CLIPPED DIAGONAL NATURAL POLICY GRADIENT

Anonymous authors
Paper under double-blind review

ABSTRACT

Natural policy gradient (NPG) methods account for the local geometry of the policy distribution, but computing directions preconditioned by the Fisher information matrix is expensive. Approximating this matrix with its diagonal yields a computationally efficient NPG algorithm, but such an algorithm has shown weak and sometimes unstable empirical performance in prior studies. We show that inverse-diagonal scaling can create nonlocal parameter displacements, while diagonal normalization can misestimate the update's length under the full Fisher metric. We introduce Clipped Diagonal NPG (CD-NPG). It clips each coordinate of the inverse-diagonal direction at a multiple of the median coordinate magnitude within its layer. It then uses the current minibatch to estimate the clipped direction's full-Fisher quadratic without forming the matrix. The method uses this estimate to rescale the direction to a target local policy change. These modifications preserve the asymptotic computational and memory complexities of diagonal NPG. Under matched hyperparameter-search budgets on six MuJoCo tasks, CD-NPG obtains returns comparable to or higher than policy gradients optimized with Adam or Muon on most tasks. Comparisons with two richer, more expensive NPG approximations are task- and training-setting-dependent. Diagnostics show substantially better local-KL calibration than diagonal normalization, while clipping ablations show that the clipped direction has fewer extreme finite-step departures from the local Fisher model than the unclipped direction.

1 INTRODUCTION

Policy-gradient (PG) methods use the gradient g from the policy-gradient theorem to seek improvements in the expected return of a parameterized policy (Sutton et al., 2000). Euclidean PG updates under different parameterizations of the same policy can induce drastically different local changes in its action distribution, affecting learning progress and stability. Natural policy gradient (NPG) methods reduce this dependence with the policy Fisher information matrix F, which measures local change in the action distribution (Amari, 1998; Kakade, 2001). The NPG direction $F^{-1}g$ gives the largest linearized improvement in expected return for a fixed amount of Fisher-measured policy change.

Obtaining $F^{-1}g$ at neural-network scale is NPG's main computational challenge. Canonical NPG approximations reduce this cost but retain substantial overhead. For example, matrix-free conjugate gradient (CG) retains the full Fisher implicitly and solves its system using repeated Fisher–vector products (Schulman et al., 2015). K-FAC instead retains structured within-layer correlations through Kronecker-factored blocks that must be constructed and factorized (Martens & Grosse, 2015; Wu et al., 2017).

The simplest nontrivial approximation retains only the diagonal of F, discarding all cross-coordinate structure. For $D = \mathrm{diag}(F)$, each D_{ii} measures the local sensitivity of the policy distribution to parameter θ_i. Diagonal NPG forms $q = D^{-1}g$ and scales it by $\sqrt{2\delta/(q^\top Dq)}$ to target a predicted local-KL radius δ under the diagonal model. This algorithm requires only $O(BP)$ work and $O(P)$ persistent optimizer state for batch size B and P policy parameters, without iterative linear solves or matrix factorizations. It therefore has the same asymptotic computational and persistent-memory complexities as the widely adopted Adam optimizer. However, prior reinforcement-learning studies have reported weak and sometimes unstable empirical performance for diagonal-NPG variants (Gebotys et al., 2022; Sun, 2026). These findings raise the central question of this work: can diagonal NPG be made effective and reliable without sacrificing its low cost?

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.

We identify two distinct vulnerabilities in the direct construction. First, small diagonal entries can amplify gradient-estimation noise into large coordinate displacements, causing an update to leave the region where the current Fisher is a reliable local model. Second, $q^\top Dq$ omits the cross-coordinate terms in $q^\top Fq$ and can therefore mismeasure the proposal's full-Fisher length.

To address these issues, we introduce two targeted modifications to diagonal NPG. First, we clip each coordinate of the diagonal-NPG proposal $D^{-1}g$ at a multiple of the median coordinate magnitude in its layer. This limits the nonlocal displacements caused by large-magnitude coordinate outliers in the proposal. Second, we use a matrix-free minibatch estimate of the current-policy full-Fisher quadratic along the clipped proposal to estimate its local-KL length and rescale it to a target radius. We call the resulting algorithm clipped diagonal natural policy gradient (CD-NPG). Coordinatewise clipping and JVP-based evaluation of directional Fisher quadratics have each been used in other settings. Our contribution is to use them to address the direction and step-length failures we identify in diagonal NPG. CD-NPG preserves diagonal NPG's asymptotic computational and persistent-memory complexities, albeit at a higher constant-factor cost. Appendix A reviews the relevant precedents.

We evaluate CD-NPG on six MuJoCo tasks under both on- and off-policy training. Under matched hyperparameter-search budgets, CD-NPG achieves returns comparable to or higher than those of Adam + PG and Muon + PG on most tasks in the on-policy setting, with clearer advantages in the off-policy setting. Comparisons with richer and more expensive NPG approximations are mixed and training-setting dependent. In particular, K-FAC-NPG tends to outperform CD-NPG under on-policy training, whereas CD-NPG tends to outperform K-FAC-NPG under off-policy training, with substantial task-to-task variation. Component ablations and policy-motion diagnostics further examine the failure modes and the roles of the two proposed modifications. In the controlled width-scaling benchmark, the complete CD-NPG update takes 1.28–1.43 times as long as Adam + PG across the tested widths. It uses the same two P-element persistent optimizer-state buffers as Adam, while its estimated peak explicit-tensor memory remains close to Adam's.

In summary, diagonal NPG can be made into a practical and computationally efficient policy optimizer after its direction and step-length issues are addressed.

2 Background

Policy-gradient methods optimize a parameterized policy $\pi_\theta(a \mid s)$ directly. For discount factor $\gamma \in [0, 1)$, we denote the normalized expected discounted return as $J(\theta) = (1-\gamma)\mathbb{E}_{\pi_\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ and the corresponding normalized discounted state-visitation distribution as $d^{\pi_\theta}(\cdot) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \Pr_{\pi_\theta}(s_t \in \cdot)$. Let $A^{\pi_\theta}(s, a)$ be the exact advantage under the current policy. The policy-gradient theorem (Sutton et al., 2000) gives

$$g = \nabla_\theta J(\theta) = \mathbb{E}_{\substack{s\sim d^{\pi_\theta} \\ a\sim\pi_\theta(\cdot|s)}} \left[A^{\pi_\theta}(s, a)\nabla_\theta \log \pi_\theta(a \mid s)\right]. \tag{1}$$

Policy-gradient methods use g to construct updates that seek to improve $J(\theta)$.

In this work, we use updates constructed by Adam (Kingma & Ba, 2015) and Muon (Jordan et al., 2024; Liu et al., 2025) as our baseline algorithms. Adam maintains moving averages of the objective gradient and its elementwise square; its scaling statistic therefore includes advantage magnitudes and minibatch aggregation. Muon approximately orthogonalizes matrix-valued momentum with Newton–Schulz iterations, defining geometry from the parameter update rather than the policy distribution. Following Jordan et al. (2024), Muon updates the hidden-to-hidden weight matrices, while auxiliary Adam updates the remaining actor parameters.

NPG instead uses the policy Fisher to connect a parameter update to the local change it induces in the policy distribution. For a sufficiently small update $\Delta\theta$, the KL from π_θ to $\pi_{\theta+\Delta\theta}$, averaged over $s \sim d^{\pi_\theta}$ with this state distribution held fixed, has the second-order expansion $\frac{1}{2}\Delta\theta^\top F(\theta)\Delta\theta + o(\|\Delta\theta\|^2)$, where

$$F(\theta) = \mathbb{E}_{\substack{s\sim d^{\pi_\theta} \\ a\sim\pi_\theta(\cdot|s)}} \left[\nabla_\theta \log \pi_\theta(a \mid s)\nabla_\theta \log \pi_\theta(a \mid s)^\top\right] \tag{2}$$

is the Fisher information matrix. Maximizing the linearized improvement under a local KL budget $\delta > 0$ gives

$$\max_{\Delta\theta}\ g^\top\Delta\theta \quad \text{s.t.} \quad \tfrac{1}{2}\Delta\theta^\top F\Delta\theta \le \delta. \tag{3}$$

Table 1: Dominant actor-update complexity for a dense MLP actor with batch size B, P parameters, L affine layers of maximum width d, and L_{M} width-d Muon matrices. Here $A_B = O(BLd)$ is the shared activation and autodiff workspace, and K_{CG} and S are CG and Newton–Schulz iteration counts. These bounds exploit dense affine-layer structure and are therefore architecture-specific.

Actor update	total actor-update work	persistent memory	peak memory
Adam + PG	$O(BP)$	$O(P)$	$O(A_B + P)$
Muon + PG	$O(BP + SL_{\mathrm{M}}d^3)$	$O(P)$	$O(A_B + P)$
CD-NPG	$O(BP)$	$O(P)$	$O(A_B + P)$
CG-NPG	$O(K_{\mathrm{CG}}BP)$	$O(P)$	$O(A_B + P)$
K-FAC-NPG	$O(BP + L(Bd^2 + d^3))$	$O(P + Ld^2)$	$O(A_B + P + Ld^2)$

For positive-definite F and nonzero g, the solution follows the natural policy-gradient direction $d = F^{-1}g$ (Amari, 1998; Kakade, 2001). Making the local KL constraint tight determines the step size α, giving the complete update

$$d = F^{-1}g, \qquad \alpha = \sqrt{\frac{2\delta}{d^\top F d}}, \qquad \Delta\theta = \alpha d. \tag{4}$$

Computing the full-Fisher direction is expensive. Conjugate gradient (CG) avoids quadratic storage by approximately solving $(F + \lambda_{\mathrm{CG}} I)d = g$ with Fisher–vector products; K_{CG} iterations cost $O(K_{\mathrm{CG}}BP)$ work and $O(P)$ solver workspace (Schulman et al., 2015). K-FAC instead approximates each layer block by a Kronecker product of second-moment factors for the layer inputs and back-propagated score errors. This retains structured within-layer correlations while omitting cross-layer blocks (Martens & Grosse, 2015; Wu et al., 2017). For a width-d layer, constructing the factors costs $O(Bd^2)$ and factoring their damped systems costs $O(d^3)$. Appendix D gives the complete baseline updates, and Table 1 summarizes the costs of all methods for dense MLP actors.

3 Clipped Diagonal Natural Policy Gradient

We first describe direct diagonal NPG and analyze its step-length and direction failures, then present our modifications.

3.1 Direct Diagonal NPG

A naive way to approximate NPG is to replace the Fisher information matrix F in equation 3 by its diagonal $D = \mathrm{diag}(F)$. The exact natural direction $F^{-1}g$ is then approximated by $D^{-1}g$, which divides each gradient coordinate by its corresponding Fisher-diagonal entry. At update t, we estimate the gradient and Fisher diagonal from a minibatch $\{(s_i, a_i, \widehat{A}_i)\}_{i=1}^B$, where $\widehat{A}_i$ is an advantage estimate for (s_i, a_i):

$$\widetilde{g}_t = \frac{1}{B}\sum_{i=1}^{B} \widehat{A}_i \nabla_\theta \log \pi_\theta(a_i \mid s_i), \qquad \widetilde{v}_t = \frac{1}{B}\sum_{i=1}^{B} \big[\nabla_\theta \log \pi_\theta(a_i \mid s_i)\big]^{\odot 2}. \tag{5}$$

Derivatives are evaluated at θ_{t-1} with the minibatch and advantages held fixed. Here $\odot$ denotes elementwise multiplication, and $x^{\odot 2} = x \odot x$. Advantage weights appear only in the gradient: curvature measures policy sensitivity, not reward magnitude. Each per-example score in equation 5 must be squared before averaging; squaring the batch-mean score does not give the same Fisher-diagonal estimate.

Both minibatch estimates are noisy, and directly dividing one noisy estimate by the other can be unstable, especially when a Fisher-diagonal estimate is small. We therefore smooth the gradient and Fisher diagonal separately before forming their ratio. Initialize $m_0 = v_0 = 0$ and maintain $m_t = \beta_1 m_{t-1} + (1-\beta_1)\widetilde{g}_t$ and $v_t = \beta_2 v_{t-1} + (1-\beta_2)\widetilde{v}_t$. The bias-corrected estimates $\widehat{m}_t = m_t/(1-\beta_1^t)$ and $\widehat{v}_t = v_t/(1-\beta_2^t)$ remove the initial shrinkage caused by the zero initialization. Let $D_t = \mathrm{diag}(\max(\widehat{v}_t, \epsilon_F))$, where $\epsilon_F > 0$ is a numerical floor. The resulting unclipped diagonal direction is

$$u_t = D_t^{-1}\widehat{m}_t. \tag{6}$$

Applying the KL-budget scaling in equation 4 with F replaced by D_t gives

$$\Delta\theta_t^{\mathrm{diag}} = \sqrt{\frac{2\delta}{\max(u_t^\top D_t u_t, \epsilon_Q)}}\, u_t, \tag{7}$$

where $\epsilon_Q > 0$ is a numerical floor. This is the direct diagonal approximation to the NPG trust-region update: D_t determines both the update direction and its predicted KL.

This construction is superficially similar to Adam because both maintain coordinatewise statistics, but their denominators estimate different quantities. Adam divides an exponential moving average (EMA) of the gradient by the square root of an EMA of squared objective gradients, which mixes reward or advantage magnitude with policy sensitivity. In contrast, the diagonal construction divides its gradient EMA by an EMA of unweighted squared policy scores, targeting the ideal population form $\mathrm{diag}(F)^{-1}g$, and uses a Fisher quadratic to set the global step length. Locally, if a scalar parameter affects the policy with sensitivity j, its gradient scales as j while its Fisher entry scales as j^2. Adam's normalization does not supply the inverse-j factor needed to cancel weak sensitivity, whereas a diagonal-natural direction does so.

These geometric differences do not establish that diagonal NPG generally outperforms Adam. Proposition 1 in Appendix B gives a narrower theoretical comparison between ideal diagonal NPG and sign stochastic gradient descent (sign-SGD) (Bernstein et al., 2019). Sign-SGD corresponds to Adam with $\beta_1 = \beta_2 = 0$ when its numerical stabilizer is omitted. Under matched exact current-policy KL and sufficiently small steps, diagonal NPG has larger local return when the benefit of preserving heterogeneous coordinate magnitudes exceeds its relative cost under the full Fisher; sign-SGD has larger local return when the reverse holds.

For a general layer ℓ with P_ℓ parameters, a direct batched implementation forms its $B \times P_\ell$ matrix of per-example score gradients and averages their elementwise squares. Processing layers sequentially avoids materializing the full $B \times P$ score matrix, but requires $O(B \max_\ell P_\ell)$ temporary storage. For fully connected layers, standard layer-wise contractions compute the same Fisher diagonal without materializing these per-example parameter gradients (Martens & Sutskever, 2012; Balles et al., 2017; Dangel et al., 2020). Appendix C.2 derives the fused score-backward computation used for our MLP actors. In either case, reducing the per-example score squares across all parameters requires $O(BP)$ work. The next question is whether the resulting diagonal supplies a reliable direction and step length.

3.2 Calibrating the local policy-space length

The Fisher diagonal can be a useful preconditioner yet badly miscalibrate the step length because its quadratic omits cross-coordinate terms. For example, consider a P-parameter Fisher matrix with unit diagonal and constant positive off-diagonal correlation $0 < \rho < 1$:

$$\begin{aligned} F = (1-\rho)I + \rho\mathbf{1}\mathbf{1}^\top, \qquad & D = \mathrm{diag}(F) = I, \qquad q = \mathbf{1}, \\ q^\top D q = P, \qquad & q^\top F q = P\big[1 + (P-1)\rho\big], \\ h_{\mathrm{diag}} = \sqrt{2\delta/P}\, q, \qquad & \tfrac{1}{2} h_{\mathrm{diag}}^\top F h_{\mathrm{diag}} = \big[1 + (P-1)\rho\big]\delta. \end{aligned} \tag{8}$$

Thus the diagonal quadratic underestimates the full quadratic, and the diagonal-normalized local KL overshoots its target by $1 + (P-1)\rho$. This factor grows linearly with P for fixed $\rho > 0$.

For an actual update, let $\overline{F}_t$ denote the Fisher that retains all parameter correlations, integrates actions under the current policy, and averages over the current minibatch states. Its quadratic along q is

$$\mathcal{Q}_t(q) = q^\top \overline{F}_t q = \frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{a\sim\pi_{\theta_{t-1}}(\cdot|s_i)}\left[\left(q^\top \nabla_\theta \log \pi_\theta(a \mid s_i)\right)^2\right], \tag{9}$$

where derivatives are evaluated at θ_{t-1}.

A constant mismatch could be absorbed into the nominal δ, but the quadratic-mismatch factor $\zeta_t(q) = \mathcal{Q}_t(q)/(q^\top D_t q)$ changes with the policy, minibatch, and direction. Tuning for its maximum makes typical steps too small; tuning for a typical value leaves occasional large overshoots.

The mismatch is also parameterization dependent because taking a matrix diagonal is coordinate dependent. In the eigenvector coordinates of the example above, the corresponding ratio is one,

whereas it is $1 + (P-1)\rho$ in the original coordinates. Consequently, the nominal δ needed for the same policy-space step can change substantially under reparameterization.

Calibrating the scale of a chosen direction q does not require solving a full-Fisher system. We only need the scalar $\mathcal{Q}_t(q)$, which can be evaluated using one output Jacobian–vector product (JVP) followed by an output-space Fisher contraction. This evaluation costs $O(BP)$ work and $O(A_B + P)$ peak memory and requires no additional persistent state. We then rescale q by $\sqrt{2\delta/\max(\mathcal{Q}_t(q), \epsilon_Q)}$. This is the standard quadratic scaling used by TRPO after its approximate conjugate-gradient solve (Schulman et al., 2015); here, the direction is instead chosen by the diagonal statistic, and we omit TRPO's subsequent line search.

3.3 Controlling extreme coordinate ratios

Calibrating the step length still leaves a separate direction failure. When a Fisher-diagonal estimate is very small, division by it can result in a large parameter displacement. One coordinate can then dominate the direction and leave the region where the current Fisher is an accurate local model.

A two-dimensional continuous-action bandit makes the failure concrete. Let the policy $\pi_\theta = \mathcal{N}(\mu(\theta), I_2)$, where $\mu(\theta) = (\theta_1, \varepsilon\theta_2 + \frac{h}{2}\theta_2^2)$ and $\theta_0 = (0, 0)$, and define the reward for action a as $r(a) = 1 - \frac{1}{2}[(a_1 - 1)^2 + a_2^2]$. Its expected reward is

$$J(\theta) := \mathbb{E}_{a\sim\pi_\theta}[r(a)] = -\tfrac{1}{2}\big[(\mu_1(\theta) - 1)^2 + \mu_2(\theta)^2\big]. \tag{10}$$

The leading constant 1 in $r(a)$ cancels the two fixed unit-variance terms in the expectation, so $J(\theta_0) = -\frac{1}{2}$, without affecting gradients. At θ_0, the second mean coordinate is only ε-sensitive. A small gradient error there nevertheless becomes an increasingly large displacement after inverse-diagonal scaling as $\varepsilon \to 0$. Even after full-Fisher normalization, the exact finite-step KL can greatly exceed its local prediction because the nonlinear sensitivity changes along that displacement.

