$$

$$\hat{\lambda}_{t+1} = \left[\hat{\lambda}_t + \eta_{t+1} g_{t+1}\right]_+. \tag{55}$$

Equations (50) to (55) are equivalent to Algorithm 3 of Basir and Senocak (2023) with its primal minimization replaced by a single optimizer step on the augmented Lagrangian and with projections added for inequalities.

The method ties the dual step size to the extrapolation gain, $\eta_t = \rho_t$. The published defaults are $\eta = 0.01$, $\beta = 0.99$, $\epsilon = 10^{-8}$ and $\rho_0 = 1$.

B.2 Hyperparameter choices

AdaMult's default. The decay $\beta = 0.99$ and absolute floor $\epsilon = 10^{-8}$ in Eq. (5) match `PyTorch`'s RMSProp defaults. We chose the step size $r = 0.01$, damping $p = 0.1$, and numerical guards during development, informed in part by synthetic diagnostics. One diagnostic used the scalar problem

$$\min_{\theta \in \mathbb{R}} \ \tfrac{1}{2}(\theta - 1)^2 \quad \text{s.t.} \quad a\theta \le 0.$$

We tested the inequality-constrained problem above, an equality-constrained variant with $a\theta = 0$, and a noisy inequality variant with Gaussian noise of standard deviation 0.5 added to constraint observations. The scale a ranged from 10^{-2} to 10^2 for the noiseless inequality variant and from 10^{-1} to 10 for the other two variants. Development runs explored grids over the step size r, damping p, RMS decay β, and relative floor δ. This default was used unchanged across all six tasks.

Baseline tuning. For every baseline, we vary the dual step size η on the grids in Table 3. PI control combines two sweeps: step size at $K_p = 1$, and gain K_p at $\eta = 0.1$ ($\eta = 0.035$ on Safe RL), with $\rho = \eta K_p$. All other parameters retain their defaults.

For the ingredient removals in Table 2, we tune only the dual step size: 16 values from $\{1, 2.5, 6.3\} \times 10^k$ in $[10^{-4}, 10]$ on the four supervised tasks, and five decade-spaced values on Safe RL (10^{-3} to 10) and Safe DPO (10^{-4} to 1). Across-task selection compares the four steps shared by all six grids, $\{10^{-3}, 10^{-2}, 10^{-1}, 1\}$.

Per-task selection. We assess feasibility using the criteria in §4. For each baseline and task, we select the configuration feasible on the most seeds, breaking ties by mean performance across all five seeds. If no configuration is feasible on any seed, we select the one with the smallest mean violation across seeds.

Across-task selection. For each baseline, we consider only configurations evaluated on all six tasks: 16 for GA and 13 for each other baseline. The shared PI configurations all have $K_p = 1$. We pool each configuration's 30 runs (five seeds per task) and select the one with the most feasible runs.

Performance breaks ties: we rank configurations within each task by their mean score across the five seeds, then choose the tied configuration with the best average rank across tasks. We use ranks because accuracy and reward have different units. Further ties use the average within-task rank of mean violation, then smaller hyperparameter values.

Table 3: Baseline search spaces, identical on every seed. Swept step sizes and gains use $\{1, 1.6, 2.5, 4, 6.3\} \times 10^k$ within each interval, including both endpoints. For PI control, the count combines the two sweeps described above, counting any overlap once.

Method	Step size η	Gain K_p	Configurations
Rate, Multi-group, Feasible Learning, Sparsity and Safe DPO			
GA	$[10^{-4}, 10^{1}]$	—	26
PI control	$[10^{-4}, 10^{1}]$	$[10^{-2}, 10^{2}]$	46
ADAM	$[10^{-5}, 1]$	—	26
RMSPROP	$[10^{-5}, 1]$	—	26
RMS-ALM	$[10^{-5}, 1]$	—	26
Safe RL			
GA	$[0.0025, 2.5]$	—	16
PI control	$[0.0025, 0.63]$	$[10^{-2}, 10^{2}]$	34
ADAM	$[0.0025, 0.63]$	—	13
RMSPROP	$[0.0025, 0.63]$	—	13
RMS-ALM	$[0.0025, 0.63]$	—	13

B.3 RATE-CONSTRAINED CLASSIFICATION

Predicting whether annual income exceeds \$50,000 on ADULT (Kohavi, 1996) under a recall floor, an instance of the rate constraints of Cotter et al. (2019):[8]

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}\big[\ell_{\mathrm{CE}}(\boldsymbol{\theta})\big] \quad \text{s.t.} \quad \tau - \mathrm{TPR}(\boldsymbol{\theta}) \le 0, \tag{56}$$

with TPR the true-positive rate and $\tau = 0.90$. Since recall is not differentiable, we use the proxy arrangement of Cotter et al. (2019): the multiplier is updated on true recall, while the primal replaces recall with a differentiable surrogate: the mean sigmoid output over positive training examples,

$$\widetilde{\mathrm{TPR}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} \sigma\big(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\big), \tag{57}$$

with $\mathcal{P}$ the positive training examples, $f_{\boldsymbol{\theta}}$ the network's logit and σ the logistic function.

Every primal and dual update uses the full training set. Under the 1% tolerance, a run is considered feasible when its final training recall is at least 0.891.

ADULT (OpenML version, Vanschoren et al., 2014) has $48{,}842$ examples, which we split $80/20$ into train and test once, identically for every seed. We drop the `sex` attribute, standardize the numeric features, and one-hot encode the categorical ones, leaving 96 features.

Table 4: Rate-constrained classification.

Architecture	MLP ($96 \times 64 \times 1$), ReLU
Optimizer	ADAM
Step size η_θ	10^{-3}
Batch size B	39,074 (full batch)
Epochs	24,440
Multiplier init λ_0	1
Recall floor τ	0.90

B.4 MULTI-GROUP RATE CONSTRAINTS

The ADULT income classification problem of Appendix B.3, with one recall floor per race group, in the group-rate framework of Agarwal et al. (2018):[9]

$$\min_{\boldsymbol{\theta}} \ \mathbb{E}\big[\ell_{\mathrm{CE}}(\boldsymbol{\theta})\big] \quad \text{s.t.} \quad \tau - \mathrm{TPR}_k(\boldsymbol{\theta}) \le 0, \qquad k = 1, \dots, 5, \tag{58}$$

[8] Their Adult experiment requires recall in each of four groups to be at least 95% of overall recall; we require overall recall of at least 90%. They use a linear classifier with hinge loss; we use an MLP with cross-entropy.

[9] Their fairness constraints limit differences in prediction rates between groups; we require recall of at least 70% in every group.

with $\tau = 0.70$ and k indexing all five race groups: White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo and Other. The multipliers are updated on the groups' true recalls; the primal uses the differentiable surrogate of Eq. (57) on each group's positive examples.

Every primal and dual update uses the full training set, observing all five constraints. Under the 1% tolerance, a run is considered feasible when its final training recall is at least 0.693 in every group.

We retain all five groups and use the split and preprocessing of Appendix B.3. We exclude both `race` and `sex` from the classifier's inputs, leaving 92 features.

Table 5: Multi-group rate constraints.

Architecture	MLP ($92 \times 64 \times 1$), ReLU
Optimizer	ADAM
Step size η_θ	10^{-3}
Batch size B	39,074 (full batch)
Epochs	6,000
Multiplier init λ_0	1
Recall floor τ	0.70

B.5 FEASIBLE LEARNING

The per-example loss-bound problem of Ramirez et al. (2025b)[10] on CIFAR-10 (Krizhevsky, 2009):

$$\min_{\boldsymbol{\theta}} \ \tfrac{\omega}{2}\|\boldsymbol{\theta}\|^2 \quad \text{s.t.} \quad \ell_i(\boldsymbol{\theta}) \le \epsilon, \qquad i = 1, \dots, N, \tag{59}$$

with ℓ_i the cross-entropy on training example i, ω the weight decay, $N = 50{,}000$ and $\epsilon = 0.3$.

We project every method's dual iterates onto $[0, \gamma]$, with $\gamma = 1/B$ and batch size B, and apply the same bound to extrapolated multipliers passed to the primal.[11] For ADAMULT, we cap the state so that the multiplier returned after RMS normalization is at most γ: using the current scale a_t, the additive accumulator is capped at γa_t, and the mirror state at $\log(\gamma a_t)$.[12] Each minibatch supplies the constraints of its sampled examples. If all sampled multipliers are zero, we skip the primal SGD step, including momentum and weight decay.

We train with random crops and horizontal flips. We consider a model feasible if at most 1% of the un-augmented training examples (500 of $50{,}000$) exceed the loss bound. This counts violated constraints, not the size of their violations.

Table 6: Feasible Learning.

Architecture	RESNET-18 (He et al., 2016)
Optimizer	SGD
Momentum	0.9
Step size η_θ	0.1
Scheduler	cosine decay
Weight decay	5×10^{-4}
Batch size B	128
Epochs	200
Multiplier bound γ	$1/B$
Multiplier init λ_0	$1/B$
Loss bound ϵ	0.3

[10] They use $\epsilon = 0.51$ and unbounded multipliers initialized at 0; we use $\epsilon = 0.3$ and the multiplier bound of Boero et al. (2026).

[11] This corresponds to the ℓ_1-resilient relaxation of Boero et al. (2026): each constraint admits a slack $u_i \ge 0$, penalized by $\gamma \sum_i u_i$.

[12] The mirror log-state interval shifts down with a tighter task cap, preserving its width.

B.6 CONTROLLED SPARSITY

The density-constrained problem of Gallego-Posada et al. (2022)[13] on CIFAR-10, with a hard-concrete stochastic gate (Louizos et al., 2018) per model weight. The gate $m_i \in [0, 1]$ multiplies the model weight θ_i, is redrawn at every training forward pass, and is exactly 0 with positive probability:

$$\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathbb{E}_{\boldsymbol{m}|\boldsymbol{\phi}}\big[\ell_{\mathrm{CE}}(\boldsymbol{\theta} \odot \boldsymbol{m})\big] \quad \text{s.t.} \quad d_k(\boldsymbol{\phi}) = \tau, \qquad k = 1, \ldots, 21, \tag{60}$$

with $\boldsymbol{\phi}$ the gate parameters, $\odot$ the elementwise product and $d_k(\boldsymbol{\phi})$ the expected fraction of active weights in layer k. We constrain all 21 layers (the convolutions and classifier) to density $\tau = 0.30$. The density is closed-form and differentiable in $\boldsymbol{\phi}$, so the constraint, unlike the objective, is deterministic. At evaluation we use their deterministic medians, as in Gallego-Posada et al. (2022).

We use their unstructured training hyperparameters (Appendix L), with random crops and horizontal flips. A run is considered feasible when every layer's expected density is within 0.003 of τ, the 1% tolerance.

Table 7: Controlled sparsity.

Architecture	RESNET-18
Optimizer, weights $\boldsymbol{\theta}$	SGD
Momentum	0.9
Optimizer, gates $\boldsymbol{\phi}$	ADAM
Step size η_θ	0.1
Step size η_ϕ	0.03
Scheduler, η_θ	$\times 0.1$ at epochs 30, 60, 90
Weight decay	5×10^{-4}
Batch size B	100
Epochs	120
Gate-logit initialization	$\mathcal{N}(\log 19, 0.01^2)$
Gate-logit bounds	$[\log 0.01, \log 100]$
Multiplier init μ_0	0 (RMS-ALM: 1)
Density target τ	0.30

B.7 SAFETY-CONSTRAINED DPO

LoRA fine-tuning (Hu et al., 2021) of ALPACA-7B for helpfulness under a safety preference constraint, with DPO training settings adapted from Wachi et al. (2024):[14]

$$\min_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{help}}_{\mathrm{DPO}}(\boldsymbol{\theta}) \quad \text{s.t.} \quad \tau - p_{\mathrm{safe}}(\boldsymbol{\theta}) \le 0, \tag{61}$$

with $\mathcal{L}^{\mathrm{help}}_{\mathrm{DPO}}$ the DPO loss (Rafailov et al., 2023) on helpfulness preferences and $\tau = 0.9$. The safety preference rate p_{safe} is the fraction of training pairs on which the policy's implicit reward ranks the safer response higher, averaged over the 32 pairs accumulated for one optimizer step. Since this rate is not differentiable, we use the proxy arrangement of Cotter et al. (2019): the multiplier is updated on the true constraint violation $\tau - p_{\mathrm{safe}}$, while the primal uses the safety DPO loss as a differentiable surrogate.[15]

We train on the 26,874 pairs of PKU-SAFERLHF-30K (Ji et al., 2023a), each labeled by helpfulness and safety.

After training, we draw 300 prompts from the test split of PKU-SAFERLHF-30K (2,989 examples), with a different draw per seed. The fine-tuned model generates one response per prompt using greedy

[13] They constrain from above, $d_k \le \tau$; we impose equality, which pushes an over-sparse layer back to the target.

[14] They train helpfulness and safety in sequence; we optimize them jointly. They fine-tune all weights; we train rank-8 LoRA adapters. They use length 512 and AdamW; we use length 1024, 8-bit AdamW and gradient clipping at norm 1.

[15] Except for RMS-ALM, accumulated primal gradients use the preceding window's multiplier; the dual then updates from the window's mean preference rate. RMS-ALM evaluates its post-primal residual on the next window, before training on it, following Eqs. (50) to (55).

decoding, with at most 128 new tokens. We score each generated response together with its prompt using the Beaver reward and cost models (Dai et al., 2024). We report mean reward as performance and consider a run feasible if its mean cost is at most zero. We allow no feasibility tolerance for Safe DPO, unlike the other tasks.[16]

Table 8: Safety-constrained DPO.