The example motivates controlling anomalous coordinates before normalization but does not prescribe a unique remedy. We adopt a simple layer-wise clipping rule: because neural layers can have different update scales, we define a separate threshold for each layer. Let $\mathcal{I}_\ell$ index the weights and bias of layer ℓ. Throughout, when the number of values is even, median denotes the lower of the two middle values. We set

$$\tau_{t,\ell} = c_{\mathrm{rel}}\,\mathrm{median}\big(\{|u_{t,j}| : j \in \mathcal{I}_\ell\}\big), \qquad c_{\mathrm{rel}} \in [1, \infty). \tag{11}$$

For the no-clipping ablation only, $c_{\mathrm{rel}} = +\infty$ denotes the setting $d_t = u_t$ directly rather than evaluation of the threshold above. The clipped direction is

$$d_{t,j} = \mathrm{clip}(u_{t,j}, -\tau_{t,\ell}, \tau_{t,\ell}), \qquad j \in \mathcal{I}_\ell. \tag{12}$$

Exact medians across all layers can be computed in $O(P)$ work using linear-time selection (Blum et al., 1973), and applying the coordinatewise clipping also costs $O(P)$. Finally, full-Fisher normalization sets the step length:

$$Q_t^{\mathrm{int}} = \mathcal{Q}_t(d_t) = d_t^\top \overline{F}_t d_t, \qquad \Delta\theta_t = \sqrt{\frac{2\delta}{\max(Q_t^{\mathrm{int}}, \epsilon_Q)}}\, d_t, \qquad \theta_t = \theta_{t-1} + \Delta\theta_t. \tag{13}$$

Figure 1 illustrates how clipping addresses this failure. In the raw diagonal proposal, inverse-diagonal scaling amplifies a weakly policy-sensitive coordinate until it dominates the direction. Full-Fisher normalization rescales the direction but preserves its coordinate ratios. Unless δ is made extremely small, the resulting parameter displacement can leave the region in which the current Fisher is an accurate local model. Layer-wise clipping first changes the direction by capping the outlier relative to the median coordinate magnitude; full-Fisher normalization then calibrates the clipped direction's local policy-space length.

4 MuJoCo Evaluation

We evaluate CD-NPG on six continuous-control MuJoCo benchmarks: Swimmer-v4, Hopper-v4, Walker2d-v4, HalfCheetah-v4, Ant-v4, and Humanoid-v4, using EnvPool (Todorov et al., 2012; Weng

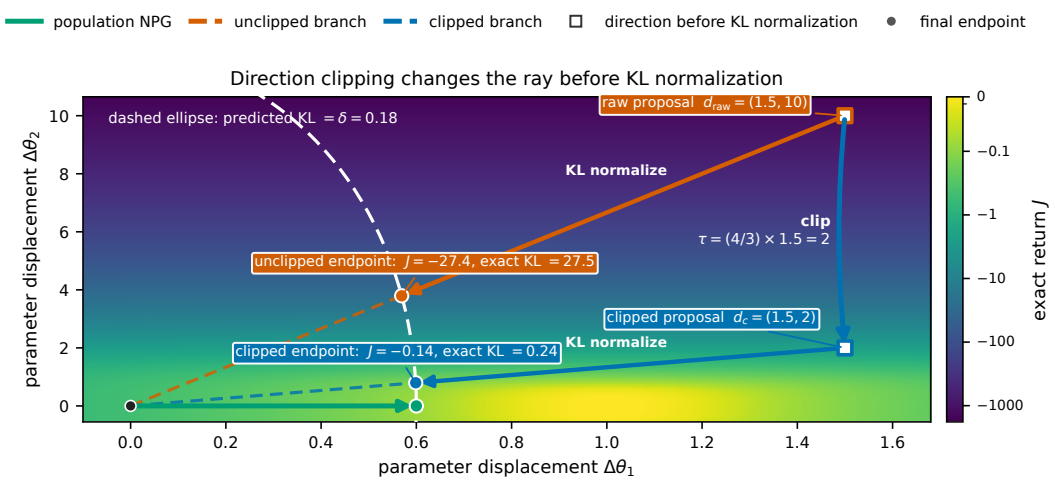

Figure 1: Clipping changes coordinate ratios before scalar KL normalization. With $h = 1$, $\varepsilon = 0.05$, $\xi = 0.5$, $\delta = 0.18$, and $c_{\text{rel}} = 4/3$, hollow squares show the raw directions, and filled points show their full-Fisher-normalized updates. The layer proposal is $(1.5, 10)$; under this convention, its median is 1.5, giving threshold 2. Starting from $J(\theta_0) = -0.5$, the unclipped diagonal update reaches $J = -27.4$ and exact KL 27.47, whereas clipping reaches $J = -0.14$ and exact KL 0.24. The green arrow is population NPG. Appendix E gives the complete construction and threshold calculation. All quantities are analytic; there are no sampled runs, selected configurations, or failed trials.

et al., 2022). The comparison methods are Adam + PG, Muon + PG, CG-NPG with 16 conjugate-gradient iterations, and K-FAC-NPG. The main-text figures focus on the primary comparison of CD-NPG with Adam + PG and Muon + PG. For readability, the setting-rank curves comparing CD-NPG with CG-NPG and K-FAC-NPG are reported in Appendix F.4; the main text summarizes this richer-metric comparison and refers to the appendix for the complete curves.

For all algorithms, we evaluate an on-policy version and an off-policy version. The on-policy version collects a fresh minibatch of 64 transitions and performs one actor update and one critic update. The off-policy version follows the PPO construction (Schulman et al., 2017): each method collects 2,048 transitions under its behavior policy and then performs $K = 16$ epochs of actor and critic optimization. At every actor update, all methods optimize the same objective with a clipped importance-sampling ratio, and each epoch reshuffles the rollout into 32 minibatches of 64. For Adam + PG, this off-policy construction gives PPO. Across methods, the rollout schedule, critic training, and advantage-estimation procedure are shared, although each method necessarily collects its own trajectories. Appendix F gives the complete algorithmic and training details.

All methods use the same architecture: separate actor and critic MLPs, each with two hidden layers of 64 ReLU units. We tune two method-specific hyperparameters for each algorithm: Adam + PG varies the actor learning rate and global gradient-clipping threshold; Muon + PG varies its matrix and auxiliary-Adam learning rates; CD-NPG varies the target local KL radius δ and layer-wise relative threshold c_{rel}; and CG-NPG and K-FAC-NPG each vary δ and their respective damping parameters. Every base grid contains $6 \times 6 = 36$ configurations. Thus the tuning budgets are matched in configuration count, although the method-specific search spaces are not equivalent.

Each configuration is trained for five million environment transitions with ten seeds. For each configuration and seed, the final-10% return is the episode-count-weighted mean of the logged returns over the final 10% of training, approximately 500,000 transitions. We report the mean and standard error over the ten seeds. Within each task and method, the 36 configurations are sorted by mean final-10% return, with rank 1 best.

Figure 2 reports the setting-rank curves for Adam + PG, Muon + PG, and CD-NPG. Under on-policy training, CD-NPG's best settings are competitive and usually strongest, but the curves overlap and cross substantially, so the ordering remains task- and rank-dependent. Under off-policy training, the separation is clearer: on most tasks, the CD-NPG curve remains above the Adam + PG and Muon +

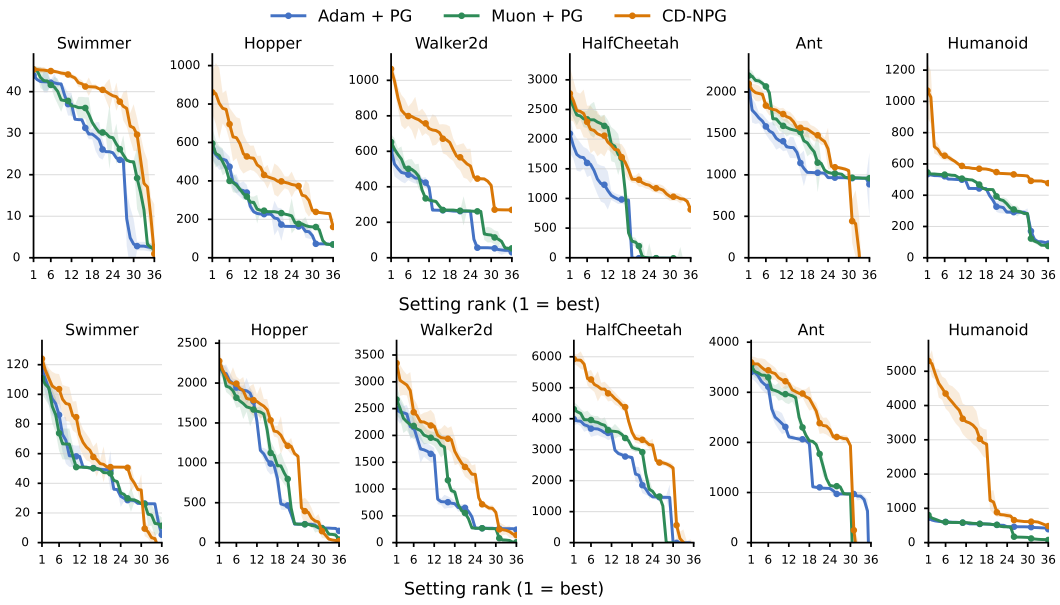

Figure 2: The setting-rank curves show that CD-NPG remains competitive across a broad range of hyperparameter choices. Its separation from Adam + PG and Muon + PG is clearest under off-policy training. The comparison covers six MuJoCo tasks. The top row shows on-policy training; the bottom row shows off-policy training with $K = 16$ actor and critic epochs per rollout. Within each task and method, settings are ranked independently by mean final-10% return, with rank 1 best and all 36 grid settings shown. Every configuration has ten finite tuning-seed scores. Curves are ten-seed means and bands are seed-level standard errors; task-specific vertical scales begin at zero. These are tuning-set summaries, not held-out estimates.

PG curves across a larger portion of the 36-setting grid. Thus the off-policy advantage is not limited to a single selected configuration; it also reflects a broader range of effective hyperparameter choices, although the ordering is not uniform at every rank on every task. This pattern persists at batch size 256 and with a $(256, 256)$ actor (Appendix Figures 5 and 6).

Figure 3 compares the actor-update costs of the three primary methods on HalfCheetah. CD-NPG incurs only a modest runtime overhead relative to Adam + PG and scales more favorably than Muon + PG as the actor width increases. Its persistent memory matches Adam's, while its peak explicit-tensor memory remains comparable to the two baselines. Similar timing trends hold across the other tasks (Appendix F.7). These fixed-setting kernel measurements isolate actor updates rather than end-to-end training throughput.

Figure 7 in Appendix F.4 compares CD-NPG with the richer CG-NPG and K-FAC-NPG approximations. K-FAC-NPG tends to be stronger under on-policy training, whereas CD-NPG tends to be stronger under off-policy training; CG-NPG is competitive but has no consistent advantage. Thus richer Fisher structure does not uniformly improve either the best return or robustness across the tuning grid. Figure 15 in Appendix F.7 reports the corresponding actor-update costs. Here the ordering is clearer: CD-NPG is consistently faster and scales more gently with actor width than both richer approximations. Under the same explicit-tensor accounting, Table 3 shows that its persistent memory lies between those of CG-NPG and K-FAC-NPG, while its peak memory remains close to CG-NPG's and substantially below K-FAC-NPG's.

We next isolate the roles of direction clipping and full-Fisher normalization. Figure 4 uses HalfCheetah to examine the two modifications. Panels (a)–(b) focus on clipping. Without clipping, returns plateau at larger target radii, whereas layer-wise clipping continues to benefit from increasing δ. Panel (b) explains this difference: the loss of locality illustrated in Figure 1 also occurs during neural-network training. Even after the clipped and raw directions are separately normalized to the same full-Fisher local budget, clipped updates remain close to the target finite-step KL, while

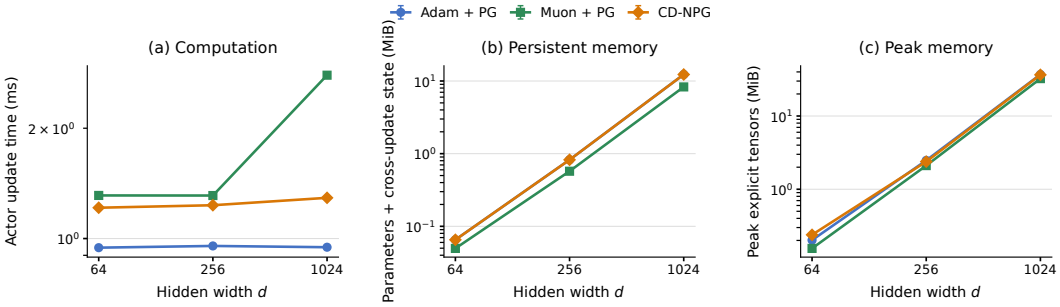

Figure 3: CD-NPG remains close to Adam's actor-update cost and scales more gently than Muon as actor width increases. HalfCheetah results use batch size $B = 64$. (a) Synchronized time per complete actor update (mean $\pm$ standard error over 200 updates after 100 warm-up updates). (b) Persistent memory: actor parameters and fixed policy buffers plus algorithm-state tensors retained across updates. (c) A manual source-level ledger of peak explicit tensors, including the supplied actor batch and the largest set of named update temporaries that are simultaneously live in any algorithmic phase. Rollout and critic work, CUDA-library workspaces, autograd-engine bookkeeping, allocator cache, and framework scaffolding are excluded. For a uniform-precision comparison, every method uses 32-bit floating-point (FP32) parameters, gradients, optimizer state, and update arithmetic; method-specific low-precision or reduced-refresh optimizations are excluded. All vertical axes are logarithmic.

raw directions with large coordinate displacements can leave the local regime by many orders of magnitude. Clipping therefore mitigates finite-step nonlocality rather than merely changing the nominal step length. Appendix Figures 11 and 12 extend this locality comparison across tasks on the training and held-out states, respectively.

Panels (c)–(d) examine step-length calibration. Panel (c) shows that $Q_t^{\mathrm{int}}/Q_t^{\mathrm{diag}}$ grows systematically with actor size, providing an empirical counterpart to the correlated-Fisher example in Section 3.2. The diagonal quadratic therefore increasingly understates the direction's full-Fisher length as the model grows. Panel (d) further shows that this discrepancy is not a fixed scale factor: full-Fisher normalization tracks the target KL throughout training, whereas the diagonal-normalized counterfactual both overshoots and varies over time. Appendix Figure 8 extends these finite-step KL traces across all tasks and both training settings, showing that the mismatch varies across conditions and over training rather than behaving as a single fixed scale factor. Thus the two modifications address distinct failures: clipping keeps the direction local, while full-Fisher normalization calibrates its policy-space length. Appendix F.5 extends these diagnostics across tasks and training settings, and Appendix F.6 reports the full clipping ablations.

5 Discussion and Conclusion

Our main evaluation focuses on continuous-control MuJoCo tasks with fixed-variance Gaussian policies parameterized by MLPs. Appendix G reports a preliminary extension to four Atari tasks using categorical policies parameterized by Nature-CNNs. Within the tested on-policy grids, CD-NPG outperforms Adam on all four tasks and CG-NPG on all four, while remaining broadly competitive with Muon across the tuning distributions. K-FAC-NPG is strongest overall in Figure 16. Thus CD-NPG remains competitive in a substantially different policy class, although this evidence is limited to four on-policy Atari tasks. The K-FAC-NPG result is also consistent with within-layer correlation structure being particularly useful for convolutional policies.

Several empirical questions remain unanswered. The evaluation contains only six MuJoCo and four Atari tasks, and the Atari study considers only on-policy training. The task coverage and hyperparameter searches are therefore not broad enough to establish generality across environments, training settings, or model scales. The evaluated policy classes are limited to fixed-variance Gaussian MLPs and categorical Nature-CNNs; learned variances, recurrent policies, larger architectures, and other observation modalities remain open.

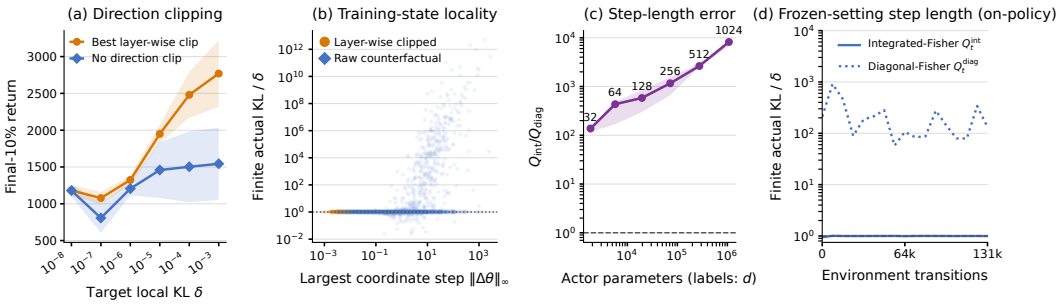

Figure 4: Direction clipping and integrated-Fisher normalization address distinct failures. On-policy HalfCheetah diagnostics show: (a) final-10% return for the best layer-wise threshold at each δ versus no direction clipping at the same δ (ten-seed mean $\pm$ standard error after five million transitions). Selection and reporting use the same tuning seeds. (b) Sample-average finite-step KL on the same 64 training states used for step normalization, divided by δ and plotted against the largest coordinate displacement. The clipped update and raw counterfactual are separately normalized by their own Q_t^{int}. The scatter contains 5,865 correlated seed–checkpoint records, obtained from three seeds by logging the first and final updates and every 16th update over 31,250 updates per seed. (c) $Q_t^{\mathrm{int}}/Q_t^{\mathrm{diag}}$ versus actor parameter count for $d \in \{32, 64, 128, 256, 512, 1024\}$; points are medians of three within-seed temporal medians and bands span the seeds over 16,384 transitions. (d) Checkpoint-wise three-seed medians of sample-average finite-step KL divided by δ for the applied integrated-Fisher normalization (solid) and non-mutating diagonal counterfactual (dotted), at the setting frozen to the final-10%-return winner of the main HalfCheetah on-policy grid through 131,072 transitions. Dashed horizontal lines mark one.

The method also leaves theoretical and algorithmic questions unresolved. Its proposal direction remains coordinate- and layer-partition dependent and discards Fisher correlations; the scalar full-Fisher normalization uses those correlations only to measure the selected direction's length. Clipping provides neither reparameterization invariance nor an absolute parameter-step bound. The scalar quadratic is conditional on sampled states, so current-policy action integration does not guarantee a distribution-wide finite KL.

Taken together, the experiments show that, in the evaluated settings, diagonal NPG can be a practical and competitive policy optimizer when its direction and step length are controlled. Performance relative to Adam, Muon, CG-NPG, and K-FAC-NPG remains task- and training-setting-dependent. CD-NPG therefore does not establish a uniformly superior optimizer or recover full NPG. Rather, it occupies a return–cost middle ground: it achieves competitive returns across the evaluated grids, while incurring only modest actor-update overhead over Adam and substantially lower cost than the richer natural-gradient approximations.

This return–cost position also suggests several extensions. The clipped diagonal direction could initialize CG, potentially allowing a few Fisher–vector products to recover part of the benefit of a longer solve. Possible approaches between the diagonal method and K-FAC include adapting unitwise quasi-diagonal metrics from neural-network natural-gradient training, using smaller layer-wise blocks, or applying K-FAC selectively on layers where correlations matter most. These directions treat CD-NPG as a low-cost foundation for richer natural-gradient methods rather than as an endpoint.

Reproducibility Statement

Appendix B provides the theoretical details, Appendices C and D specify the complete proposed and baseline updates, and Appendix E gives the analytic finite-step locality example. Appendices F and G report the MuJoCo and Atari environments, architectures, training protocols, hyperparameter grids, seed counts, evaluation rules, numerical safeguards, and timing environment. The experimental campaigns underlying the reported figures were audited at the raw-log level, and the return summaries follow the complete-case rules specified in Appendix F.

AI USE STATEMENT

Generative AI tools assisted with research brainstorming and conceptual framing; implementation, debugging, and experiment orchestration; analysis and interpretation of results; visualization; and manuscript organization and language editing. All AI-assisted code, analyses, figures, and text were reviewed by the authors. Code and experiments were checked through testing and raw-log inspection, and mathematical and empirical claims were manually verified. The authors take responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the aid of generative AI.

REFERENCES

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Lukas Balles, Javier Romero, and Philipp Hennig. Coupling adaptive batch sizes with learning rates. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler (eds.), *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence (UAI)*, Sydney, Australia, August 2017. URL https://www.auai.org/uai2017/proceedings/papers/141.pdf. Paper 141.

Jeremy Bernstein, Jiawei Zhao, Kamyar Azizzadenesheli, and Anima Anandkumar. signSGD with majority vote is communication efficient and fault tolerant. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1810.05291.

Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Sciences*, 7(4):448–461, 1973. doi: 10.1016/S0022-0000(73)80033-9.

Semih Cayci, Niao He, and R. Srikant. Finite-time analysis of entropy-regularized neural natural actor-critic algorithm. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=BkEqk7pS1I.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJlrF24twB.

Brennan Gebotys, Alexander Wong, and David A. Clausi. Understanding the effects of second-order approximations in natural policy gradient reinforcement learning. *arXiv preprint arXiv:2201.09104*, 2022. URL https://arxiv.org/abs/2201.09104.

Ian J. Goodfellow. Efficient per-example gradient computations. *arXiv preprint arXiv:1510.01799*, 2015. URL https://arxiv.org/abs/1510.01799.

Roger Grosse and James Martens. A kronecker-factored approximate fisher matrix for convolution layers. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 573–582, 2016. URL https://proceedings.mlr.press/v48/grosse16.html.

Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Théophane Weber, David Silver, and Hado Van Hasselt. Muesli: Combining improvements in policy optimization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4214–4226, 2021. URL https://proceedings.mlr.press/v139/hessel21a.html.

Dongseong Hwang. FAdam: Adam is a natural gradient optimizer using diagonal empirical fisher information. *arXiv preprint arXiv:2405.12807*, 2024. URL https://arxiv.org/abs/2405.12807.

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

Sham M. Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 2001. URL https://proceedings.neurips.cc/paper/2001/hash/4b86abe48d358ecf194c56c69108433e-Abstract.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Hong Liu, Zhiyuan Li, David Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=3xHDeA8Noi.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, et al. Muon is scalable for LLM training. *arXiv preprint arXiv:2502.16982*, 2025.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2408–2417, 2015. URL https://proceedings.mlr.press/v37/martens15.html.

James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller (eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pp. 479–535. Springer Berlin Heidelberg, 2 edition, 2012. doi: 10.1007/978-3-642-35289-8_27.

Yann Ollivier. Riemannian metrics for neural networks I: Feedforward networks. *Information and Inference: A Journal of the IMA*, 4(2):108–153, 2015. doi: 10.1093/imaiai/iav006.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994. doi: 10.1162/neco.1994.6.1.147.

Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. Parallel training of DNNs with natural gradient and parameter averaging. In *International Conference on Learning Representations Workshop*, 2015. URL https://arxiv.org/abs/1410.7455.

Nicol N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14(7):1723–1738, 2002. doi: 10.1162/08997660260028683.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1506.02438.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Arsalan Sharifnassab, Mohamed Elsayed, Kris De Asis, A. Rupam Mahmood, and Richard S. Sutton. Intentional updates for streaming reinforcement learning. *arXiv preprint arXiv:2604.19033*, 2026. URL https://arxiv.org/abs/2604.19033. Accepted at the 43rd International Conference on Machine Learning (ICML 2026).

Mingfei Sun. Randomized advantage transformation (RAT): Computing natural policy gradients via direct backpropagation. *arXiv preprint arXiv:2605.18591*, 2026. URL https://arxiv.org/abs/2605.18591. Accepted at the 43rd International Conference on Machine Learning (ICML 2026).

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12, 2000.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Jiayi Weng, Min Lin, Shengyi Huang, Bo Liu, Denys Makoviichuk, Viktor Makoviychuk, Zichen Liu, Yufan Song, Ting Luo, Yukun Jiang, Zhongwen Xu, and Shuicheng Yan. EnvPool: A highly parallel reinforcement learning environment execution engine. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Yuhuai Wu, Elman Mansimov, Roger B. Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/361440528766bbaaaa1901845cf4152b-Abstract.html.

Gorkem Yilmaz. A cross-domain empirical benchmark of quantum-inspired and classical optimization algorithms for machine learning. *Quantum Machine Intelligence*, 8(103), 2026. doi: 10.1007/s42484-026-00444-y.

Appendix Contents

Appendix A: Related Work 12

Appendix B: Local Comparison of Diagonal NPG and Sign-SGD 13

Appendix C: Complete Clipped Diagonal Natural Policy Gradient 15

Appendix D: Baseline Algorithms 20

Appendix E: Analytic Example of Finite-Step Loss of Locality 24

Appendix F: Complete MuJoCo Evaluation 25

Appendix G: Atari Results 39

A Related Work

Approximate NPG methods differ mainly in how much Fisher correlation structure they retain. TRPO approximately solves the full system with matrix-free Fisher–vector products, while K-FAC and Kronecker-factored trust-region actor–critic (ACKTR) retain structured layer-wise blocks (Schulman et al., 2015; Martens & Grosse, 2015; Wu et al., 2017). Replacing the Fisher by its diagonal discards all cross-coordinate score correlations and reduces the natural-gradient solve to elementwise scaling, avoiding both iterative linear solves and block factorizations. Diagonal and related elementwise Fisher approximations have previously been evaluated in policy-gradient methods. In their baseline comparison of Fisher approximations for NPG, Gebotys et al. (2022) reported that the diagonal method had the lowest aggregate performance and by far the worst stability score, with late-training divergence in multiple environments; its main advantage was computation time. Their diagonal statistic is the coordinatewise square of a batch-mean score, rather than the mean of coordinatewise squared per-example scores used here. Sun (2026) develop a richer sample-space NPG construction computed by direct backpropagation; their Sophia baseline is built around a diagonal score-Fisher approximation and performs poorly across the reported continuous-control tasks, substantially below their richer construction.

The closest precedent for our step-length normalization is TRPO. After conjugate gradient produces a proposal direction d, TRPO applies the same Fisher-quadratic normalization as CD-NPG, $\alpha = \sqrt{2\delta/(d^\top F d)}$, before performing a line search (Schulman et al., 2015). The methods differ in how they choose d: TRPO approximately solves the full-Fisher system, whereas CD-NPG clips an inverse-diagonal proposal. CD-NPG uses the shared normalization only to set the proposal's scalar length and does not add a line search. Other methods provide related global step controls. K-FAC uses a separately evaluated exact-Fisher quadratic to rescale an approximate natural-gradient direction, and its convolutional extension clips such a direction by a Fisher or damped metric norm (Martens &

Grosse, 2015; Grosse & Martens, 2016). ACKTR instead caps its step using an approximate-Fisher quadratic (Wu et al., 2017). The protocol of Gebotys et al. (2022) tunes between backtracking line search and ACKTR-style step-size clipping, and Sun (2026) applies global ℓ_2 clipping to the policy update. A related non-NPG method, Intentional Policy Gradient, controls the change in the sampled action's log probability using RMSProp-style diagonal scaling (Sharifnassab et al., 2026). These mechanisms control overall update magnitude but do not change coordinate ratios created by inverse-diagonal scaling.

The closest algebraic precedent for our direction clipping is Sophia, which applies an elementwise cap after diagonal Hessian or Gauss–Newton preconditioning to bound extreme coordinate updates when the curvature model is unreliable (Liu et al., 2024). The clipping operations are algebraically similar, but the approximated matrices are different: CD-NPG diagonalizes the policy Fisher, whereas Sophia diagonalizes a loss Hessian or Gauss–Newton curvature matrix. CD-NPG also uses a layer-relative median threshold rather than Sophia's fixed elementwise cap. Other methods impose coarser constraints on preconditioned directions. FAdam uses per-tensor RMS clipping, a recent diagonal second-moment natural-gradient benchmark uses a per-tensor ℓ_2 clip (Hwang, 2024; Yilmaz, 2026), and Povey et al. (2015) impose a maximum change on each layer. For neural natural actor–critic methods, Cayci et al. (2024) project the natural direction into hidden-unit group-norm balls. Related methods instead change the retained structure or the optimization target: quasi-diagonal metrics retain selected bias–weight correlations (Ollivier, 2015), while Muesli clips advantages and interprets the result in policy-logit coordinates (Hessel et al., 2021). Thus the individual ideas of diagonal-curvature clipping, group- or layer-level constraints, and Fisher-metric normalization all have precedents. Our contribution is their policy-specific, role-separated combination: an unweighted per-example policy-score diagonal, a within-layer median-relative coordinate cap, and a separate action-integrated current-policy Fisher normalization.

B Local Comparison of Diagonal NPG and Sign-SGD

A diagonal-natural direction and a sign direction preserve the same ascent signs but use different information about relative coordinate magnitudes. The question is whether the sensitivity-corrected magnitudes improve local policy optimization or whether cross-coordinate Fisher terms make them harmful. We answer this question after scaling the two directions to equal exact current-policy KL. The result separates the benefit of magnitude heterogeneity from its relative cost under the full Fisher.

Let J denote expected return, and fix parameters θ, a locally fixed-support differentiable policy family, and a state weighting P_X. The exact current-policy KL along a displacement Δ is

$$\mathcal{K}_\theta(\Delta) = \mathbb{E}_{x\sim P_X}\left[D_{\mathrm{KL}}(\pi_\theta(\cdot \mid x) \,\|\, \pi_{\theta+\Delta}(\cdot \mid x))\right]. \tag{14}$$

Let $g = \nabla J(\theta)$ and $F = \nabla^2_\Delta \mathcal{K}_\theta(0)$. Under the usual regularity conditions, F is the current-policy Fisher

$$F = \mathbb{E}_{\substack{x\sim P_X \\ a\sim\pi_\theta(\cdot|x)}}\left[\nabla \log \pi_\theta(a \mid x) \nabla \log \pi_\theta(a \mid x)^\top\right].$$

Write $D = \mathrm{diag}(F)$. Assume $g \neq 0$, let $\mathcal{S} = \{i : g_i \neq 0\}$, and assume $D_{ii} > 0$ on $\mathcal{S}$. The population diagonal-natural and sign directions are

$$p_i = \begin{cases} g_i/D_{ii}, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}, \end{cases} \qquad s = \mathrm{sign}(g), \qquad \mathrm{sign}(0) = 0. \tag{15}$$

The direction s deliberately removes all relative-magnitude information while preserving the ascent signs. It is the population comparator associated with sign-SGD (Bernstein et al., 2019), not an endpoint of the median-relative clipping range used by CD-NPG.

The comparison separates magnitude heterogeneity from the Fisher correlations that a diagonal approximation omits. Define

$$\begin{gathered} W = \sum_{i\in\mathcal{S}} D_{ii}, \qquad w_i = \frac{D_{ii}}{W}, \qquad q_i = |p_i|, \qquad \bar{q}_w = \sum_{i\in\mathcal{S}} w_i q_i, \\ H_{\mathrm{mag}} = \frac{\sum_{i\in\mathcal{S}} w_i q_i^2}{\bar{q}_w^2} = 1 + \frac{\sum_{i\in\mathcal{S}} w_i (q_i - \bar{q}_w)^2}{\bar{q}_w^2}. \end{gathered} \tag{16}$$

Thus H_{mag} is the weighted heterogeneity of the diagonal-natural magnitudes; it equals one exactly when those magnitudes are constant on the active coordinates. To measure the relative effect of the omitted Fisher terms, define

$$\zeta(v) = \frac{v^\top F v}{v^\top D v}, \qquad C_{\mathrm{corr}} = \frac{\zeta(p)}{\zeta(s)}, \qquad \rho_F(v) = \frac{g^\top v}{\sqrt{v^\top F v}}. \tag{17}$$

The relative cross-coordinate factor C_{corr} can be above or below one; it is not intrinsically a penalty.

Assume J is twice continuously differentiable near θ, $\mathcal{K}_\theta$ is three times continuously differentiable near zero, and $p^\top F p, s^\top F s > 0$. Under standard local domination and moment conditions, these assumptions hold at regular points for fixed-covariance Gaussian policies with smooth means and categorical softmax policies with finite action spaces and smooth logits. For sufficiently small $\delta > 0$, let $\alpha_v(\delta) > 0$ be the local positive solution of $\mathcal{K}_\theta(\alpha_v v) = \delta$, for $v \in \{p, s\}$, and set $\theta_v^+(\delta) = \theta + \alpha_v(\delta) v$.

Proposition 1 (Magnitude benefit versus Fisher-correlation cost)**.** *The two directions satisfy*

$$\left(\frac{\rho_F(p)}{\rho_F(s)}\right)^2 = \frac{H_{\mathrm{mag}}}{C_{\mathrm{corr}}}. \tag{18}$$

When their steps are matched by exact current-policy KL,

$$J\big(\theta_p^+(\delta)\big) - J\big(\theta_s^+(\delta)\big) = \sqrt{2\delta}\,\rho_F(s)\left[\sqrt{\frac{H_{\mathrm{mag}}}{C_{\mathrm{corr}}}} - 1\right] + O(\delta). \tag{19}$$

Consequently, for all sufficiently small δ, the exact-KL-matched endpoint along p has larger J when $H_{\mathrm{mag}} > C_{\mathrm{corr}}$, whereas the endpoint along s has larger J when $H_{\mathrm{mag}} < C_{\mathrm{corr}}$. Equality is inconclusive at this order.

Proof. For either fixed direction v, the local expansions are

$$\mathcal{K}_\theta(\alpha v) = \tfrac{1}{2}\alpha^2 v^\top F v + O(\alpha^3), \qquad J(\theta + \alpha v) = J(\theta) + \alpha g^\top v + O(\alpha^2).$$

They give

$$\alpha_v(\delta) = \sqrt{\frac{2\delta}{v^\top F v}} + O(\delta), \qquad J(\theta_v^+(\delta)) - J(\theta) = \sqrt{2\delta}\,\rho_F(v) + O(\delta).$$

It remains to compare the two coefficients. Let $A = g^\top D^{-1} g$ on $\mathcal{S}$ and $L = \|g\|_1$. Since $g_i = D_{ii} s_i q_i$,

$$A = W \sum_i w_i q_i^2, \qquad L = W \bar{q}_w, \qquad H_{\mathrm{mag}} = \frac{AW}{L^2}.$$

Moreover, $p^\top D p = A$, $s^\top D s = W$, and $g^\top p = A$, $g^\top s = L$. Substitution therefore yields

$$\left(\frac{\rho_F(p)}{\rho_F(s)}\right)^2 = \frac{A^2/(p^\top F p)}{L^2/(s^\top F s)} = \frac{AW}{L^2}\frac{\zeta(s)}{\zeta(p)} = \frac{H_{\mathrm{mag}}}{C_{\mathrm{corr}}}.$$

Combining this identity with the two local return expansions proves the claim. □

The proposition gives a precise reason to preserve diagonal magnitudes rather than automatically reduce the proposal to its signs. If all q_i are equal, then p is a positive multiple of s, so their exact-KL-matched endpoints coincide. If the q_i are nonconstant and $\zeta(p) \le \zeta(s)$, then $H_{\mathrm{mag}} > 1 \ge C_{\mathrm{corr}}$, which is sufficient for the endpoint along p to have larger local return. In particular, when $F = D$, $C_{\mathrm{corr}} = 1$, and any nonconstant set of q_i favors p. Cross-coordinate Fisher effects reverse the ordering only when they make $C_{\mathrm{corr}} > H_{\mathrm{mag}}$.

The diagonal-Fisher case also marks an important boundary for clipping. Let $D_{\mathcal{S}}$ and $g_{\mathcal{S}}$ denote the restrictions to the active coordinates. When $F = D$, Cauchy–Schwarz gives, for every nonzero v supported on $\mathcal{S}$,

$$\frac{g^\top v}{\sqrt{v^\top D v}} \le \sqrt{g_{\mathcal{S}}^\top D_{\mathcal{S}}^{-1} g_{\mathcal{S}}} = \frac{g^\top p}{\sqrt{p^\top D p}}. \tag{20}$$

Thus clipping an exact population direction p cannot improve its ideal local gain. Clipping must instead earn its value by controlling estimation error or a finite-step failure.

The proposition is a local population comparison of two directions after exact policy-KL matching. It is designed to remove step-length conventions from that comparison; it does not model the diagonal normalizer of a particular optimizer, gradient-estimation or Fisher-estimation error, EMA lag, or a finite clipping rule. It therefore gives no universal ranking between practical diagonal NPG and sign-based optimization. Instead, it identifies the boundary: diagonal-magnitude heterogeneity helps when its local-gain benefit exceeds the relative full-Fisher cost created by cross-coordinate terms.

For clipped diagonal NPG, the implication is limited but important. Relative coordinate magnitudes cannot be treated as dispensable in general, so clipping should not be justified as an approximation to a sign direction. A selective rule can preserve this potentially useful information while addressing failures of the sampled finite update. Whether it does so in training lies outside the population comparison and remains an empirical question.

C Complete Clipped Diagonal Natural Policy Gradient

This appendix gives the complete definitions needed to implement CD-NPG. It specifies the on-policy and off-policy actor objectives, the fused score statistics, the action-integrated Fisher quadratic used for step-length calibration, the full update, and its leading dense-network cost.

C.1 Actor minibatches and objective

Every algorithm receives an actor minibatch $\mathcal{D}_t = \{(s_i, a_i, \widehat{A}_i)\}_{i=1}^{B}$ and treats $\widehat{A}_i$ as fixed during the actor update. In Section 4, $\widehat{A}_i$ is a generalized advantage estimate (GAE) produced by the shared critic (Schulman et al., 2016) and is held fixed across off-policy optimization epochs. Advantage estimation is therefore outside the actor optimizers below. For the behavior policy $\pi_{\theta_{t,\mathrm{old}}}$ associated with minibatch $\mathcal{D}_t$, define the likelihood ratio and, depending on the algorithm, the on-policy log-policy or off-policy PPO objective (Schulman et al., 2017) by

$$\rho_{t,i}(\theta) := \frac{\pi_\theta(a_i \mid s_i)}{\pi_{\theta_{t,\mathrm{old}}}(a_i \mid s_i)},$$

$$\mathcal{L}_t(\theta) := \begin{cases} \frac{1}{B}\sum_i \widehat{A}_i \log \pi_\theta(a_i \mid s_i), & \text{on-policy policy gradient,} \\ \frac{1}{B}\sum_i \min\big(\rho_{t,i}(\theta)\widehat{A}_i, \mathrm{clip}(\rho_{t,i}(\theta), 0.8, 1.2)\widehat{A}_i\big), & \text{off-policy PPO.} \end{cases}$$

At the on-policy evaluation point $\theta = \theta_{t,\mathrm{old}}$, the gradient of the log-policy objective equals the gradient of the unclipped likelihood-ratio surrogate. The two are therefore gradient-equivalent at that point, not identical objectives over a finite parameter neighborhood. Throughout the algorithm appendices, $\odot$ and $\oslash$ denote elementwise multiplication and division, respectively. Functions such as $\sqrt{\cdot}$ and $\max$ are also applied elementwise when their arguments are vectors.

C.2 Fused score backward pass

For an actor with L affine layers, let $H_{t,0}$ be the matrix of minibatch observations and, for $j = 1, \ldots, L$, define

$$z_{t,j} := H_{t,j-1} W_j^\top + \mathbf{1}_B b_j^\top, \qquad H_{t,j} := \phi_j(z_{t,j}).$$

Thus $H_{t,j-1} \in \mathbb{R}^{B \times d_{j-1}}$ contains the input activations and $z_{t,j} \in \mathbb{R}^{B \times d_j}$ the preactivations of layer j. An ordinary differentiable forward pass evaluates $\log \pi_{\theta_{t-1}}(a_i \mid s_i)$ for every example and retains the usual reverse-mode intermediates $\{H_{t,j-1}, z_{t,j}\}_{j=1}^{L}$. During the corresponding backward pass, the gradient arriving at layer j's preactivations is

$$E_{t,j} := \nabla_{z_{t,j}} \sum_{i=1}^{B} \log \pi_{\theta_{t-1}}(a_i \mid s_i) \in \mathbb{R}^{B \times d_j}.$$

Because minibatch examples do not interact in the actor forward pass, row i of $E_{t,j}$ is exactly the score error for example i. To make the advantage weighting and PPO clipping explicit, define

$$\alpha_{t,i} := \frac{\partial \mathcal{L}_t}{\partial \log \pi_\theta(a_i \mid s_i)}\bigg|_{\theta=\theta_{t-1}} = \frac{1}{B}\begin{cases} \widehat{A}_i, & \text{on-policy policy gradient,} \\ \rho_{t,i}(\theta_{t-1})\widehat{A}_i, & \text{off-policy PPO, unclipped branch,} \\ 0, & \text{off-policy PPO, clipped branch.} \end{cases}$$

The same score-backward pass therefore supplies both the advantage-weighted policy gradient and the unweighted per-example score errors needed for the Fisher diagonal in Algorithm 1.