Model	`alpaca-7b-reproduced` (Dai et al., 2024; Taori et al., 2023)
LoRA rank / scaling α	8 / 16
LoRA dropout	0.05
LoRA modules	attention query and value projections
Max length	1024 tokens
Precision	bf16
DPO β	0.1
Optimizer	AdamW (Loshchilov and Hutter, 2019), 8-bit (Dettmers et al., 2022)
Step size η_θ	2×10^{-5}
Scheduler	cosine, 3% warmup
Weight decay	10^{-2}
Batch size B	32 (2×16, accumulated)
Gradient clipping	1.0
Epochs	3
Multiplier init λ_0	1
Safety floor τ	0.9

B.8 SAFE RL

On SAFETYHALFCHEETAHVELOCITY-V1 in SAFETY-GYMNASIUM (Ji et al., 2023b), we use PPO (Schulman et al., 2017) to optimize the Lagrangian, using OmniSafe v0.5.0 (Ji et al., 2024):

$$\max_{\pi} J_R(\pi) \quad \text{s.t.} \quad J_C(\pi) \le d, \tag{62}$$

with J_R and J_C the undiscounted episodic return and cost, cost limit $d = 25$ and horizon $T = 1000$. At each epoch, we record the mean episodic return and cost over the last 50 episodes. We update the multiplier once using $(\widehat{J}_C - d)/T$, where $\widehat{J}_C$ is this mean cost.[17]

Performance is the average of the last 10 recorded mean returns. We consider a run feasible when the average of the last 10 recorded mean costs is at most 25.25, allowing the 1% tolerance. Its baseline grids are in Table 3.

Table 9: Safe RL.

Horizon T	1000
Environment steps	10^7
Steps per epoch	20,000
Cost limit d	25
Multiplier init λ_0	1
Architecture	64×64 tanh MLPs (actor and critics)
Optimizer	ADAM
Step size η_θ	3×10^{-4}
Scheduler	linear decay (actor)
PPO (Schulman et al., 2017) iterations per epoch	40
Minibatch	64
Clip	0.2
Discount	0.99
GAE	0.95
Target KL	0.02

[16] The appendix tables still normalize mean safety cost by dividing it by the base model's mean cost, 6.34498.

[17] RMS-ALM's PPO steps optimize the augmented Lagrangian; its dual update uses the next epoch's cost estimate. PPO clipping applies to the reward term alone for RMS-ALM, and to the combined reward–cost term for the other methods.

C Complete Results per Task

C.1 Aggregation

Tables 1 and 2 weight the six tasks equally. For method m on task t, normalized score difference is

$$\Delta_{m,t} = 100\frac{\bar{s}_{m,t} - \bar{s}_{\mathrm{AdaMult},t}}{\bar{s}_{\mathrm{AdaMult},t} - b_t},$$

with each mean taken over all five seeds, including infeasible ones. For accuracy, $b_t = 0$. For Safe RL, $b_t = -664.73$ is the mean return of the initial rollout across five seeds, collected before any policy update. For Safe DPO, $b_t = 3.39138$ is the base Alpaca checkpoint's reward on the seed-0 evaluation prompts, before fine-tuning. Thus reward gaps are percentages of AdaMult's improvement over its starting policy; a common shift or positive rescaling of rewards leaves them unchanged. The worst task has the lowest feasible-seed fraction, with ties broken by lower relative score, then the task order in Table 10. Its feasibility and score refer to the same task.

Configurations are selected by Appendix B.2, using the same runs reported here. Test performance participates in selection. The tables report performance as mean $\pm$ population s.d., and normalized constraint residuals as mean / maximum across seeds; negative values indicate slack. Accuracy is in percent; feasibility and residual normalization follow §4.

C.2 DETAILED COMPARISONS

Table 10: **Per-task tuned baselines and fixed defaults.** Both ADAMULT variants use Eq. (5). Gray annotations: ADAMULT's feasible-seed count and performance minus that of the best selected baseline. Trials count configurations in the declared search grids; 0 denotes a fixed default. Filled dots mark seeds considered feasible; tinted rows have no feasible seed.

		Tuning		Outcome		
Task	Dual optimizer	Trials	Step size	Residual (%)	Feasible*	Test performance (↑)
Rate *all feasible, −0.1 acc.*	GA	26	0.63	0.02/0.29	●●●●●	81.6 ± 0.2
	PI control	46	0.0025	−0.16/0.18	●●●●●	81.6 ± 0.3
	ADAM	26	4×10^{-5}	0.09/0.40	●●●●●	81.6 ± 0.2
	RMSPROP	26	0.025	−0.11/0.87	●●●●●	81.7 ± 0.3
	RMS-ALM	26	0.01	−0.05/0.98	●●●●●	81.7 ± 0.1
	RMS-ALM default	0	Default	−0.05/0.98	●●●●●	81.7 ± 0.1
	Additive ADAMULT	0	Default	−0.29/0.04	●●●●●	80.7 ± 0.2
	ADAMULT	0	Default	−0.02/0.22	●●●●●	81.6 ± 0.2
Multi-group *5/5 feasible, −0.4 acc.*	GA	26	1.6	−2.69/0.71	●●●●●	85.1 ± 0.3
	PI control	46	10	−0.93/0.78	●●●●●	85.0 ± 0.1
	ADAM	26	1	−6.23/ − 1.76	●●●●●	84.6 ± 0.5
	RMSPROP	26	0.16	−4.37/ − 2.04	●●●●●	85.0 ± 0.1
	RMS-ALM	26	0.4	−0.88/0.45	●●●●●	85.1 ± 0.1
	RMS-ALM default	0	Default	−3.17/3.06	●●●●○	84.8 ± 0.2
	Additive ADAMULT	0	Default	−1.01/0.07	●●●●●	85.0 ± 0.1
	ADAMULT	0	Default	0.54/0.87	●●●●●	84.7 ± 0.2
Feasible Learning *all feasible, −0.3 acc.*	GA	26	10^{-4}	0.00/0.00	●●●●●	95.3 ± 0.1
	PI control	46	10^{-4}	0.00/0.00	●●●●●	95.1 ± 0.1
	ADAM	26	10^{-5}	0.00/0.00	●●●●●	95.2 ± 0.1
	RMSPROP	26	0.001	0.04/0.06	●●●●●	95.3 ± 0.1
	RMS-ALM	26	6.3×10^{-5}	0.00/0.00	●●●●●	95.3 ± 0.1
	RMS-ALM default	0	Default	0.02/0.03	●●●●●	95.2 ± 0.1
	Additive ADAMULT	0	Default	0.01/0.01	●●●●●	95.0 ± 0.2
	ADAMULT	0	Default	0.01/0.01	●●●●●	95.0 ± 0.1
Sparsity *5/5 feasible, +0.0 acc.*	GA	26	4×10^{-4}	61.34/89.01	○○○○○	92.5 ± 0.3
	PI control	46	0.1	0.01/0.02	●●●●●	91.9 ± 0.1
	ADAM	26	10^{-5}	28.43/36.88	○○○○○	92.3 ± 0.4
	RMSPROP	26	1	0.33/0.51	●●●●●	91.2 ± 0.4
	RMS-ALM	26	1	0.70/0.85	●●●●●	91.6 ± 0.1
	RMS-ALM default	0	Default	4.84/5.31	○○○○○	92.1 ± 0.2
	Additive ADAMULT	0	Default	0.04/0.13	●●●●●	90.5 ± 0.4
	ADAMULT	0	Default	0.00/0.00	●●●●●	91.9 ± 0.4
Safe RL *5/5 feasible, −59 ret.*	GA	16	0.63	−14.76/ − 1.44	●●●●●	2896 ± 42
	PI control	34	0.035	−26.11/ − 11.00	●●●●●	2900 ± 9
	ADAM	13	0.016	−13.75/0.22	●●●●●	2903 ± 21
	RMSPROP	13	0.63	−7.00/ − 5.63	●●●●●	2905 ± 29
	RMS-ALM	13	0.4	−6.42/ − 1.66	●●●●●	2886 ± 39
	RMS-ALM default	0	Default	5.95/16.38	●●○○○	2913 ± 14
	Additive ADAMULT	0	Default	−9.39/ − 0.10	●●●●●	2629 ± 337
	ADAMULT	0	Default	−66.32/ − 60.36	●●●●●	2846 ± 47
Safe DPO (7B) *5/5 feasible, +0.14 rew.*	GA	26	0.04	−10.63/ − 3.59	●●●●●	5.84 ± 0.51
	PI control	46	0.063	−35.42/ − 25.72	●●●●●	5.21 ± 0.59
	ADAM	26	0.025	−20.42/ − 10.77	●●●●●	5.65 ± 0.46
	RMSPROP	26	0.1	−20.73/ − 6.02	●●●●●	6.47 ± 0.51
	RMS-ALM	26	0.1	−24.12/ − 10.80	●●●●●	6.22 ± 0.41
	RMS-ALM default	0	Default	27.96/38.67	○○○○○	6.94 ± 0.31
	Additive ADAMULT	0	Default	−26.36/ − 19.24	●●●●●	6.27 ± 0.55
	ADAMULT	0	Default	−20.64/ − 8.87	●●●●●	6.61 ± 0.31