The score errors above must next be reduced to a parameter-space diagonal. For an affine layer, this can be done without materializing per-example parameter gradients. For $z_i = Wh_i + b$, let $H \in \mathbb{R}^{B\times d_{\text{in}}}$ contain rows $h_i^\top$ and let $E \in \mathbb{R}^{B\times d_{\text{out}}}$ contain the corresponding score errors $e_i^\top$. Since $\nabla_W \log \pi_\theta(a_i \mid s_i) = e_i h_i^\top$, element (j,k) of the per-example weight score is $e_{ij}h_{ik}$. Therefore

$$\widetilde{v}_t[W] = \frac{1}{B}(E \odot E)^\top (H \odot H), \qquad \widetilde{v}_t[b] = \frac{1}{B}\sum_{i=1}^{B} e_i \odot e_i. \tag{21}$$

This contraction is not new. Martens & Sutskever (2012, Sec. 11.3) gave the same squared-activation/squared-error matrix product for an empirical-Fisher diagonal. Goodfellow (2015) later used the related affine outer-product structure to compute per-example gradient norms; Balles et al. (2017) used the squared-matrix product for coordinatewise gradient second moments; and Dangel et al. (2020) systematized that computation in modular backpropagation. Here the contraction is applied to unweighted policy scores, so it yields the exact minibatch coordinatewise score second moment rather than the square of the batch-mean score.

Computationally, equation 21 replaces a $B \times P$ per-example parameter Jacobian by one dense matrix multiplication per layer. It uses $O(Bd_{\text{in}}d_{\text{out}}) = O(BP)$ arithmetic and produces only an $O(P)$ statistic, while reusing the activations and score errors already available to backpropagation. Explicit per-example parameter gradients would require $O(BP)$ additional storage.

The closed form is layer-specific, not a property of arbitrary modules. It avoids per-example parameter-gradient storage when each example makes only one additive contribution to a parameter; multiple contributions must be summed before they are squared (Martens & Sutskever, 2012, Sec. 11.3). The activation following an affine layer may be any parameter-free, sample-separable differentiable function because its derivative is already contained in E. For convolutional or recurrent weights shared across positions or time, however, each per-example weight gradient sums several outer products before it is squared, so cross-position terms prevent direct use of equation 21. Parameterized nonlinear modules and modules that couple examples, such as training-mode batch normalization, also need dedicated rules. A layer-specific contraction or batched/chunked per-example automatic differentiation remains exact but can have different memory and runtime costs. The MuJoCo actors studied here are dense multilayer perceptrons (MLPs), for which equation 21 applies to every weight and bias tensor.

During off-policy epochs, this unweighted moment averages current-parameter scores over stored behavior-policy actions, so it is an off-policy score metric rather than the current-policy Fisher diagonal. Step length is instead calibrated by Q_t^{int} below, whose action expectation uses the current policy at the stored states.

C.3 Action-integrated Fisher quadratics

The stored-action score moment above does not measure the current policy's local KL during off-policy epochs. To set the step length in current-policy geometry, we instead integrate over current-policy actions at the stored states. Define the resulting Fisher and its directional quadratic by

$$\overline{F}_t := \frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{a\sim\pi_{\theta_{t-1}}(\cdot|s_i)}[q_{t,i}(a)q_{t,i}(a)^\top], \qquad Q_t^{\text{int}} := d_t^\top \overline{F}_t d_t, \tag{22}$$

where $q_{t,i}(a) = \nabla_\theta \log \pi_\theta(a \mid s_i)|_{\theta=\theta_{t-1}}$. The state expectation is a minibatch average; the action expectation uses the current policy even when the stored actions come from an older policy. Let

the policy be a differentiable distribution $p_{\eta_\theta(s)}$ with output parameters $\eta_\theta(s)$. The derivation below applies beyond fixed-variance Gaussian policies.

Only the scalar $d_t^\top \overline{F}_t d_t$ is needed because CD-NPG has already chosen d_t. This is a cheaper special case of standard Fisher pullback and trust-region machinery: the natural gradient uses the pullback $J^\top G J$, and TRPO computes its products without forming the matrix before scaling a proposed direction by $\sqrt{2\delta/(d^\top F d)}$ (Amari, 1998; Schulman et al., 2015). A policy-output Jacobian–vector product (JVP) supplies Jd_t, after which an output-space Fisher contraction gives the scalar; no vector–Jacobian product (VJP) or iterative Fisher solve is necessary. Directional derivatives and linear-time Fisher products follow established constructions (Pearlmutter, 1994; Schraudolph, 2002); we spell out the JVP here to fix the quantity used by the implementation.

For a differentiable vector-valued function $f : \mathbb{R}^P \to \mathbb{R}^K$, define

$$\mathrm{JVP}(f, \theta, d) := \left(f(\theta), \dot{f}_\theta[d]\right), \qquad \dot{f}_\theta[d] := \frac{\mathrm{d}}{\mathrm{d}\varepsilon} f(\theta + \varepsilon d)\bigg|_{\varepsilon=0} = J_f(\theta) d. \tag{23}$$

Thus a JVP returns the ordinary forward value together with the change of that value induced to first order by the parameter direction d.

For the current minibatch, define the policy-output map and its Jacobian at the current iterate by

$$\Phi_t(\theta) := \left\{\eta_\theta(s_i) : (s_i, a_i, \widehat{A}_i) \in \mathcal{D}_t\right\}, \qquad J_{\eta,t,i} := \nabla_\theta \eta_\theta(s_i)|_{\theta=\theta_{t-1}}. \tag{24}$$

Thus the subscript t records both the minibatch $\mathcal{D}_t$ and the evaluation point θ_{t-1}. The JVP in the CD-NPG direction d_t returns $\dot{\eta}_{t,i} := J_{\eta,t,i} d_t$. Let the output-space Fisher be

$$G(\eta) := \mathbb{E}_{a\sim p_\eta}\left[\nabla_\eta \log p_\eta(a) \nabla_\eta \log p_\eta(a)^\top\right]. \tag{25}$$

The standard Fisher pullback identity gives

$$Q_t^{\mathrm{int}} = \frac{1}{B}\sum_{i=1}^{B} \dot{\eta}_{t,i}^\top G(\eta_{t,i}) \dot{\eta}_{t,i}, \qquad \eta_{t,i} = \eta_{\theta_{t-1}}(s_i). \tag{26}$$

This evaluates the required scalar after one JVP; because the direction is already fixed, no VJP, parameter-space Fisher–vector product, or CG solve is required.

Equation 26 separates the policy-network JVP from a small contraction in the output space of the action distribution. The following specializations show that this construction is not tied to a particular action family.

For a diagonal Gaussian, let $\Sigma := \mathrm{diag}(\sigma_1^2, \ldots, \sigma_A^2)$ and $p_\eta = \mathcal{N}(\mu, \Sigma)$. For fixed σ, take $\eta = \mu$. With $J_{\mu,t,i} := \nabla_\theta \mu_\theta(s_i)|_{\theta=\theta_{t-1}}$,

$$q_{t,i}(a) = J_{\mu,t,i}^\top \Sigma^{-1}(a - \mu_{t,i}), \qquad Q_t^{\mathrm{int}} = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{A} \frac{\dot{\mu}_{t,i,j}^2}{\sigma_j^2}, \quad \dot{\mu}_{t,i} = J_{\mu,t,i} d_t. \tag{27}$$

This follows from $\mathbb{E}[(a-\mu)(a-\mu)^\top] = \Sigma$. If the policy also predicts $\ell = \log\sigma$, take $\eta = (\mu, \ell)$; the mean–log-scale Fisher is block diagonal and the contraction becomes

$$Q_t^{\mathrm{int}} = \frac{1}{B}\sum_{i=1}^{B}\sum_{j=1}^{A} \left(\frac{\dot{\mu}_{t,i,j}^2}{\sigma_{t,i,j}^2} + 2\dot{\ell}_{t,i,j}^2\right). \tag{28}$$

Both tangents are returned by one JVP through the policy outputs.

For a categorical policy with logits $z \in \mathbb{R}^A$ and probabilities $\pi = \mathrm{softmax}(z)$, the output-space Fisher is $G(z) = \mathrm{diag}(\pi) - \pi\pi^\top$. If $J_{z,t,i} := \nabla_\theta z_\theta(s_i)|_{\theta=\theta_{t-1}}$ and $\dot{z}_{t,i} := J_{z,t,i} d_t$, then

$$Q_t^{\mathrm{int}} = \frac{1}{B}\sum_{i=1}^{B}\left[\sum_{a=1}^{A} \pi_{t,i,a} \dot{z}_{t,i,a}^2 - \left(\sum_{a=1}^{A} \pi_{t,i,a} \dot{z}_{t,i,a}\right)^2\right]. \tag{29}$$

Thus a JVP of the logits followed by a categorical-variance reduction computes the exact action-integrated quadratic; no action enumeration by backward passes is required.

When the output-space Fisher contraction is unavailable, draw $a_{i,r} \sim \pi_{\theta_{t-1}}(\cdot \mid s_i)$ and use

$$Q_t^{\text{sample}} = \frac{1}{BR} \sum_{i=1}^{B} \sum_{r=1}^{R} \left[\frac{\mathrm{d}}{\mathrm{d}\varepsilon} \log \pi_{\theta_{t-1}+\varepsilon d_t}(a_{i,r} \mid s_i) \Big|_{\varepsilon=0} \right]^2. \tag{30}$$

Conditional on the states, this is an unbiased but noisier estimator of Q_t^{int}. It is distinct from the realized finite-step KL. If the latter is not analytic, it can instead be estimated after proposing $\theta_t = \theta_{t-1} + \Delta\theta_t$ by

$$\widehat{K}_t^{\text{sample}} = \frac{1}{BR} \sum_{i=1}^{B} \sum_{r=1}^{R} \left[\log \pi_{\theta_{t-1}}(a_{i,r} \mid s_i) - \log \pi_{\theta_t}(a_{i,r} \mid s_i) \right], \quad a_{i,r} \sim \pi_{\theta_{t-1}}(\cdot \mid s_i). \tag{31}$$

The first quantity predicts infinitesimal KL from a Fisher quadratic; the second estimates the finite-step KL and is a diagnostic or line-search signal, not part of the default update.

C.4 Complete CD-NPG update

CD-NPG is the proposed diagonal policy-geometry method. It uses EMAs of the policy gradient and the diagonal score second moment to construct a diagonally preconditioned direction and clip it relative to the median update magnitude within each layer. It then evaluates that direction under the action-integrated current-policy Fisher for KL normalization. This separates inexpensive diagonal direction estimation from a full-parameter, action-integrated current-policy quadratic check while avoiding a parameter-space Fisher matrix.

The fused realization below assumes that the trainable actor parameters are affine-layer weights and biases and that the disjoint sets $\{\mathcal{I}_\ell\}_{\ell=1}^{L}$ cover all of them. The MuJoCo policies fix the Gaussian log standard deviation, so no trainable parameter is left ungrouped. A policy with additional trainable parameters requires both an explicit group assignment and a score-diagonal estimator for those parameters.

Algorithm 1 CD-NPG

Input: differentiable policy $\pi_{\theta_0}(\cdot \mid s) = p_{\eta_{\theta_0}(s)}$, with $G(\eta) = \mathbb{E}_{a\sim p_\eta}[\nabla_\eta \log p_\eta(a) \nabla_\eta \log p_\eta(a)^\top]$; actor minibatches $\mathcal{D}_t$, δ, finite $c_{\mathrm{rel}} \in [1, \infty)$, a disjoint layer partition $\{\mathcal{I}_\ell\}_{\ell=1}^L$ of all trainable actor parameters, $\beta_1, \beta_2 \in [0, 1)$, $\epsilon_F = 10^{-12}$, $\epsilon_Q = 10^{-12}$.

$m_0 \leftarrow 0;\ v_0 \leftarrow 0$ ▷ *initialize EMAs*
for $t = 1, 2, \ldots$ **do**
  $\big(\{\log \pi_{\theta_{t-1}}(a_i \mid s_i)\}_{i=1}^B, \{H_{t,j-1}, z_{t,j}\}_{j=1}^L\big) \leftarrow$
  $\mathrm{forward}(\pi_{\theta_{t-1}}, \mathcal{D}_t)$
  $\alpha_{t,i} \leftarrow \left.\dfrac{\partial \mathcal{L}_t}{\partial \log \pi_\theta(a_i \mid s_i)}\right|_{\theta=\theta_{t-1}}, \quad i = 1, \ldots, B$
  during backward of $\sum_i \log \pi_{\theta_{t-1}}(a_i \mid s_i)$, **for** $j = L, \ldots, 1$ **do**
    $E_{t,j} \leftarrow \nabla_{z_{t,j}} \sum_i \log \pi_{\theta_{t-1}}(a_i \mid s_i)$ ▷ *gradient exposed by autograd*
    $g_t[W_j] \leftarrow E_{t,j}^\top \mathrm{diag}(\alpha_t) H_{t,j-1}; \quad g_t[b_j] \leftarrow E_{t,j}^\top \alpha_t$
    $f_t[W_j] \leftarrow \dfrac{1}{B}(E_{t,j} \odot E_{t,j})^\top (H_{t,j-1} \odot H_{t,j-1})$
    $f_t[b_j] \leftarrow \dfrac{1}{B}(E_{t,j} \odot E_{t,j})^\top \mathbf{1}_B$ ▷ *diagonal score moment*
  end for
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) f_t$
  $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
  $M_t \leftarrow \max(v_t / (1 - \beta_2^t), \epsilon_F)$ ▷ *elementwise diagonal metric*
  $u_t \leftarrow \widehat{m}_t \oslash M_t$
  $d_t \leftarrow u_t$
  for $\ell = 1, \ldots, L$ **do**
    $\tau_{t,\ell} \leftarrow c_{\mathrm{rel}} \,\mathrm{median}(\{|u_{t,k}| : k \in \mathcal{I}_\ell\})$ ▷ *layer-relative threshold*
    $d_t[\mathcal{I}_\ell] \leftarrow \mathrm{clip}(u_t[\mathcal{I}_\ell], -\tau_{t,\ell}, \tau_{t,\ell})$
  end for
  $\{\eta_{t,i}, \dot{\eta}_{t,i}\}_{i=1}^B \leftarrow \mathrm{JVP}(\Phi_t, \theta_{t-1}, d_t)$ ▷ $\dot{\eta}_{t,i} = J_{\eta,t,i} d_t$
  $Q_t^{\mathrm{int}} \leftarrow \dfrac{1}{B} \sum_{i=1}^B \dot{\eta}_{t,i}^\top G(\eta_{t,i}) \dot{\eta}_{t,i}$ ▷ *current-policy integrated Fisher*
  $\theta_t \leftarrow \theta_{t-1} + \sqrt{2\delta / \max(Q_t^{\mathrm{int}}, \epsilon_Q)}\, d_t$
end for

Output: updated policy π_{θ_t}.

C.5 Leading dense actor computation

Count one multiplication and one addition as two floating-point operations (FLOPs), and let $M_a = \sum_{l=1}^L d_{l-1} d_l$ be the number of actor weight coefficients and $R_a = \sum_{l=2}^L d_{l-1} d_l$ the coefficients traversed while propagating errors between layers. For a dense actor, an Adam forward and backward pass requires $2BM_a + 2BR_a + 2BM_a = 4BM_a + 2BR_a$ leading matrix-multiplication FLOPs, corresponding to one forward pass, error propagation, and weight-gradient contractions. Before KL normalization, our fused computation uses the same $2BM_a$ forward and $2BR_a$ score-error propagation, then contracts the layer-wise score errors once with the PPO coefficients to form the gradient ($2BM_a$) and once with squared activations to form the Fisher diagonal ($2BM_a$), totaling $6BM_a + 2BR_a$. After the direction is known, the JVP reevaluates the primal network ($2BM_a$), contracts activations with the parameter direction ($2BM_a$), and propagates activation tangents through later layers ($2BR_a$). Integrated normalization therefore gives $10BM_a + 4BR_a$ leading actor FLOPs. Relative to Adam, CD-NPG's additional work is $6BM_a + 2BR_a$, plus $O(B \sum_l (d_{l-1} + d_l))$ elementwise operations, the output-space Fisher contraction, layer-wise order-statistic selection, and $O(P)$ fused optimizer-state operations. These leading FLOP counts omit order-statistic comparisons. Exact medians can be selected in aggregate with $O(P)$ work in principle. The current training and timing backends use a library sort in every layer to obtain this median, incurring $O(\sum_\ell |\mathcal{I}_\ell| \log |\mathcal{I}_\ell|)$ comparisons. The diagonal-Gaussian and categorical contractions described above cost $O(BA)$. When $R_a \simeq M_a$, the resulting CD-NPG-to-Adam actor-only FLOP ratio is $7/3$; including a similarly sized critic, whose work is shared by both methods, gives approximately $5/3$. These arithmetic ratios do not predict wall time exactly because small matrix multiplications and the fused state-update kernels have different utilization and memory traffic.

D BASELINE ALGORITHMS

Every baseline receives the same actor minibatch and optimizes the same objective $\mathcal{L}_t$ defined in Appendix C.1. The subsections below specify each baseline independently and explain the optimizer geometry represented by each baseline in the comparison.

D.1 ADAM

Adam is our coordinatewise adaptive first-order baseline (Kingma & Ba, 2015). It averages the advantage-weighted policy gradient and its elementwise square, so its denominator mixes policy sensitivity with advantages and minibatch noise rather than estimating a policy Fisher. This baseline provides a standard, linear-memory optimizer with temporal gradient normalization against which to test policy-space curvature.

Algorithm 2 Adam policy-gradient actor baseline

Input: π_{θ_0}, actor minibatches $\mathcal{D}_t$, η, $c_g \in (0, +\infty]$, $\beta_1, \beta_2 \in [0, 1)$; $\epsilon_{\mathrm{Adam}} = 10^{-8}$. The choice $c_g = +\infty$ is the unclipped setting.

$m_0 \leftarrow 0;\ v_0 \leftarrow 0$ ▷ *initialize moments*
for $t = 1, 2, \ldots$ **do**
  $g_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta)|_{\theta_{t-1}}$ ▷ *ascent policy gradient*
  $\bar{g}_t \leftarrow g_t \min(1, c_g/(\|g_t\|_2 + 10^{-6}))$ ▷ *global-norm clipping*
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\bar{g}_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)\bar{g}_t \odot \bar{g}_t$
  $\widehat{m}_t \leftarrow m_t/(1 - \beta_1^t); \quad \widehat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ ▷ *bias correction*
  $\theta_t \leftarrow \theta_{t-1} + \eta \widehat{m}_t \oslash (\sqrt{\widehat{v}_t} + \epsilon_{\mathrm{Adam}})$
end for

Output: updated policy π_{θ_t}.

D.2 MUON

Adam controls for coordinatewise adaptation without a policy Fisher. To test whether structured non-Fisher matrix geometry changes that comparison, we use Muon (Jordan et al., 2024; Liu et al., 2025). It applies Nesterov momentum followed by an approximate polar orthogonalization to eligible hidden-to-hidden weight matrices. Input and output weights, biases, and other scalar or vector parameters use auxiliary Adam. Muon therefore tests a structured update geometry that is neither coordinatewise nor defined by the policy distribution. Its Newton–Schulz arithmetic can use single precision (FP32) or bfloat16 (BF16). In our experiments, all parameters, gradients, optimizer state, and update arithmetic use FP32, including the Newton–Schulz iterations, to keep numerical precision uniform across methods.