Table 11: **Baselines tuned across all six tasks.** Fixed defaults are unchanged from Table 10.

Task	Method	Shared configuration	Residual (%)	Feasible	Performance
Rate	GA	0.04	$-0.10/0.24$	$5/5$	81.46 ± 0.25
	PI control	0.1, $K_p = 1$	$-0.07/0.25$	$5/5$	81.29 ± 0.06
	ADAM	0.1	$-0.21/0.00$	$5/5$	81.40 ± 0.26
	RMSPROP	0.63	$-1.26/-0.05$	$5/5$	80.75 ± 0.65
	RMS-ALM	0.63	$-0.51/0.79$	$5/5$	81.07 ± 0.84
	RMS-ALM default	Default	$-0.05/0.98$	$5/5$	81.68 ± 0.08
	Additive ADAMULT	Default	$-0.29/0.04$	$5/5$	80.69 ± 0.25
	ADAMULT	Default	$-0.02/0.22$	$5/5$	81.62 ± 0.18
Multi-group	GA	0.04	$0.79/0.87$	$5/5$	84.37 ± 0.15
	PI control	0.1, $K_p = 1$	$-0.41/0.13$	$5/5$	84.60 ± 0.19
	ADAM	0.1	$-11.07/0.81$	$5/5$	84.12 ± 0.62
	RMSPROP	0.63	$-4.15/-1.26$	$5/5$	84.83 ± 0.19
	RMS-ALM	0.63	$-0.90/1.10$	$4/5$	85.10 ± 0.11
	RMS-ALM default	Default	$-3.17/3.06$	$4/5$	84.82 ± 0.19
	Additive ADAMULT	Default	$-1.01/0.07$	$5/5$	85.03 ± 0.12
	ADAMULT	Default	$0.54/0.87$	$5/5$	84.68 ± 0.16
Feasible Learning	GA	0.04	$0.00/0.01$	$5/5$	95.04 ± 0.08
	PI control	0.1, $K_p = 1$	$0.15/0.19$	$5/5$	94.21 ± 0.17
	ADAM	0.1	$2.22/2.40$	$0/5$	93.05 ± 0.12
	RMSPROP	0.63	$0.02/0.03$	$5/5$	95.18 ± 0.14
	RMS-ALM	0.63	$0.01/0.02$	$5/5$	95.17 ± 0.11
	RMS-ALM default	Default	$0.02/0.03$	$5/5$	95.19 ± 0.15
	Additive ADAMULT	Default	$0.01/0.01$	$5/5$	94.99 ± 0.20
	ADAMULT	Default	$0.01/0.01$	$5/5$	95.02 ± 0.06
Sparsity	GA	0.04	$197.65/201.76$	$0/5$	90.80 ± 0.48
	PI control	0.1, $K_p = 1$	$198.43/200.48$	$0/5$	89.76 ± 0.27
	ADAM	0.1	$200.40/204.68$	$0/5$	86.61 ± 0.42
	RMSPROP	0.63	$0.53/0.94$	$5/5$	90.69 ± 0.33
	RMS-ALM	0.63	$0.59/0.67$	$5/5$	90.58 ± 0.36
	RMS-ALM default	Default	$4.84/5.31$	$0/5$	92.11 ± 0.19
	Additive ADAMULT	Default	$0.04/0.13$	$5/5$	90.49 ± 0.43
	ADAMULT	Default	$0.00/0.00$	$5/5$	91.92 ± 0.38
Safe RL	GA	0.04	$6.87/235.10$	$4/5$	2815 ± 96
	PI control	0.1, $K_p = 1$	$-43.25/-8.52$	$5/5$	2682 ± 347
	ADAM	0.1	$-53.08/-19.49$	$5/5$	2863 ± 50
	RMSPROP	0.63	$-7.00/-5.63$	$5/5$	2905 ± 29
	RMS-ALM	0.63	$-11.31/-7.37$	$5/5$	2850 ± 69
	RMS-ALM default	Default	$5.95/16.38$	$2/5$	2913 ± 14
	Additive ADAMULT	Default	$-9.39/-0.10$	$5/5$	2629 ± 337
	ADAMULT	Default	$-66.32/-60.36$	$5/5$	2846 ± 47
Safe DPO	GA	0.04	$-10.63/-3.59$	$5/5$	5.84 ± 0.51
	PI control	0.1, $K_p = 1$	$-56.31/-42.52$	$5/5$	4.69 ± 0.60
	ADAM	0.1	$-84.52/-69.11$	$5/5$	4.36 ± 0.45
	RMSPROP	0.63	$-63.56/-46.10$	$5/5$	5.38 ± 0.56
	RMS-ALM	0.63	$-68.74/-57.18$	$5/5$	5.21 ± 0.27
	RMS-ALM default	Default	$27.96/38.67$	$0/5$	6.94 ± 0.31
	Additive ADAMULT	Default	$-26.36/-19.24$	$5/5$	6.27 ± 0.55
	ADAMULT	Default	$-20.64/-8.87$	$5/5$	6.61 ± 0.31

Table 12: **Ingredient removals at a single step size selected across all six tasks.**

Task	Method	Shared configuration	Residual (%)	Feasible	Performance
Rate	− normalization	0.01	$-0.00/0.21$	$5/5$	81.44 ± 0.25
	− multiplicative	0.01	$-0.29/0.04$	$5/5$	80.69 ± 0.25
	− extrapolation	0.01	$-0.10/0.66$	$5/5$	81.51 ± 0.31
	AdaMult	Default	$-0.02/0.22$	$5/5$	81.62 ± 0.18
Multi-group	− normalization	0.01	$-1.99/0.87$	$5/5$	84.10 ± 0.13
	− multiplicative	0.01	$-1.01/0.07$	$5/5$	85.03 ± 0.12
	− extrapolation	0.01	$0.49/0.87$	$5/5$	84.64 ± 0.18
	AdaMult	Default	$0.54/0.87$	$5/5$	84.68 ± 0.16
Feasible Learning	− normalization	0.01	$0.00/0.00$	$5/5$	95.22 ± 0.13
	− multiplicative	0.01	$0.01/0.01$	$5/5$	94.99 ± 0.20
	− extrapolation	0.01	$0.00/0.00$	$5/5$	95.11 ± 0.13
	AdaMult	Default	$0.01/0.01$	$5/5$	95.02 ± 0.06
Sparsity	− normalization	0.01	$29.41/32.30$	$0/5$	92.00 ± 0.30
	− multiplicative	0.01	$0.04/0.13$	$5/5$	90.49 ± 0.43
	− extrapolation	0.01	$179.40/179.90$	$0/5$	84.64 ± 2.48
	AdaMult	Default	$0.00/0.00$	$5/5$	91.92 ± 0.38
Safe RL	− normalization	0.01	$-82.08/-75.57$	$5/5$	2854 ± 43
	− multiplicative	0.01	$-9.39/-0.10$	$5/5$	2629 ± 337
	− extrapolation	0.01	$-60.98/-44.93$	$5/5$	2706 ± 354
	AdaMult	Default	$-66.32/-60.36$	$5/5$	2846 ± 47
Safe DPO	− normalization	0.01	$3.37/9.58$	$2/5$	6.17 ± 0.32
	− multiplicative	0.01	$-26.36/-19.24$	$5/5$	6.27 ± 0.55
	− extrapolation	0.01	$-24.21/-17.27$	$5/5$	6.13 ± 0.37
	AdaMult	Default	$-20.64/-8.87$	$5/5$	6.61 ± 0.31

C.3 RATE-CONSTRAINED CLASSIFICATION

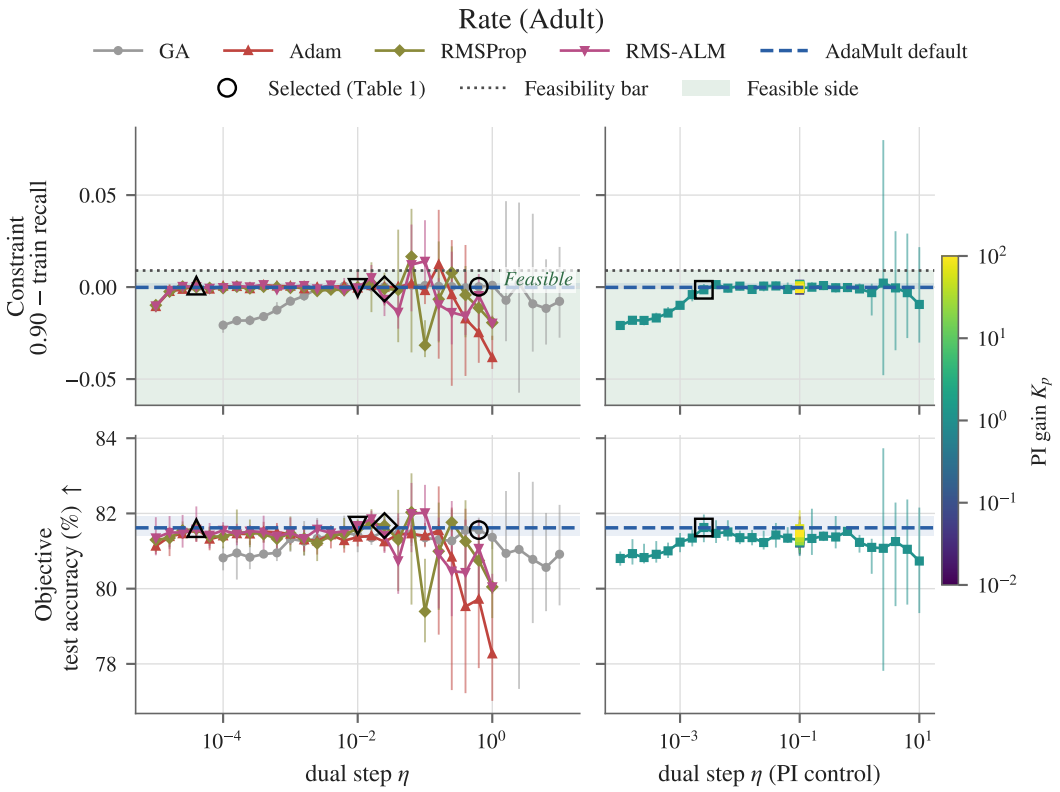

Figure 2: Rate: signed $0.90-$ training recall (top; cutoff 0.009) and test accuracy (bottom). Left: GA, ADAM, RMSPROP and RMS-ALM; right: PI control, colored by gain. Points and bars show five-seed means and ranges. Rings mark per-task selections; dashed lines and bands show ADAMULT's default; dotted lines bound the shaded feasible region. Subsequent baseline sweeps use these conventions. Setup in Appendix B.3.