Algorithm 3 Muon policy-gradient actor baseline with auxiliary Adam

Input: actor parameters $\theta_0 = (\mathcal{W}_0, \Psi_0)$, where $\mathcal{W}$ denotes the hidden-to-hidden matrix weights and Ψ denotes all auxiliary parameters (input and output weights, scalars, and vectors); actor minibatches $\mathcal{D}_t$, η_{Muon}, η_{aux}, $\mu, \beta_1^{\text{aux}}, \beta_2^{\text{aux}} \in [0, 1)$; Newton–Schulz arithmetic $\tau_{\text{NS}} \in \{\text{FP32}, \text{BF16}\}$; $(a, b, c) = (3.4445, -4.7750, 2.0315)$, $\epsilon_{\text{aux}} = 10^{-8}$.

$U_0^W \leftarrow 0$ for each $W \in \mathcal{W}$; $m_0^\psi, v_0^\psi \leftarrow 0$ for each $\psi \in \Psi$ ▷ *initialize state*
for $t = 1, 2, \ldots$ **do**
  $g_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta)|_{\theta_{t-1}}$
  for each $W \in \mathcal{W}, W \in \mathbb{R}^{m \times n}$ **do**
    $G_t^W \leftarrow \text{slice}(g_t, W)$ ▷ *matrix gradient*
    $U_t^W \leftarrow \mu U_{t-1}^W + (1 - \mu) G_t^W$
    $H_t^W \leftarrow (1 - \mu) G_t^W + \mu U_t^W$ ▷ *Nesterov momentum*
    $H_t^W \leftarrow \text{cast}(H_t^W, \tau_{\text{NS}})$; transpose if $m > n$
    $X_0 \leftarrow H_t^W / (\|H_t^W\|_F + 10^{-7})$
    for $s = 0, \ldots, 4$ **do**
      $A_s \leftarrow X_s X_s^\top$
      $X_{s+1} \leftarrow a X_s + b A_s X_s + c A_s^2 X_s$
    end for
    undo transpose; cast X_5 to the parameter dtype
    $W_t \leftarrow W_{t-1} + \eta_{\text{Muon}} \sqrt{\max(1, m/n)} X_5$
  end for
  for each $\psi \in \Psi$ **do**
    $g_t^\psi \leftarrow \text{slice}(g_t, \psi)$
    $m_t^\psi \leftarrow \beta_1^{\text{aux}} m_{t-1}^\psi + (1 - \beta_1^{\text{aux}}) g_t^\psi$
    $v_t^\psi \leftarrow \beta_2^{\text{aux}} v_{t-1}^\psi + (1 - \beta_2^{\text{aux}}) (g_t^\psi \odot g_t^\psi)$
    $\widehat{m}_t^\psi \leftarrow m_t^\psi / (1 - (\beta_1^{\text{aux}})^t)$; $\widehat{v}_t^\psi \leftarrow v_t^\psi / (1 - (\beta_2^{\text{aux}})^t)$ ▷ *bias correction*
    $\psi_t \leftarrow \psi_{t-1} + \eta_{\text{aux}} \widehat{m}_t^\psi \oslash (\sqrt{\widehat{v}_t^\psi} + \epsilon_{\text{aux}})$ ▷ *auxiliary Adam*
  end for
end for

Output: updated policy π_{θ_t}.

D.3 Matrix-free CG-NPG and its relation to TRPO

Both non-Fisher baselines omit the policy metric that motivates NPG. CG-NPG therefore serves as the full-Fisher iterative control. It forms neither a Fisher matrix nor an EMA: every conjugate-gradient iteration applies the current-minibatch action-integrated Fisher to a parameter-space search direction.

The matrix-free product can be expressed through JVPs and VJPs. Appendix C.3 defines $\text{JVP}(f, \theta, v)$. For an output-space vector u, define its vector–Jacobian product (VJP)

$$\text{VJP}(f, \theta, u) := J_f(\theta)^\top u. \tag{32}$$

For the minibatch output map Φ_t and current-iterate Jacobians $J_{\eta,t,i}$ defined in equation 24, a JVP in an arbitrary CG search direction v gives $\dot{\eta}_{t,i}(v) := J_{\eta,t,i} v$. Applying the output-space Fisher and then a VJP yields the Fisher–vector product (FVP)

$$\text{FVP}_t(v) := \overline{F}_t v = \frac{1}{B} \sum_{i=1}^{B} J_{\eta,t,i}^\top G(\eta_{t,i}) J_{\eta,t,i} v = \text{VJP}\left(\Phi_t, \theta_{t-1}, \left\{ \frac{1}{B} G(\eta_{t,i}) \dot{\eta}_{t,i}(v) \right\}_{i=1}^{B}\right). \tag{33}$$

The JVP–output-Fisher–VJP composition therefore evaluates $\text{FVP}_t(v)$ without materializing the policy-output Jacobian or Fisher matrix. Define the damped matrix-free operator used by CG as

$$\text{DampedFVP}_t(v) := \text{FVP}_t(v) + \lambda_{\text{CG}} v. \tag{34}$$

Damping regularizes the linear solve. Following the practical TRPO convention, the same damped operator also defines the quadratic used to scale the initial proposal. The resulting regularized local model is $\frac{1}{2} \Delta\theta^\top (\overline{F}_t + \lambda_{\text{CG}} I) \Delta\theta$. TRPO's subsequent line search still evaluates the sample-average finite-step KL, which does not include the ridge term.

The damped quadratic can be recovered directly from the CG residual. CG solves the damped linear system $(\overline{F}_t + \lambda_{\mathrm{CG}} I)z = g_t$ and its iterates maintain the residual identity

$$r_k = g_t - \mathrm{DampedFVP}_t(z_k) = g_t - (\overline{F}_t + \lambda_{\mathrm{CG}} I) z_k. \tag{35}$$

Each z_k is a Krylov-subspace approximation to the damped natural-gradient direction $(\overline{F}_t + \lambda_{\mathrm{CG}} I)^{-1} g_t$. After the fixed budget of K_{CG} iterations, Algorithm 4 sets $d_t = z_{K_{\mathrm{CG}}}$. Equation 35 at $k = K_{\mathrm{CG}}$ therefore gives

$$\mathrm{DampedFVP}_t(d_t) = g_t - r_{K_{\mathrm{CG}}}, \qquad Q_t^{\mathrm{CG}} := d_t^\top \mathrm{DampedFVP}_t(d_t) = d_t^\top (g_t - r_{K_{\mathrm{CG}}}). \tag{36}$$

This identity holds for any finite K_{CG} in exact arithmetic; it does not assume that CG has converged. It lets the algorithm evaluate the damped quadratic used by practical TRPO from the stored final residual, without another Fisher–vector product. In floating-point arithmetic the equality holds up to the accumulated residual-recursion error.

For an on-policy rollout evaluated once at θ_{t-1}, the gradient of the TRPO likelihood-ratio surrogate equals g_t, and the Hessian of its average local KL constraint equals $\overline{F}_t$ (Schulman et al., 2015). Consequently, when $\mathcal{D}_t$ is the same full-rollout batch and the damping and CG conventions are matched, Algorithm 4 computes the same constrained quadratic step as TRPO before its backtracking line search. TRPO additionally evaluates the finite-step KL and surrogate improvement and shrinks or rejects the proposal. Our implementation instead accepts the locally normalized step directly, uses a fixed budget of K_{CG} CG iterations instead of a residual-based stopping rule, permits minibatch updates, and fixes the Gaussian log standard deviation in the MuJoCo experiments. The equivalence therefore applies to the on-policy, single-update setting; off-policy PPO updates are not TRPO updates.

Algorithm 4 Current-minibatch action-integrated matrix-free CG-NPG

Input: differentiable policy $\pi_{\theta_0}(\cdot \mid s) = p_{\eta_{\theta_0}(s)}$, with $G(\eta) = \mathbb{E}_{a \sim p_\eta}[\nabla_\eta \log p_\eta(a) \nabla_\eta \log p_\eta(a)^\top]$; actor minibatches $\mathcal{D}_t$, δ, $K_{\mathrm{CG}} \in \mathbb{N}_{>0}$, $\lambda_{\mathrm{CG}} > 0$, $\epsilon_Q = 10^{-12}$. No quantities are averaged across updates.

for $t = 1, 2, \ldots$ **do**
  $g_t \leftarrow \nabla_\theta \mathcal{L}_t(\theta)|_{\theta_{t-1}}$
  $z \leftarrow 0$; $r \leftarrow g_t$; $\xi \leftarrow r^\top r$; $p \leftarrow r$
  if $\xi = 0$ **then** $\theta_t \leftarrow \theta_{t-1}$; **continue** ▷ *zero gradient*
  for $k = 0, \ldots, K_{\mathrm{CG}} - 1$ **do** ▷ *at most K_{CG} operator evaluations*
    $w \leftarrow \mathrm{DampedFVP}_t(p)$; $\chi \leftarrow p^\top w$
    if $\chi \le 0$ or any quantity is nonfinite **then mark invalid; break**
    $\gamma \leftarrow \xi / \chi$; $\quad z^+ \leftarrow z + \gamma p$
    $r^+ \leftarrow r - \gamma w$; $\quad \xi^+ \leftarrow (r^+)^\top r^+$
    if $\xi^+ = 0$ **then** $z \leftarrow z^+$; $r \leftarrow r^+$; **break** ▷ *exact convergence; avoid $0/0$*
    $p \leftarrow r^+ + (\xi^+ / \xi) p$; $z \leftarrow z^+$; $r \leftarrow r^+$; $\xi \leftarrow \xi^+$
  end for
  if invalid then $\theta_t \leftarrow \theta_{t-1}$; **continue**
  $d_t \leftarrow z$; $\quad Q_t^{\mathrm{CG}} \leftarrow d_t^\top (g_t - r)$ ▷ *stored residual; no extra Fisher product*
  $\theta_t \leftarrow \theta_{t-1} + \sqrt{2\delta / \max(Q_t^{\mathrm{CG}}, \epsilon_Q)}\, d_t$ ▷ *no direction clip or line search*
end for

Output: updated policy π_{θ_t}.

D.4 K-FAC-NPG

K-FAC-NPG serves as a structured-curvature reference between a diagonal metric and a full matrix-free solve (Martens & Grosse, 2015; Wu et al., 2017). For each affine layer, it approximates the Fisher block by a Kronecker product of activation and score-error second moments, retains EMAs of both factors and the policy gradient, and solves the two factor systems by Cholesky. It models structured within-layer correlations but discards cross-layer Fisher blocks; its dense factorizations also make it substantially more expensive than CD-NPG at large width.

Under on-policy training, the score-error factors estimate structured current-policy Fisher blocks. During off-policy training with rollout reuse, however, the score errors are evaluated at the current

parameters on stored behavior-policy actions. The resulting factors therefore define a structured behavior-action score metric rather than current-policy Fisher blocks.

The activations $H_{t,j-1}$, preactivations $z_{t,j}$, and score errors $E_{t,j}$ are defined in Appendix C.2. To precondition a layer's weight and bias jointly, K-FAC absorbs the bias into homogeneous coordinates,

$$\bar{H}_{t,j-1} := [\, H_{t,j-1} \;\; \mathbf{1}_B \,], \qquad \bar{W}_j := [\, W_j \;\; b_j \,], \qquad z_{t,j} = \bar{H}_{t,j-1} \bar{W}_j^\top.$$

Let $r_j = 1$ when layer j has a bias and $r_j = 0$ otherwise, and define

$$\bar{d}_{A,j} := d_{j-1} + r_j, \qquad d_{G,j} := d_j.$$

An augmented layer gradient or direction is therefore in $\mathbb{R}^{d_{G,j} \times \bar{d}_{A,j}}$, its activation factor is in $\mathbb{R}^{\bar{d}_{A,j} \times \bar{d}_{A,j}}$, and its score-error factor is in $\mathbb{R}^{d_{G,j} \times d_{G,j}}$. For a layer with a bias, the final column of an augmented gradient or direction is the bias vector. For a layer without a bias, bars denote the unaugmented quantities. Let $\epsilon_{\mathrm{tr}} > 0$ be the smallest positive normal number in the factor arithmetic; it is a numerical safeguard rather than a tuned hyperparameter.

K-FAC solves each factor system without forming an explicit inverse. For a symmetric positive-definite matrix M, $\mathrm{CholeskySolve}(M, B)$ denotes the solution X of $MX = B$, where B may be a vector or matrix. The implementation uses a Cholesky factorization and two triangular solves without forming M^{-1}. The FP32 implementation first symmetrizes M and attempts this factorization without modification. If that factorization fails, it adds the smallest successful diagonal jitter from the sequence $\epsilon_{\mathrm{mach}}\ \mathrm{mean}(|\operatorname{diag} M|) \times \{1, 10, \ldots, 10^7\}$. This numerical fallback is not a searched damping hyperparameter. If jitter is added, the corresponding $\widetilde{\bar{A}}_{t,j}$ or $\widetilde{G}_{t,j}$ below denotes the stabilized matrix actually factorized; the direction and Q_t use those same stabilized factors.

Algorithm 5 EMA K-FAC-NPG with affine biases

Input: π_{θ_0}, affine actor layers, actor minibatches $\mathcal{D}_t$, δ, $\lambda_{\mathrm{KFAC}} > 0$, $\beta_1, \beta_2 \in [0, 1)$, $\epsilon_Q = 10^{-12}$.

$\bar{m}_{0,j}, \bar{A}_{0,j}, G_{0,j} \leftarrow 0$ for every layer j
for $t = 1, 2, \ldots$ **do**
  $\big(\{\log \pi_{\theta_{t-1}}(a_i \mid s_i)\}_{i=1}^{B}, \{\bar{H}_{t,j-1}, z_{t,j}\}_{j=1}^{L}\big) \leftarrow$
  forward$(\pi_{\theta_{t-1}}, \mathcal{D}_t)$
  $\alpha_{t,i} \leftarrow \dfrac{\partial \mathcal{L}_t}{\partial \log \pi_\theta(a_i \mid s_i)}\Big|_{\theta=\theta_{t-1}}, \quad i = 1, \ldots, B$
  during backward of $\sum_i \log \pi_{\theta_{t-1}}(a_i \mid s_i)$, **for** $j = L, \ldots, 1$ **do**
    $E_{t,j} \leftarrow \nabla_{z_{t,j}} \sum_i \log \pi_{\theta_{t-1}}(a_i \mid s_i)$
    $\bar{g}_{t,j} \leftarrow E_{t,j}^\top \operatorname{diag}(\alpha_t) \bar{H}_{t,j-1}$ ▷ *last column is the bias gradient*
    $\bar{m}_{t,j} \leftarrow \beta_1 \bar{m}_{t-1,j} + (1-\beta_1)\bar{g}_{t,j}$
    $\bar{A}_{t,j} \leftarrow \beta_2 \bar{A}_{t-1,j} + \dfrac{1-\beta_2}{B} \bar{H}_{t,j-1}^\top \bar{H}_{t,j-1}$
    $G_{t,j} \leftarrow \beta_2 G_{t-1,j} + \dfrac{1-\beta_2}{B} E_{t,j}^\top E_{t,j}$ ▷ *K-FAC factor EMAs*
    $\widehat{\bar{m}}_{t,j} \leftarrow \bar{m}_{t,j}/(1-\beta_1^t)$
    $\widehat{\bar{A}}_{t,j} \leftarrow \bar{A}_{t,j}/(1-\beta_2^t); \quad \widehat{G}_{t,j} \leftarrow G_{t,j}/(1-\beta_2^t)$
    $s_{t,j}^A \leftarrow \max\Big(\mathrm{Tr}(\widehat{\bar{A}}_{t,j})/\bar{d}_{A,j}, \epsilon_{\mathrm{tr}}\Big)$
    $s_{t,j}^G \leftarrow \max\Big(\mathrm{Tr}(\widehat{G}_{t,j})/d_{G,j}, \epsilon_{\mathrm{tr}}\Big)$
    $\kappa_{t,j} \leftarrow \sqrt{s_{t,j}^A / s_{t,j}^G}$ ▷ *trace-balanced damping ratio*
    $\widetilde{\bar{A}}_{t,j} \leftarrow \widehat{\bar{A}}_{t,j} + \kappa_{t,j}\sqrt{\lambda_{\mathrm{KFAC}}} I_{\bar{d}_{A,j}}$
    $\widetilde{G}_{t,j} \leftarrow \widehat{G}_{t,j} + \kappa_{t,j}^{-1}\sqrt{\lambda_{\mathrm{KFAC}}} I_{d_{G,j}}$
    $X_{t,j} \leftarrow \mathrm{CholeskySolve}(\widetilde{G}_{t,j}, \widehat{\bar{m}}_{t,j})$ ▷ *solve* $\widetilde{G}_{t,j} X_{t,j} = \widehat{\bar{m}}_{t,j}$
    $\bar{D}_{t,j} \leftarrow \mathrm{CholeskySolve}(\widetilde{\bar{A}}_{t,j}, X_{t,j}^\top)^\top$ ▷ *last column is the bias direction*
  end for
  for $j = 1, \ldots, L$ **do**
    if $r_j = 1$ **then**
      $D_t[W_j] \leftarrow \bar{D}_{t,j}[:, 1{:}d_{j-1}]; \quad D_t[b_j] \leftarrow \bar{D}_{t,j}[:, \bar{d}_{A,j}]$
    else $D_t[W_j] \leftarrow \bar{D}_{t,j}$
    end if
  end for
  $d_t \leftarrow \mathrm{vec}(D_t)$ ▷ *restore the actor parameter layout*
  $Q_t \leftarrow \sum_j \mathrm{Tr}(\bar{D}_{t,j}^\top \widetilde{G}_{t,j} \bar{D}_{t,j} \widetilde{\bar{A}}_{t,j})$
  $\theta_t \leftarrow \theta_{t-1} + \sqrt{2\delta / \max(Q_t, \epsilon_Q)}\, d_t$
end for

Output: updated policy π_{θ_t}.

Here β_1 controls the gradient EMA and β_2 controls both Kronecker-factor EMAs. They are optimizer hyperparameters, not numerical constants prescribed by Algorithm 5.

E Analytic Example of Finite-Step Loss of Locality

This example uses the on-policy geometry from Section 3: the population gradient and Fisher are defined under the same policy.

Figure 1 is an analytic example of a two-dimensional continuous-action bandit: there is no state or context input, and the reward depends only on the action $a = (a_1, a_2)$. It isolates a finite-step locality failure without assigning the weak coordinate a larger standardized noise perturbation. The policy is $\pi_\theta = \mathcal{N}(\mu(\theta), I_2)$, with two active coordinates and mean

$$\mu(\theta) = \begin{bmatrix} \theta_1 \\ \varepsilon\theta_2 + \frac{h}{2}\theta_2^2 \end{bmatrix}, \qquad (\theta_1, \theta_2) = (0, 0), \tag{37}$$

The reward is $r(a) = 1 - \frac{1}{2}[(a_1 - 1)^2 + a_2^2]$. The expected return is $J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[r(a)] = -\frac{1}{2}[(\mu_1(\theta) - 1)^2 + \mu_2(\theta)^2]$; the additive constant in r cancels the contribution of unit action variance.