C.4 MULTI-GROUP RATE CONSTRAINTS

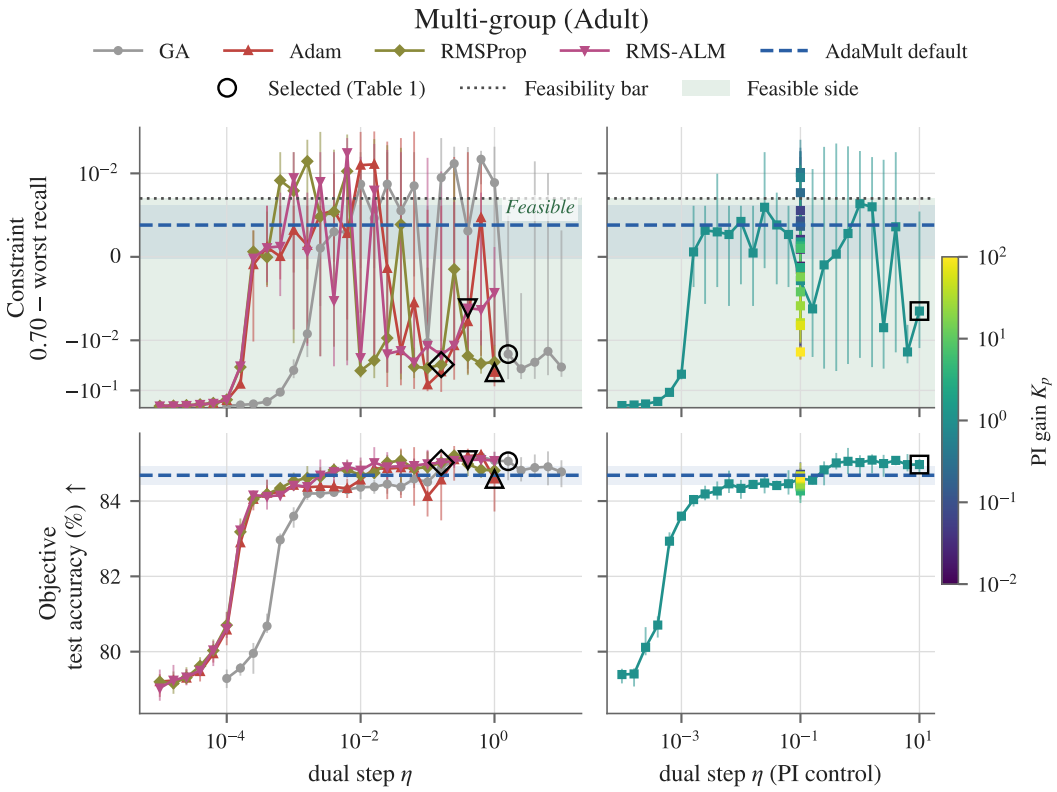

Figure 3: Multi-group: signed 0.70− worst-group training recall (top; cutoff 0.007) and test accuracy (bottom). Setup in Appendix B.4.

Table 13: Held-out Multi-group results for the selections in Table 10. Recall is the minimum across groups within each seed; means and population s.d.s use five training seeds on one data split.

Method	Test accuracy (%)	Worst-group test recall (%)	Seeds $\geq 70\%$
GA	85.06 ± 0.28	6.67 ± 8.16	0/5
PI control	84.97 ± 0.13	13.33 ± 6.67	0/5
ADAM	84.59 ± 0.48	13.33 ± 12.47	0/5
RMSPROP	85.02 ± 0.05	13.33 ± 6.67	0/5
RMS-ALM	85.09 ± 0.12	13.33 ± 6.67	0/5
RMS-ALM default	84.82 ± 0.19	13.33 ± 6.67	0/5
Additive ADAMULT default	85.03 ± 0.12	13.33 ± 6.67	0/5
ADAMULT	84.68 ± 0.16	10.00 ± 8.16	0/5

C.5 FEASIBLE LEARNING

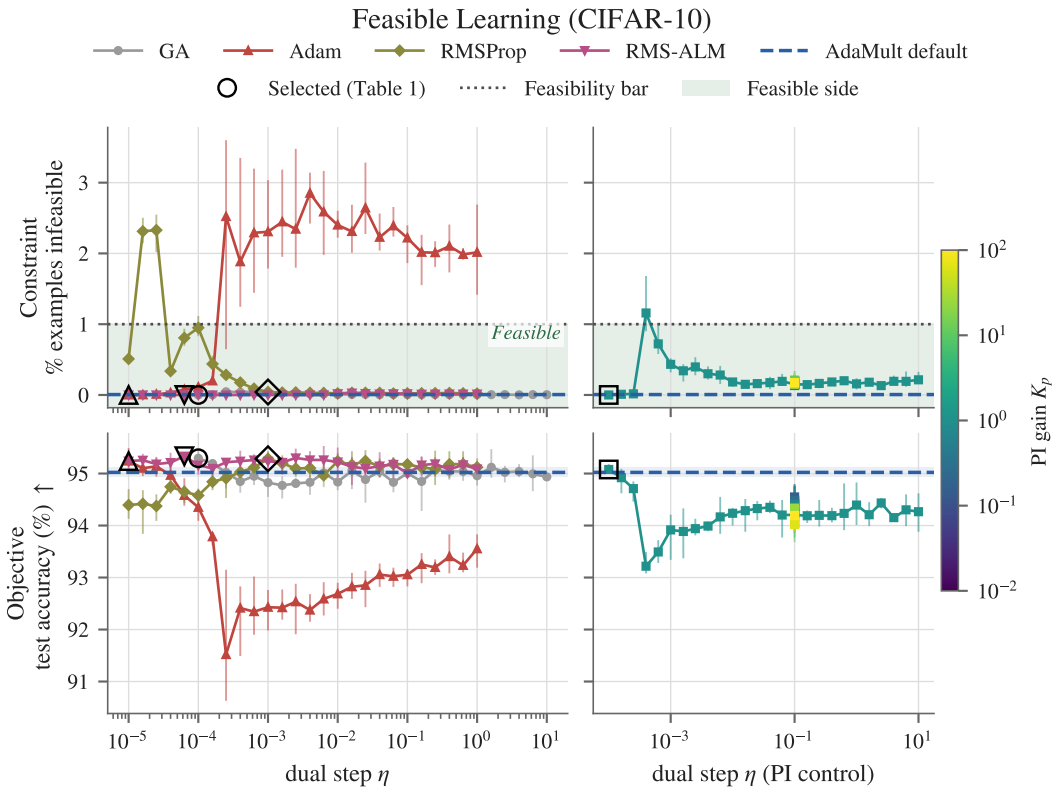

Figure 4: Feasible Learning: percentage of training examples above the loss bound (top; cutoff 1%) and test accuracy (bottom). Setup in Appendix B.5.

C.6 CONTROLLED SPARSITY

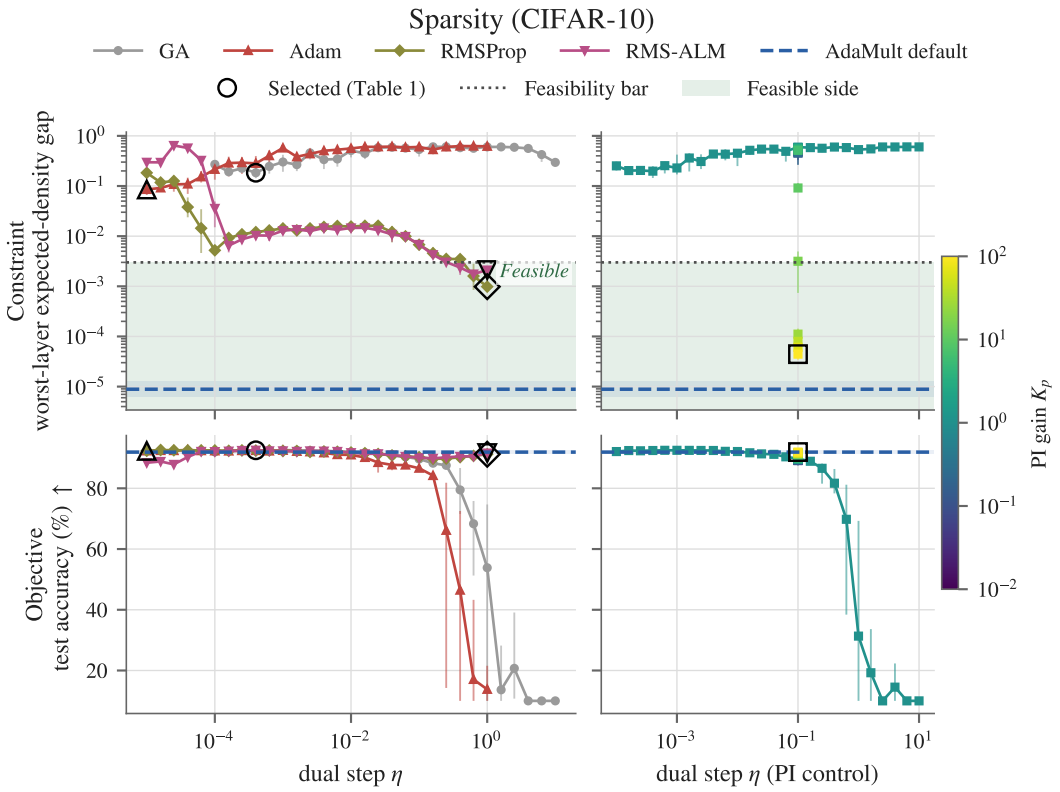

Figure 5: Sparsity: largest absolute layer-wise expected-density error (top; cutoff 0.003) and test accuracy (bottom). Setup in Appendix B.6.

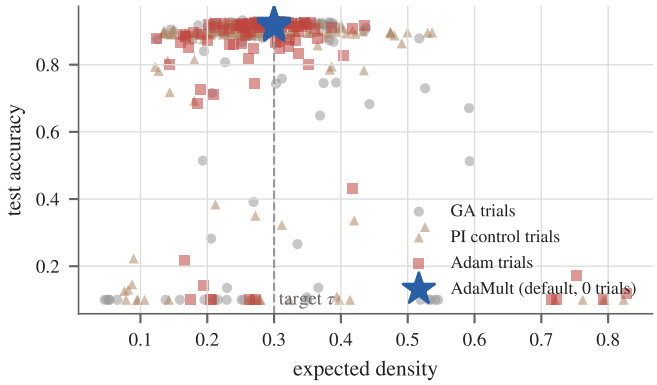

Figure 6: Test accuracy versus overall expected density for the GA, PI and ADAM grids and ADAMULT's default (five seeds each). The vertical line marks $\tau = 0.3$; overall density does not measure per-layer feasibility.

C.7 SAFETY-CONSTRAINED DPO

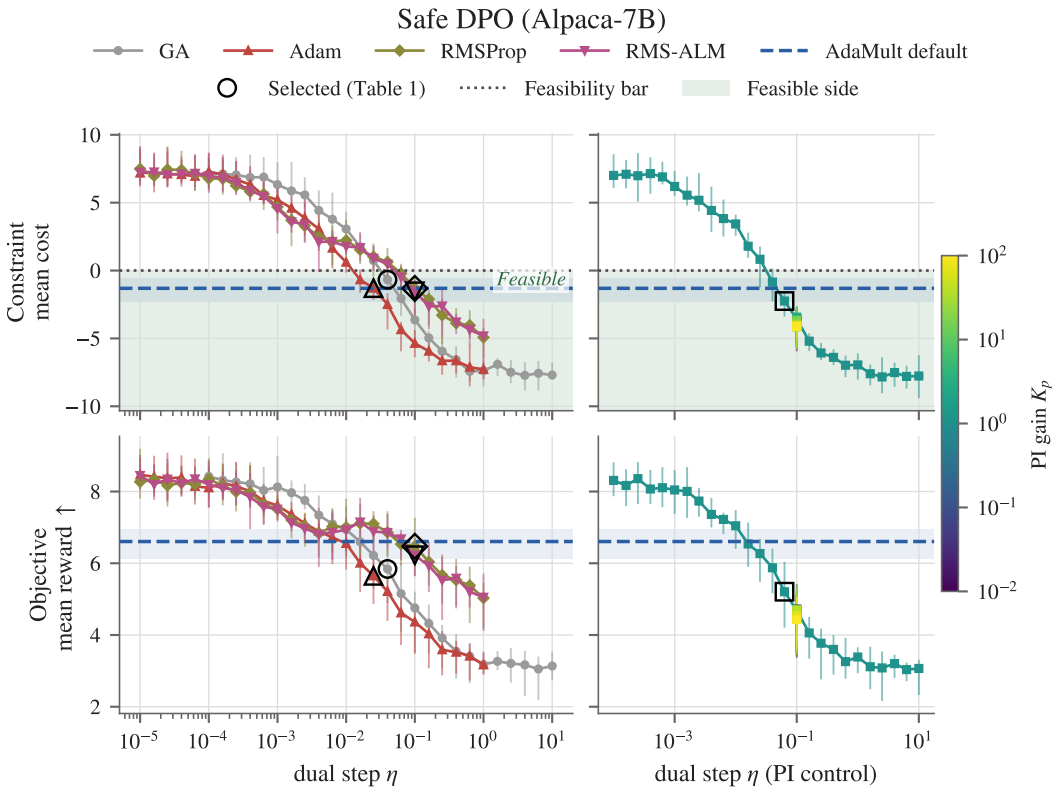

Figure 7: Safe DPO: evaluated mean cost (top; cutoff zero) and mean reward (bottom). Setup in Appendix B.7.