Thus $\mu(\theta_0) = (0, 0)$ gives $J(\theta_0) = -\frac{1}{2}$. For a finite parameter step $p = (p_1, p_2)$ from θ_0, the Gaussian KL has the closed form

$$\begin{aligned} K^{\text{actual}}(p) &= D_{\text{KL}}(\pi_{\theta_0} \| \pi_{\theta_0+p}) = \tfrac{1}{2}\left[p_1^2 + \left(\varepsilon p_2 + \tfrac{h}{2} p_2^2\right)^2\right], \\ K^{\text{predicted}}(p) &= \tfrac{1}{2} p^\top F(\theta_0) p = \tfrac{1}{2}(p_1^2 + \varepsilon^2 p_2^2). \end{aligned} \tag{38}$$

The action expectation is evaluated analytically; neither quantity is estimated from a dataset or sampled actions. At θ_0, the population gradient is $g = (1, 0)$ and the Fisher is $F = \text{diag}(1, \varepsilon^2)$. To model a possible minibatch error, we prescribe the same standardized perturbation ξ in both coordinates rather than estimate it from a realized minibatch:

$$\widehat{g} = (1 + \xi,\, \varepsilon\xi), \qquad F^{-1}\widehat{g} = (1 + \xi,\, \xi/\varepsilon). \tag{39}$$

Thus the absolute gradient error is smaller in the second coordinate, yet its preconditioned parameter displacement grows as $1/\varepsilon$. Importantly, ε cancels in the locally Fisher-whitened direction $F^{1/2}F^{-1}\widehat{g} = (1 + \xi, \xi)$: the example does not claim that low Fisher alone amplifies local policy-space noise. For the figure's values $h = 1$, $\varepsilon = 0.05$, $\xi = 0.5$, and $\delta = 0.18$, the layer proposal is $(1.5, 10)$. Its median magnitude is 1.5, so $c_{\text{rel}} = 4/3$ gives threshold 2 and the clipped direction $(1.5, 2)$. Because the Fisher is diagonal at θ_0, the full-Fisher normalization used in Figure 1 is exactly normalization by $d^\top F d$. It gives

$$p_{\text{raw}} \approx (0.569210, 3.794733), \qquad p_{\text{clip}} \approx (0.598671, 0.798228), \qquad p_{\text{NPG}} = (0.6, 0),$$

where p_{NPG} normalizes the population direction $F^{-1}g = (1, 0)$. All three have predicted KL 0.18. Substituting the unrounded endpoints into equation 38 and J gives

branch	$J(\theta_0 + p)$	$K^{\text{actual}}(p)$
unclipped	-27.396894	27.466104
clipped	-0.144792	0.243463
population NPG	-0.080000	0.180000

which round to the values reported in Figure 1.

The problem appears after taking the finite step predicted by that local model. KL normalization fixes the local policy-space length $\frac{1}{2}\Delta\theta^\top F \Delta\theta = \delta$, but does not bound the parameter-space length and leaves $\Delta\theta_2 = O(1/\varepsilon)$. This is already a nonlocal parameter step as $\varepsilon \to 0$. The ignored nonlinear policy change $\frac{h}{2}\Delta\theta_2^2$ is then $O(1/\varepsilon^2)$, so the exact KL can scale as $O(1/\varepsilon^4)$ even though the predicted KL remains δ. Layer-wise relative clipping of $F^{-1}\widehat{g}$ before normalization changes the update's coordinate ratios and bounds the weak coordinate's proposal. In this construction, the informative first coordinate supplies an $O(1)$ Fisher norm, so the normalization multiplier and the resulting $\Delta\theta_2$ remain bounded. Clipping deliberately sacrifices exact reparameterization invariance to act as a locality safeguard: it keeps the final parameter displacement within a region where the current Fisher model is more credible. Noisy Fisher denominators are one route to this failure; large gradient numerators, correlated near-null directions, and the accumulation of many coordinates can produce the same practical issue. Such weak-sensitivity coordinates can also arise from saturated nonlinearities or internal rescalings of otherwise equivalent network parameterizations.

F Complete MuJoCo Evaluation

This appendix supplies the evidence needed to interpret the MuJoCo claims in Section 4. It first fixes the common protocol, search spaces, and return estimand. It then asks whether the primary ordering persists at a larger actor minibatch and actor width, whether richer Fisher structure gives a consistent return advantage, whether full-Fisher scalar calibration reduces the diagonal step-length error, whether layer-wise clipping improves finite-step locality and return, and what these actor updates cost.

F.1 Protocol and estimand

We evaluate Swimmer-v4, Hopper-v4, Walker2d-v4, HalfCheetah-v4, Ant-v4, and Humanoid-v4 through EnvPool. The actor and critic are separate two-hidden-layer multilayer perceptrons (MLPs)

with 64 ReLU units per layer. Every linear layer is initialized orthogonally with zero bias: hidden layers use gain $\sqrt{2}$, the actor output uses gain 0.01, and the critic output uses gain 1. The Gaussian actor has fixed standard deviation 0.1, and sampled actions are clipped only at the environment interface. The critic uses Adam with learning rate $3 \cdot 10^{-4}$, $(\beta_1, \beta_2) = (0.9, 0.999)$, and a global gradient-norm clipping threshold of 1. We do not normalize observations, rewards, or advantages and use neither entropy bonuses nor learning-rate schedules. Returns use discount $\gamma = 0.99$, and advantage estimates use the same discount with GAE parameter $\lambda = 0.95$ (Schulman et al., 2016). True terminations disable value bootstrapping; time-limit truncations bootstrap from the terminal observation while stopping the GAE recurrence at the episode boundary. The critic uses the unclipped squared value loss; the configured PPO value-clip radius is therefore inactive.

The shared architecture has one consequence for the Muon baseline: only the hidden-to-hidden matrix is eligible for the matrix update. Auxiliary Adam updates the input and output layers and all biases. Muon uses five Newton–Schulz steps, no weight decay, and no gradient clipping.

For the primary batch-size-64 evaluation, each configuration uses the ten paired seeds $1, \ldots, 10$ and an approximately five-million-transition budget: 5,000,000 and 4,999,168 transitions for on-policy and off-policy training, respectively. Each seed uses one environment stream with normal resets at episode boundaries. The on-policy algorithm collects 64 consecutive on-policy transitions and performs one actor and critic update on that minibatch. The off-policy algorithm collects 2,048 transitions, divides them into 32 minibatches of 64, and applies $K = 16$ PPO epochs with a likelihood-ratio clipping threshold of 0.2; actor and critic use the same K, and the 2,048 indices are reshuffled independently at every epoch. Table 2 gives the method-specific base grids; boundary extensions, ablations, and confirmation runs remain outside these grids. In particular, the CD-NPG no-clipping control $c_{\mathrm{rel}} = +\infty$ is not a base-grid setting; it is used only in the δ-matched direction-clipping analysis in Figure 13. The frozen CD-NPG campaigns used Python 3.11.10, CUDA 12.8, Gymnasium 1.3.0, and EnvPool 1.2.5; the on-policy grid used PyTorch 2.7.1+cu128 and the off-policy grid used 2.11.0+cu128. Source snapshots and manifests record the exact environment for every campaign.

Table 2: Declared MuJoCo hyperparameter search axes for Adam + PG, CD-NPG, Muon + PG, CG-NPG, and K-FAC-NPG.

method	first search axis	second search axis	fixed state
Adam + PG	$\eta \in 10^{\{-8,\ldots,-3\}}$	$c_g \in \{0.01, 0.1, 1, 10, 100, 1000\}$	$(\beta_1, \beta_2) = (.9, .999)$, $\epsilon = 10^{-8}$
CD-NPG	$\delta \in 10^{\{-8,\ldots,-3\}}$	$c_{\mathrm{rel}} \in \{1, 3, 10, 30, 100, 300\}$	$(\beta_1, \beta_2) = (.9, .999)$, $\epsilon_F = \epsilon_Q = 10^{-12}$
Muon + PG	$\eta_{\mathrm{Muon}} \in 10^{\{-8,\ldots,-3\}}$	$\eta_{\mathrm{aux}} \in 10^{\{-8,\ldots,-3\}}$	$\mu = .95$, aux. $(\beta_1, \beta_2) = (.9, .999)$
CG-NPG	$\delta \in 10^{\{-8,\ldots,-3\}}$	$\lambda_{\mathrm{CG}} \in 10^{\{-6,\ldots,-1\}}$	$K_{\mathrm{CG}} = 16$, $\epsilon_Q = 10^{-12}$
K-FAC-NPG	$\delta \in 10^{\{-8,\ldots,-3\}}$	$\lambda_{\mathrm{KFAC}} \in 10^{\{-6,\ldots,-1\}}$	$(\beta_1, \beta_2) = (.9, .999)$, $\epsilon_Q = 10^{-12}$

We use the ten paired seeds per configuration both to select one setting per task, method, and training protocol and to summarize that selected setting. These are tuning-set summaries, not independent-seed estimates. For the batch-size-64 grids, the effective endpoint is $T = 5{,}000{,}000$ transitions for on-policy training and $T = 4{,}999{,}168$ for off-policy training. The off-policy algorithm uses $K = 16$ epochs and complete 2,048-transition rollouts. For each task, method, and protocol, selection maximizes the mean seed-level return over logged records with $0.9T < t \leq T$, approximately the final 500,000 transitions. Each record contains the mean return of completed training episodes and the number of such episodes. Within a seed, we weight these record means by their episode counts; the ten seed-level means then receive equal weight. Thus the score is a training episodic-return summary, not the return of a separately evaluated deterministic policy, an area under the learning curve (AUC), or an average over the final one million transitions. Standard-error bars use the sample standard deviation of seed-level scores divided by $\sqrt{10}$. Setting-rank plots order each method's complete base-grid configurations from best (rank 1) to worst. A configuration enters only when all ten seed scores are finite; unavailable configurations reduce the reported count M rather than receiving a synthetic zero. The task-, method-, and protocol-specific M is reported in the corresponding caption or panel title. These complete-case summaries do not estimate an unconditional failure-adjusted return. Configuration and selected-setting curves use the same ten tuning seeds and are labeled accordingly. Uncertainty is computed from seed-level summaries or paired seed-level differences, not from correlated logged time points; each figure specifies its convention.

F.2 Primary comparison at batch size 256

The primary comparisons in Figure 2 use actor minibatches of 64. To test whether their ordering depends on this choice, we independently repeat the same three 6×6 search grids at batch size 256, with ten paired seeds per configuration and the same policy, critic, actor objectives, and transition budgets. The on-policy condition collects 256 new transitions per update and ends at $T = 4{,}999{,}936$. The off-policy condition retains a 2,048-transition rollout and $K = 16$ epochs but divides each epoch into eight minibatches of 256, ending at $T = 4{,}999{,}168$. The search axes remain those in Table 2; supplemental boundary-extension cells are excluded.

Figure 5 reports the complete original grids. Increasing the minibatch size does not erase the main pattern. In the on-policy panels, CD-NPG occupies the upper part of the ranking on several tasks, while Muon + PG remains competitive on HalfCheetah and Ant. Under off-policy training, the CD-NPG curves separate more clearly on most tasks, whereas Hopper remains favorable to Adam + PG. These differences extend well beyond rank one into the middle of the grids, indicating a change in hyperparameter robustness rather than only a different selected winner. As in the batch-size-64 comparison, these are tuning-set summaries, not held-out estimates.

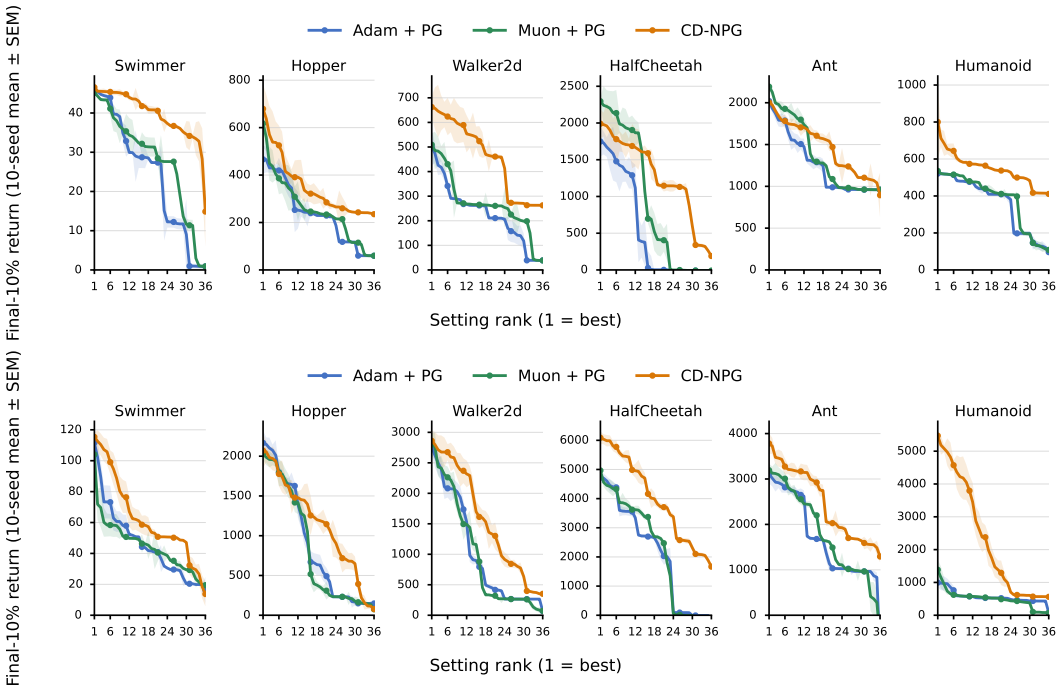

Figure 5: CD-NPG remains competitive when the actor minibatch increases to 256. Setting-rank distributions compare the three primary methods. The top row shows on-policy training; the bottom row shows off-policy training with $K = 16$ epochs. Within each task and method, the original 6×6 base-grid settings are ranked independently by mean final-10% return, with rank 1 best. Every panel contains all 36 settings for Adam + PG, Muon + PG, and CD-NPG, and every configuration has ten finite tuning-seed scores; supplemental boundary-extension cells are excluded. Curves are ten-seed means and bands are seed-level standard errors. Task-specific vertical scales begin at zero. These are tuning-set summaries, not held-out estimates.

F.3 Primary comparison with a wider ReLU actor

The primary comparisons use a two-hidden-layer actor with 64 units per layer. To test whether their ordering is specific to that width, we repeat the complete three-method comparison with a $(256, 256)$ ReLU actor while retaining the $(64, 64)$ ReLU critic, actor minibatch size 64, and all other protocol choices. For every task, method, and protocol, we independently evaluate the same method-specific 6×6 base grid as in Table 2, with ten paired seeds per configuration. The on-policy

runs use 5,000,000 transitions; the off-policy runs use 4,999,168 transitions and $K = 16$ epochs. Thus, this is a new tuning comparison at the wider architecture, not a transfer test of settings selected at width 64.

Figure 6 reports all 36 settings. The CD-NPG curves remain near the upper envelope across much of the grid on Hopper, Walker2d, HalfCheetah, and Humanoid under both training protocols. Swimmer and Ant show more crossings, with Adam + PG or Muon + PG competitive near the best ranks. The wider actor therefore preserves the broad pattern seen at width 64 rather than producing a uniform architecture-dependent reversal, but it also confirms that no method dominates every task.

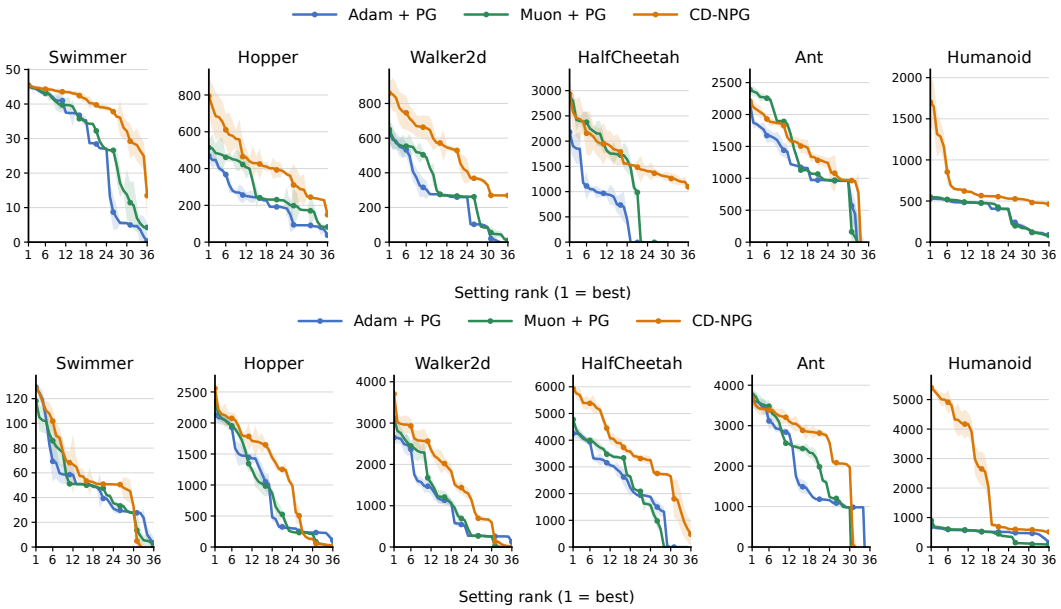

Figure 6: CD-NPG remains competitive with a wider actor, but the ordering stays task dependent. Setting-rank distributions use a $(256, 256)$ ReLU actor. The top row shows on-policy training; the bottom row shows off-policy training with $K = 16$ epochs. Within each task and method, all 36 base-grid settings are ranked independently by mean final-10% return, with rank 1 best. Every configuration has ten finite tuning-seed scores. Curves are ten-seed means and bands are seed-level standard errors. Task-specific vertical scales begin at zero. These are architecture-specific tuning-set summaries, not held-out estimates or hyperparameter-transfer results.

F.4 Comparison with richer natural-gradient approximations

The main comparison asks whether CD-NPG is competitive with non-Fisher actor optimizers. That result leaves open whether its diagonal geometry is consistently worse than richer natural-gradient approximations. Figure 7 therefore compares CD-NPG, CG-NPG with 16 iterations, and K-FAC-NPG across all complete settings with finite scores. Under on-policy training, K-FAC-NPG often occupies the upper ranks, consistent with useful within-layer correlation structure, while CD-NPG remains competitive on the tasks where that advantage is weaker. The ordering changes substantially under off-policy training: K-FAC-NPG can lose its on-policy advantage and deteriorate sharply in the lower-ranked settings, whereas CD-NPG retains broader strong regions on several of the larger tasks. CG-NPG is competitive in selected conditions but does not form a consistently dominant curve. Richer Fisher structure therefore changes the task-specific tradeoffs without producing a uniform return or robustness advantage. We keep the larger natural-gradient comparison here so that the main figure can show the three primary methods clearly; the appendix panels expose every rank and the unequal complete-setting counts.

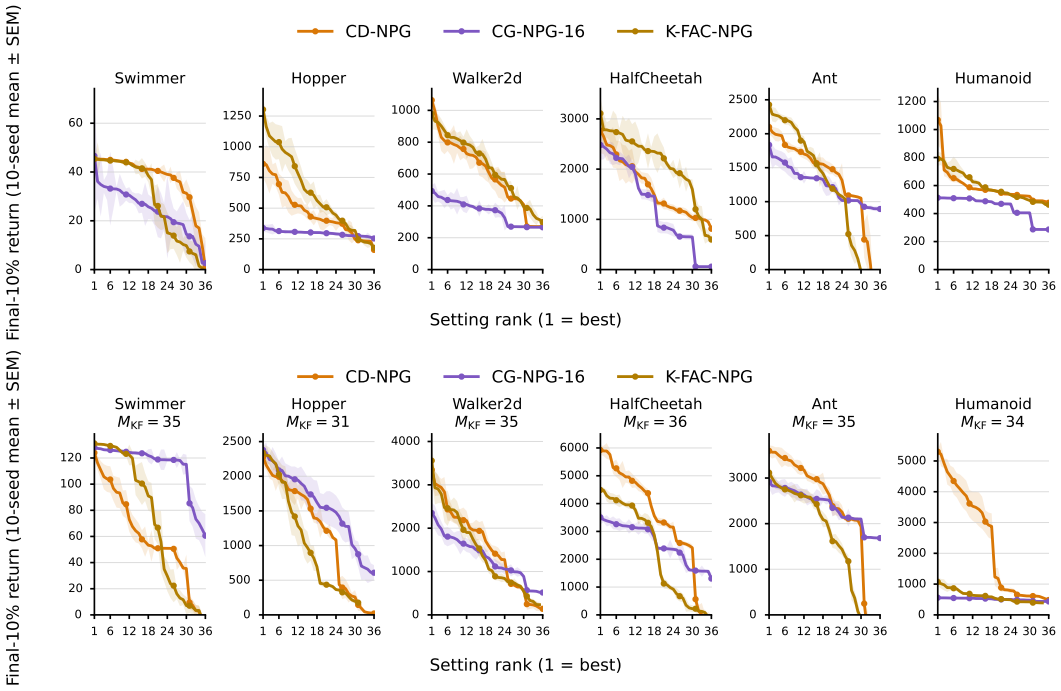

Figure 7: Richer Fisher structure does not provide a uniform return or robustness advantage. Setting-rank distributions compare CD-NPG, CG-NPG-16, and K-FAC-NPG. The top row shows on-policy training; the bottom row shows off-policy training with $K = 16$ epochs. Within each task and method, complete settings with ten finite seed scores are ranked by mean final-10% return. CD-NPG and CG-NPG-16 have 36 usable settings in every panel, as does on-policy K-FAC-NPG. Off-policy K-FAC-NPG has $M = (35, 31, 35, 36, 35, 34)$ for Swimmer through Humanoid; the second title line reports its usable count M_{KF}, and unavailable settings are excluded rather than assigned zero. Curves are ten-seed means and bands are seed-level standard errors. Task-specific vertical axes begin at zero. These are tuning-set summaries, not held-out estimates.