C.8 SAFE RL

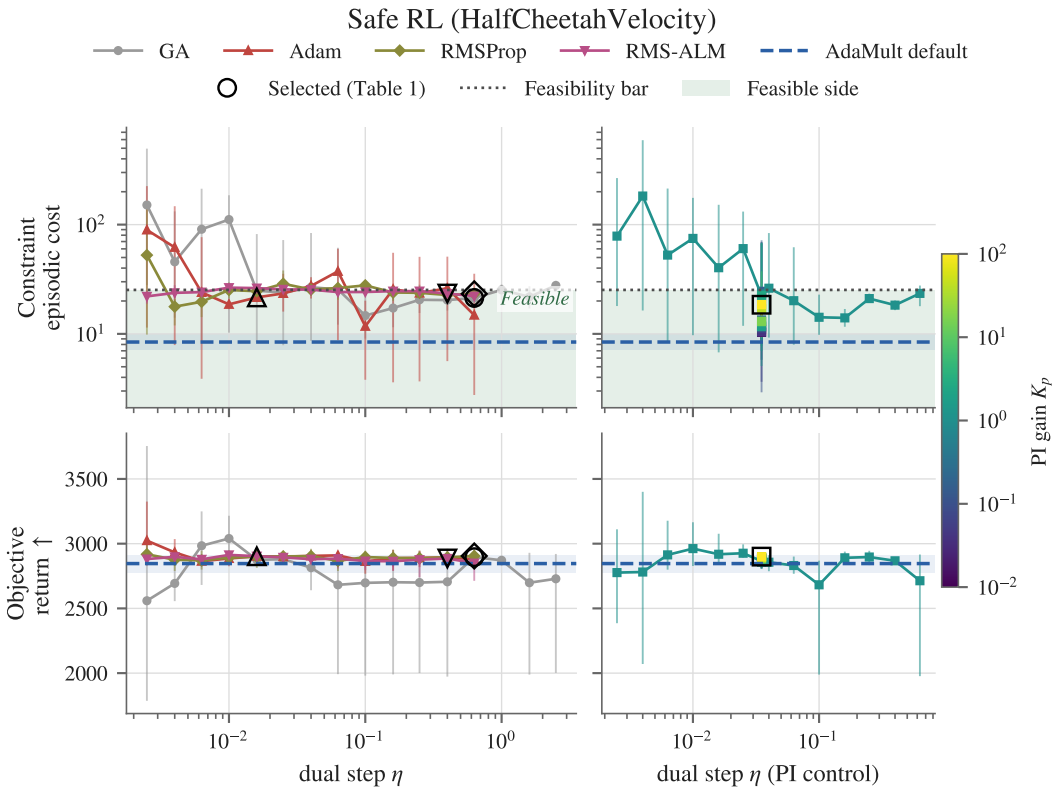

Figure 8: Safe RL: episodic cost (top; cutoff 25.25) and return (bottom) on HALFCHEETAHVELOCITY. Setup and averaging in Appendix B.8.

C.9 INGREDIENT SWEEPS

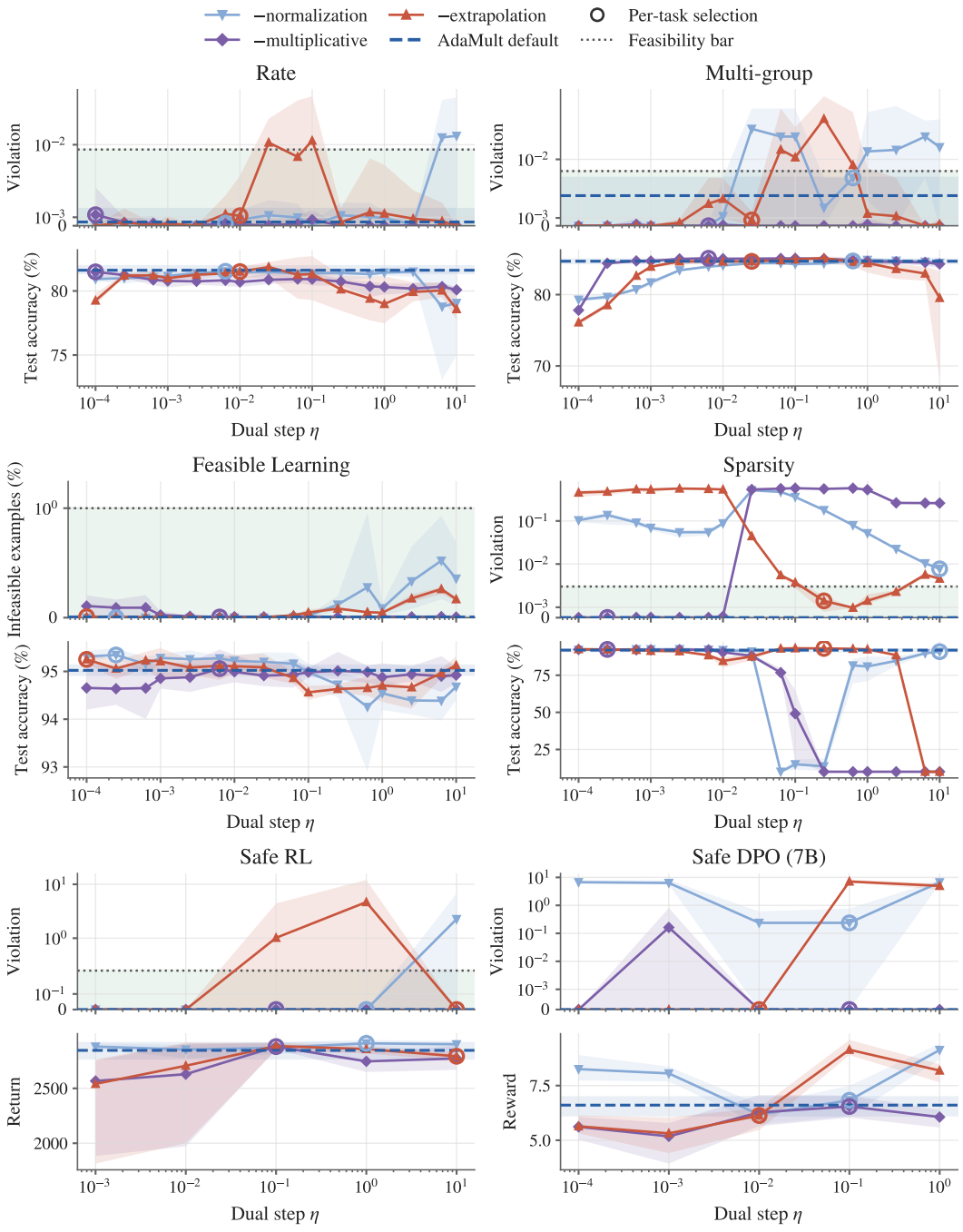

Figure 9: Full step-size sweeps for the ingredient removals in Table 2: violation above, performance below. Points and bands show five-seed means and ranges; rings mark per-task selections. Dashed lines and bands show AdaMult's default; dotted lines mark feasibility cutoffs. Violation axes are logarithmic away from zero and linear near zero.