Return rankings do not reveal why the diagonal method can remain competitive. The first mechanism question is whether its full-Fisher scalar normalization repairs the step-length error left by the diagonal direction.

F.5 Testing full-Fisher step calibration

We test this claim with a paired diagnostic using the same layer-wise median-clipped CD-NPG update as in the reported MuJoCo sweeps. For its applied clipped direction d_t, define

$$Q_t^{\mathrm{diag}} = d_t^\top D_t d_t, \qquad Q_t^{\mathrm{int}} = d_t^\top \overline{F}_t d_t.$$

At each diagnostic checkpoint we form both finite counterfactual steps

$$h_t^{\mathrm{diag}} = \sqrt{2\delta / Q_t^{\mathrm{diag}}}\, d_t, \qquad h_t^{\mathrm{int}} = \sqrt{2\delta / Q_t^{\mathrm{int}}}\, d_t,$$

and evaluate their exact fixed-covariance Gaussian action KL on the same minibatch states. Only h_t^{int} is applied, so the diagonal-step measurements do not affect the training run or optimizer state.

This diagnostic isolates the normalization mechanism rather than measuring long-horizon return. It uses seeds 1–3, 131,072 transitions per seed, and the selected setting from each completed main grid, held fixed throughout. The model and the on- and off-policy schedules follow the base protocols above. The frozen settings are:

	On-policy		Off-policy	
Task	δ	c_{rel}	δ	c_{rel}
Swimmer	10^{-6}	10	10^{-5}	300
Hopper	10^{-5}	100	10^{-6}	100
Walker2d	10^{-5}	100	10^{-5}	10
HalfCheetah	10^{-3}	30	10^{-5}	100
Ant	10^{-4}	30	10^{-6}	1
Humanoid	10^{-4}	300	10^{-5}	30

Figure 8 shows a consistent separation between the two normalizers. The applied integrated-Fisher curves remain visually pinned to the target throughout training across tasks and protocols, whereas the diagonal counterfactual stays far above the target and fluctuates substantially over time. The mismatch is therefore neither a small constant offset nor a transient early-training effect. Checkpoints within each run are correlated, so the curves are mechanism summaries rather than independent inferential samples. This practical error reflects omitted correlations, sampling noise, EMA smoothing, and the Fisher floor, as well as behavior-action moments in the off-policy setting; it is not an estimate of the isolated off-diagonal contribution of the population Fisher.

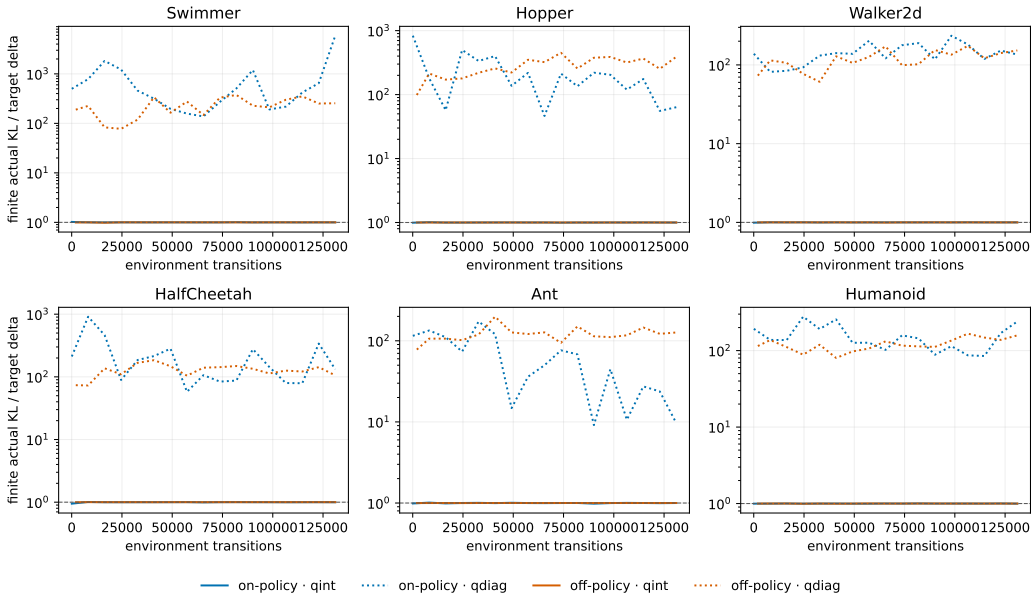

Figure 8: Integrated-Fisher normalization tracks the target KL, whereas diagonal normalization persistently overshoots. This frozen-setting diagnostic uses the final-10%-return winners of the main grids. Curves show the seed-median sample-average finite-step KL divided by δ for the applied integrated-Fisher normalization (solid) and non-mutating diagonal counterfactual (dotted). Blue and orange denote on-policy and off-policy training, respectively; dashed lines mark one.

The diagnostic above fixes the actor width at 64, so it does not show whether the diagonal-normalization error changes with parameter count. To separate model size from training duration, we run an on-policy diagnostic for 16,384 transitions at actor widths $d \in \{32, 64, 128, 256, 512, 1024\}$, with three seeds per task and the critic fixed at width 64. The task-specific (δ, c_{rel}), batch size 64, and all other choices are fixed across widths. Each seed is first summarized by its median over 33 diagnostic checkpoints, after which Figure 9 reports the three-seed median and range. Across all six tasks, the points form a steadily rising, nearly linear trend on the log–log axes, with relatively narrow seed ranges around most points. Task-specific levels differ, but the direction of the width effect is uniform: the practical diagonal-normalization error grows strongly with parameter count. The corresponding sample-average finite-step KL ratios closely track these quadratic ratios in the short diagnostics, so Figure 9 reports only the more direct quadratic comparison. This experiment

uses the sampled and EMA-processed diagonal; it does not attribute the entire trend to population off-diagonal terms alone.

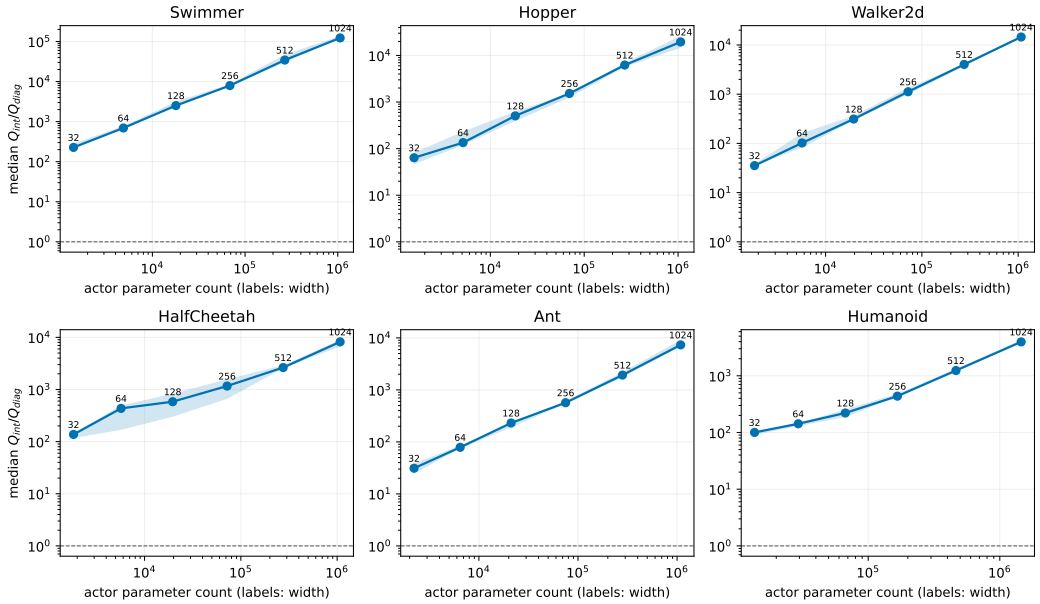

Figure 9: Diagonal-normalization error grows systematically with actor parameter count. The tested widths are $d \in \{32, 64, 128, 256, 512, 1024\}$, with labels beside points giving hidden width. Panels show $Q_t^{\text{int}}/Q_t^{\text{diag}}$ on the six tasks. The task-specific (δ, c_{rel}) values are frozen from the on-policy grids rather than selected per width. Both axes are logarithmic, and dashed lines mark one; the setup and aggregation are specified above.

F.6 Testing layer-wise direction clipping

Full-Fisher normalization calibrates scalar step length, but it does not prevent a direction from concentrating its parameter displacement in a few low-sensitivity coordinates. At the same diagnostic checkpoints, we retain the raw direction before clipping and normalize the raw and clipped directions separately by their own Q_t^{int}. We compare their finite KL at the same local Fisher budget. For each actor layer, we also record its clipping threshold and clipped fraction.

At the short diagnostic checkpoints, both branches usually remain close to the target after being normalized by their own full-Fisher quadratics. The immediate role of clipping is therefore not a systematic scalar rescaling. Figure 10 instead shows that clipping is selective: it concentrates in particular actor layers and training conditions while remaining nearly inactive in others. The thresholds also differ by orders of magnitude across layers, supporting a layer-relative reference rather than one global parameter-space scale.

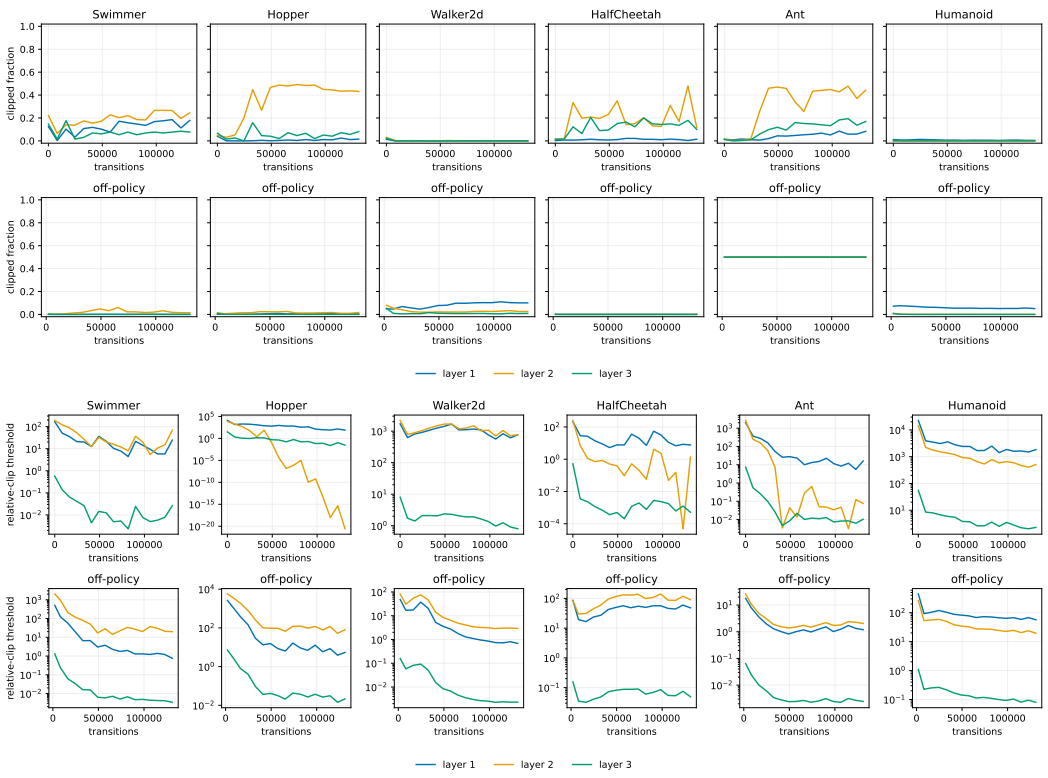

Figure 10: Layer-wise clipping is selective across layers and training conditions. This frozen-setting diagnostic uses the final-10%-return winners of the main grids. Columns are tasks. Within each diagnostic, the first row is on-policy and the second is off-policy. Upper two rows: seed-median fraction clipped in each actor layer. Lower two rows: median threshold $\tau_{t,\ell}$ from equation 11, on a log scale. Layer colors are shared across panels. Thresholds are in parameter coordinates and need not be comparable across layers.

These median summaries do not separate finite-update nonlinearity from error in estimating the current-policy state average. Each training update uses exactly 64 states. At each logged diagnostic update, we reset a separately seeded environment and use the unchanged pre-update policy to collect 1,024 held-out states. The direction and both step lengths depend only on the training states; the same candidate updates are evaluated on both sets. We run 31,250 updates (two million training transitions) for three seeds at every frozen task-specific setting. We log the first and final updates plus every 16th update, yielding 1,955 records per seed and $3 \times 1{,}955 = 5{,}865$ correlated records per panel in Figures 11 and 12.

Figure 11 shows that, on the states used for normalization, the clipped updates remain concentrated near the target, while the raw counterfactual develops a broad upper tail as its largest coordinate step grows. On the independent states in Figure 12, both clouds widen, revealing error in estimating the current-policy state average from one training minibatch. The raw branch nevertheless extends to much larger coordinate displacements and finite KL values than the clipped branch. The training-to-held-out shift is therefore shared by both directions, but the extreme nonlinear tail is much more pronounced without clipping. Clipping controls nonlocality but does not solve state-sampling error in the step-length quadratic.

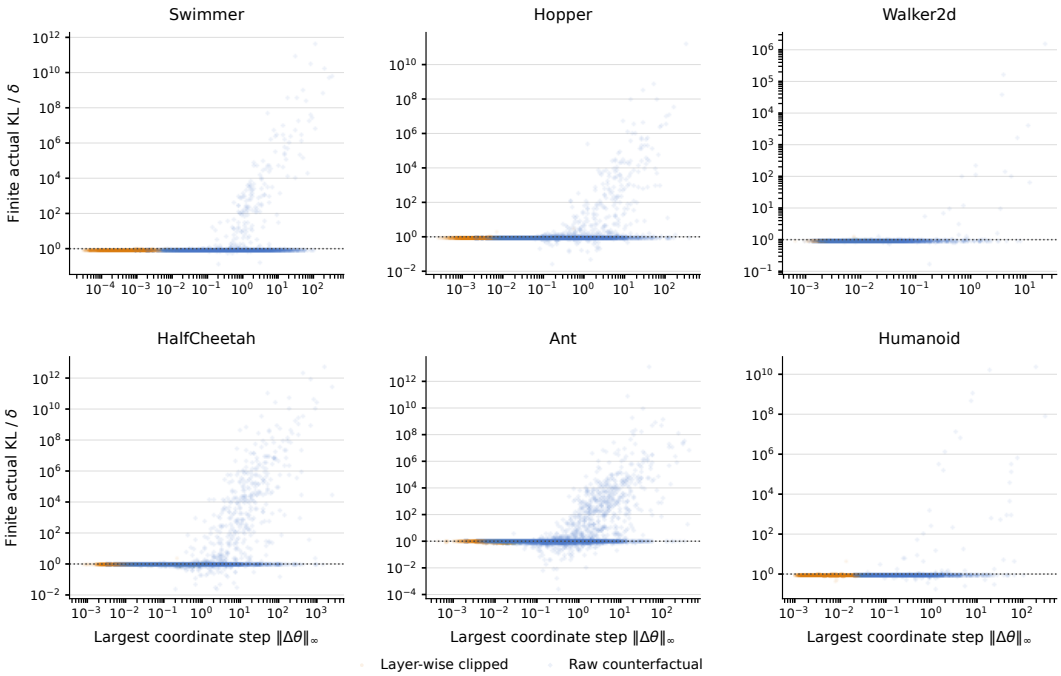

Figure 11: Clipping suppresses training-state finite-KL outliers associated with large coordinate steps. Sample-average finite-step KL divided by the target δ is evaluated after full-Fisher normalization on the 64 training states for all six tasks under on-policy training. The horizontal axis is the largest absolute coordinate step. The layer-wise clipped update and raw counterfactual are separately normalized by their own Q_t^{int} on these states. Each panel contains 5,865 correlated records obtained under the logging rule above from three seeds and 31,250 updates per seed using the task-specific frozen grid winner; dotted horizontal lines mark one.

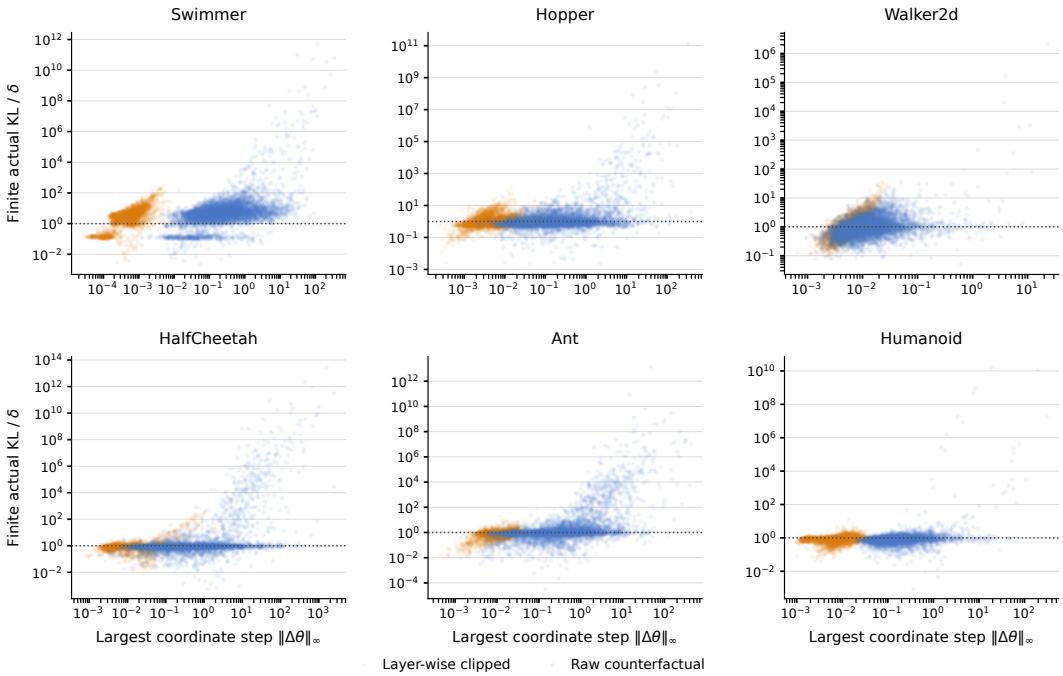

Figure 12: Held-out states reveal state-sampling error, while clipping limits the most extreme departures. Sample-average finite-step KL divided by the target δ is evaluated on an independent 1,024-state current-policy rollout for the same candidate updates as Figure 11. Direction choice and step normalization use only the 64 training states. The horizontal axis is the largest absolute coordinate step. Each panel contains the corresponding 5,865 correlated records; dotted horizontal lines mark one.

Taken together, these diagnostics separate finite-update nonlinearity from state-sampling error, but they do not show whether direction clipping improves return. We therefore extend the main-text HalfCheetah ablation to all six tasks and then compare layer-wise relative clipping with global fixed-value clipping. All directions use the same full-Fisher step normalization, so the comparisons isolate how the direction is controlled before its scalar length is set.

Figure 13 compares layer-wise relative clipping with no direction clipping at matched target radii. For each δ, the clipped curve shows the best mean return among the six $c_{\mathrm{rel}} \in \{1, 3, 10, 30, 100, 300\}$ settings, while the unclipped curve contains the unique $c_{\mathrm{rel}} = +\infty$ setting. At the smallest target radii the two curves are similar. As the radius increases, the unclipped curves often plateau or collapse, whereas the clipped curves continue to improve or retain their gains. The benefit of clipping therefore appears when the update is large enough for direction geometry to matter, rather than as a roughly constant return offset. This is a tuning-grid mechanism comparison: selection and reporting use the same ten seeds.

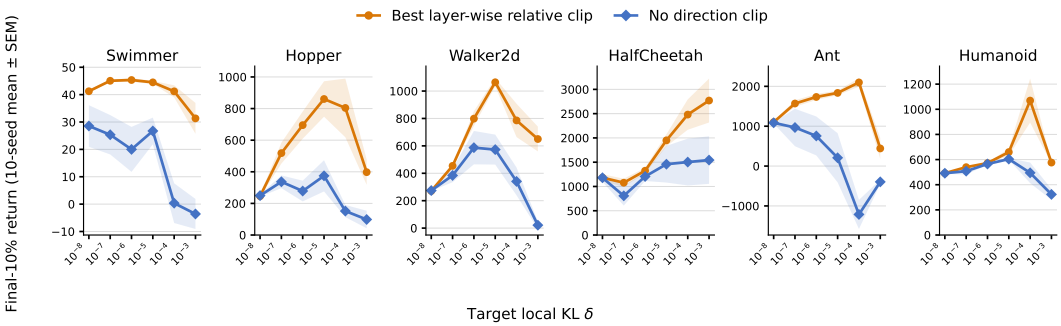

Figure 13: Clipping expands the range of useful target KL radii. This on-policy δ-matched ablation covers all six MuJoCo tasks. At each δ, the layer-wise curve shows the highest mean final-10% return among the six tested c_{rel} values; the no-clipping curve shows the single $c_{\mathrm{rel}} = +\infty$ setting. Curves are ten-seed means and bands are seed-level standard errors at five million transitions.

Layer-wise relative clipping uses a separate $c_{\mathrm{rel}}\,\mathrm{median}(|u_\ell|)$ threshold in each actor layer; global fixed-value clipping uses one absolute threshold c_d in the raw direction's units. Both geometry sweeps use $\delta \in 10^{\{-8,\ldots,-3\}}$. Layer-wise relative clipping uses $c_{\mathrm{rel}} \in \{1, 3, 10, 30, 100, 300\}$, whereas global fixed-value clipping uses $c_d \in \{0.01, 0.1, 1, 10, 100, 1000\}$. Each geometry therefore has the same declared 6×6 base-grid budget; incomplete off-policy cells are reported below.

Figure 14 reports the complete setting-rank distributions available for these two geometries. Under on-policy training, the layer-wise curves are usually above the fixed-value curves over broad parts of the grid, although the bands overlap and Ant shows a reversal. The off-policy panels are more mixed, with several crossings and no geometry maintaining an advantage across all tasks or ranks. The main difference is therefore not a universal best threshold family: grouping coordinates by layer changes both the attainable return and how quickly performance degrades away from the best settings.

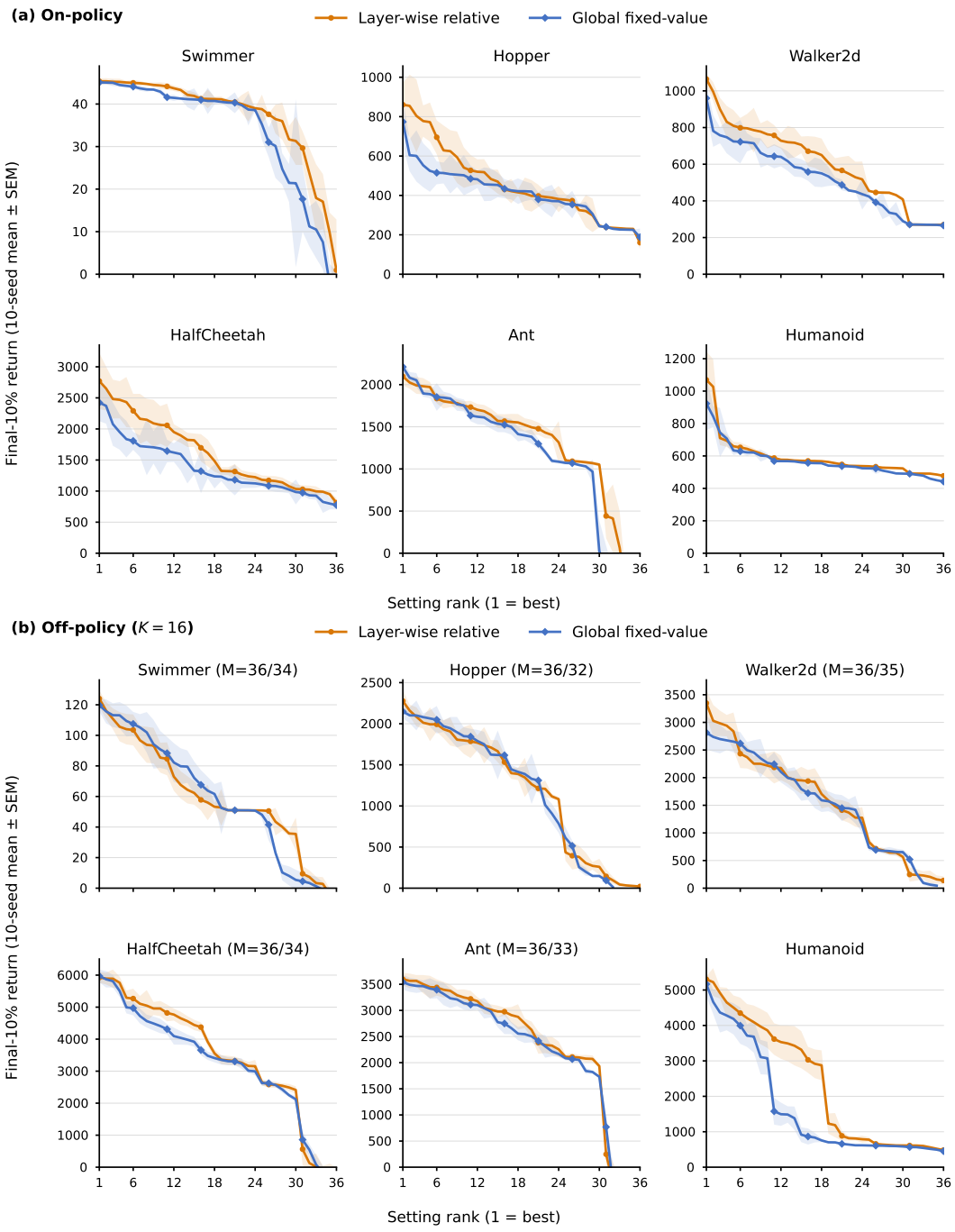

Figure 14: Layer-wise clipping is usually more robust on-policy, while the off-policy ordering is mixed. Setting-rank distributions compare layer-wise relative and global fixed-value direction clipping under (a) on-policy and (b) off-policy training with $K = 16$. Each geometry is ranked independently by mean final-10% return. All on-policy panels contain 36 complete settings per geometry, each with ten seeds. In the off-policy panels, M is shown in the title in the order layer-wise/fixed-value: Swimmer $36/34$, Hopper $36/32$, Walker2d $36/35$, HalfCheetah $36/34$, Ant $36/33$, and Humanoid $36/36$. Curves are ten-seed means and bands are seed-level standard errors; unavailable settings are excluded rather than assigned zero. These are tuning-set summaries, not held-out estimates.

F.7 ACTOR-UPDATE COST ACROSS TASK SHAPES AND WIDTHS

The return and mechanism comparisons leave one practical question: how costly are the five actor updates in Table 1 in their current implementations? Each cell contains a single, non-vmapped policy with a synthetic current-policy minibatch of size 64; rollout and critic work are excluded. Each method runs in a separate process on the same otherwise idle NVIDIA RTX PRO 4500 Blackwell Server Edition GPU, with widths $d \in \{64, 256, 1024\}$ and all six task shapes randomly interleaved within that process. Each cell receives 100 warm-up updates followed by 200 CUDA-event measurements, with the device synchronized around every complete actor update. An untimed fixed 4096×4096 general matrix multiplication (GEMM) primes the GPU clock once after warm-up and before the measurement sequence. All 90 method–width–task cells completed, and no timing samples were discarded. These scalar implementations use ordinary PyTorch reverse mode rather than per-example transforms: layer hooks expose the inputs needed by CD-NPG and K-FAC-NPG, and the CD-NPG score statistics and policy gradient share one reverse-mode pass. The timing-only CD-NPG implementation reuses parameter-shaped current-gradient tensors as the clipped direction and current-Fisher-diagonal tensors as within-update denominator scratch; neither is retained as an additional full-policy workspace across updates. For all five methods, the optimizer-specific post-backward operations are compiled as one full update graph. The CD-NPG graph includes its integrated-Fisher quadratic, the CG-NPG graph contains all 16 Fisher-vector products, and the K-FAC-NPG graph contains the EMA, damped factorizations, solves, normalization, and update. The synthetic damped K-FAC factors are positive definite, so this benchmark exercises the successful no-jitter path; the training backend retains its numerical jitter fallback.

Parameters, gradients, optimizer state, and update arithmetic are FP32 for all five methods, including Muon's five Newton–Schulz iterations. This uniform-precision benchmark excludes method-specific low-precision arithmetic and reduced update schedules: analogous approximations can accelerate other methods as well, and comparing their separate engineering frontiers is outside our scope. Adam + PG uses a global gradient-norm clipping threshold of 1. CD-NPG uses the evaluated layer-wise median rule with $c_{\mathrm{rel}} = 10$, integrated-Fisher normalization, and $\delta = 10^{-5}$. CG-NPG uses 16 iterations and damping 10^{-3}; K-FAC-NPG uses damping 10^{-3}. These fixed values define a kernel benchmark, not a time-to-target-return comparison.

Figure 15 shows a stable ordering across task shapes. Adam remains fastest, while CD-NPG stays close to it and grows only mildly with actor width. Muon becomes more expensive at the widest actors, and both CG-NPG and K-FAC-NPG remain substantially slower, with their gap from CD-NPG widening as width increases. These finite-width measurements characterize this device and implementation, not hardware-independent constants.

The benchmark uses Python 3.11.10, PyTorch 2.11.0+cu128, CUDA 12.8, cuDNN 9.19.0, NVIDIA driver 580.178.04, and compute capability 12.0.

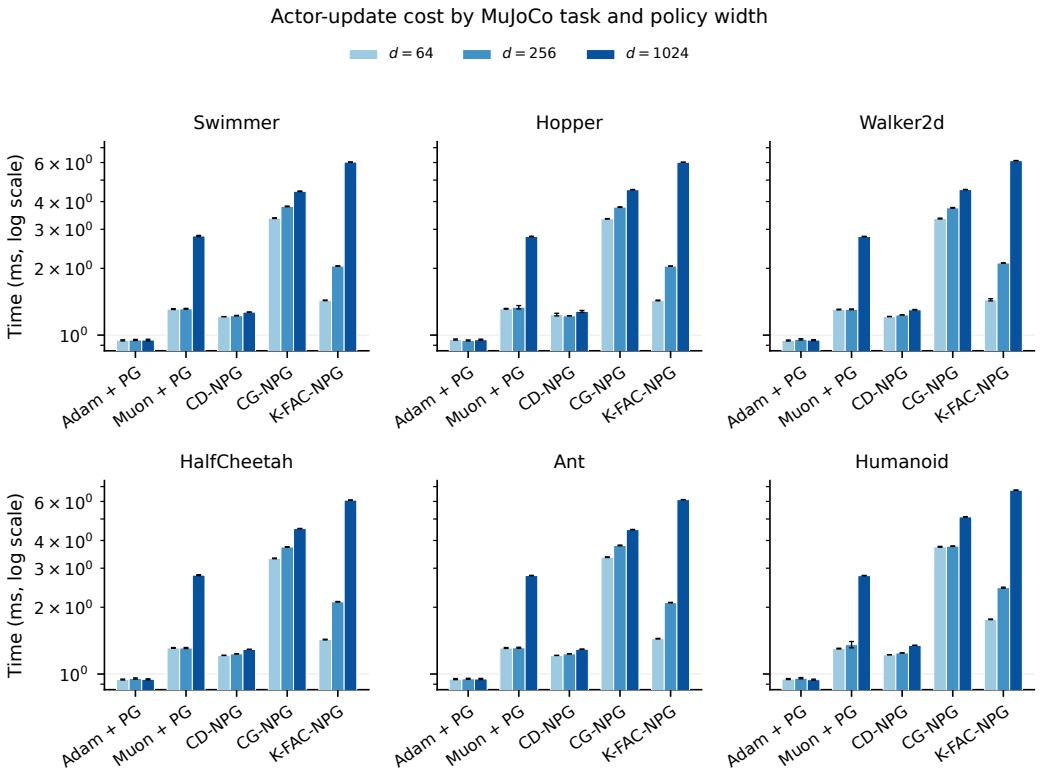

Figure 15: CD-NPG stays close to Adam's runtime and remains cheaper than the richer natural-gradient updates as width increases. Complete actor-update times are shown for all six MuJoCo policy shapes at $d \in \{64, 256, 1024\}$ and minibatch size 64 (log scale). Error bars are within-cell standard errors over the 200 synchronized measurements; the benchmark protocol and fixed settings are specified above.

Runtime is only one part of implementation cost. We therefore also record explicit tensor memory for every benchmark cell. Persistent memory includes the actor parameters and fixed policy buffers plus algorithm-state tensors retained across updates; reusable scratch tensors are not persistent. The peak estimate additionally includes the supplied actor batch and a manual source-level liveness upper bound for named explicit tensors in each update phase. CUDA-library workspaces, autograd-engine bookkeeping, allocator cache, and framework or module scaffolding are excluded. Table 3 reports the maximum over the six task shapes. Across all three widths, CD-NPG retains the same persistent footprint as Adam, because its additional score contractions, layer-wise order statistics, and output JVP are temporary. Its peak remains close to CG-NPG's and well below K-FAC-NPG's, whose stored Kronecker factors and within-update factorizations dominate at larger widths. The memory ordering therefore differs from the runtime ordering: CG-NPG has a small persistent state but pays for repeated products, whereas CD-NPG uses more persistent tensors but a much cheaper update.

Table 3: Maximum explicit tensor memory over the six MuJoCo task shapes, reported as persistent/peak MiB from the same benchmark. Persistent memory includes actor parameters and fixed policy buffers plus algorithm state retained across updates; peak additionally includes within-update explicit tensors.

Actor update	$d = 64$	$d = 256$	$d = 1024$
Adam + PG	0.34/1.11	1.91/5.82	16.63/49.98
Muon + PG	0.32/1.11	1.66/4.70	12.63/38.27
CD-NPG	0.34/1.21	1.91/4.75	16.63/43.83
CG-NPG	0.11/1.15	0.64/5.53	5.54/45.54
K-FAC-NPG	0.94/5.10	3.45/11.90	33.19/119.45

G Atari Results

Environment. We evaluate MsPacman, Pong, Q*bert, and Enduro from the EnvPool Atari v5 suite using the minimal action sets. Observations are 84×84 grayscale frames stacked over four frames, with a frame skip of four. Each episode begins with up to 30 random no-op actions. Sticky actions are disabled, losing a life does not end an episode, and episodes are capped at 27,000 steps. Rewards are not clipped.

Network and training. The policy and value networks are separate Nature-CNNs, each with three convolutional layers followed by a 512-unit fully connected layer; the policy network has approximately 1.69 million parameters. The value network is identical across methods and is trained with Adam using a learning rate of 2.5×10^{-4}, gradient-norm clipping at 0.5, and a value-loss coefficient of 0.5. The methods differ only in the policy-network update rule. In each iteration, 32 parallel environments each generate 128 steps, yielding 4,096 samples for a single policy update with one epoch and one minibatch. Training is therefore strictly on-policy and runs for 10 million environment steps.

Tuning protocol and evaluation metric. For each method and task, we run a 5×5 log-spaced grid search over the method's two hyperparameters, each spanning two orders of magnitude, with five seeds per configuration. This gives 125 runs per method–task pair and 2,500 runs across the four tasks. Grid centers are set separately for each method and task from pilot experiments, but every method uses the same number of grid points and the same span, keeping the tuning budget matched across methods. Final return is defined as the mean return over the last 10% of training steps.

Results. Figure 16 shows that K-FAC-NPG is separated from the other methods across much of each rank distribution, not only at the selected best configuration. This pattern is consistent with within-layer correlations being especially useful for convolutional policies. Among the lower-cost methods, the ordering is less uniform: CD-NPG generally lies above Adam and CG-NPG through substantial portions of the grids, while Muon often has a stronger best setting. On several tasks, however, CD-NPG degrades more gradually and overtakes Muon in the middle or tail of the rank curve. The Atari result therefore supports hyperparameter robustness rather than a universal second-place ordering.

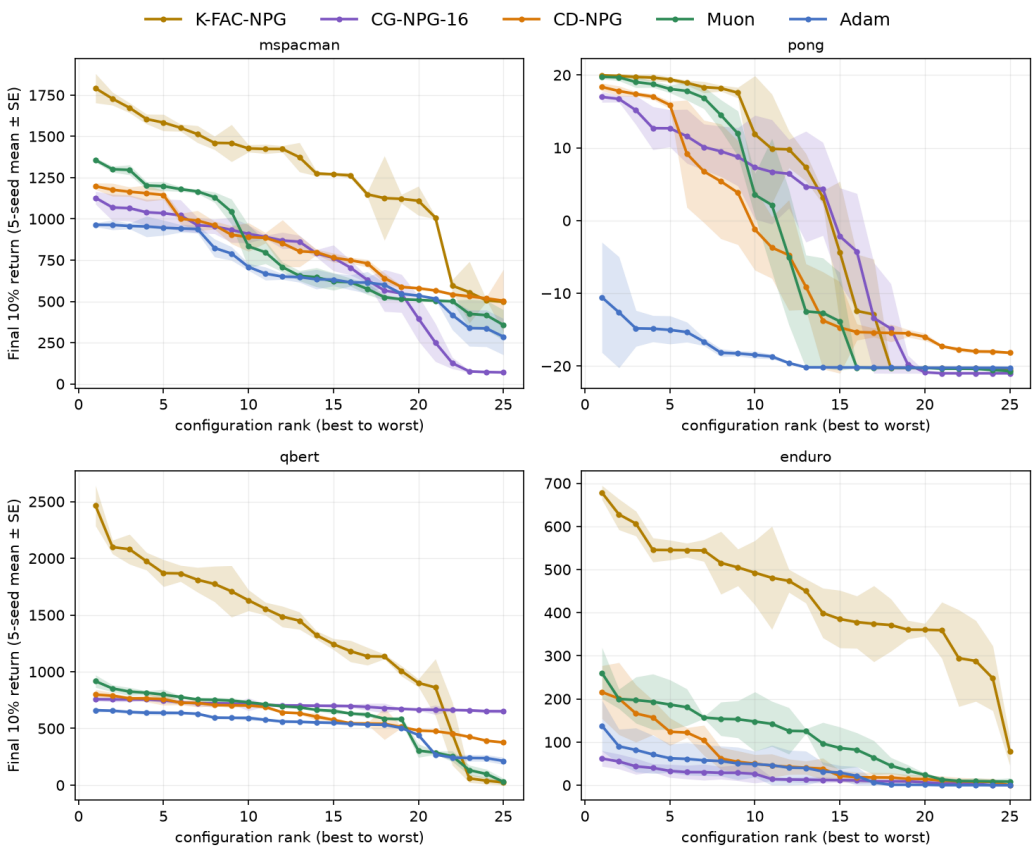

Figure 16: K-FAC-NPG dominates the Atari rank curves, while CD-NPG remains competitive and comparatively robust among the remaining methods. Configuration-rank results cover MsPacman-v5, Pong-v5, Q*bert-v5, and Enduro-v5. For each method and task, tested configurations are ordered from highest to lowest final-10% return. Curves show five-seed means and shaded bands show standard errors.