FORWARD SIMILARITY DOES NOT GUARANTEE TRAINING EQUIVALENCE: PRECISION PLACEMENT IN DIFFERENTIABLE GRAM SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Changing the numerical precision of a differentiable Gram solver may leave predictions nearly unchanged while altering the computation used to learn its parameters. We study this distinction in controlled Spectral Koopman Attention (SKA) and an independent episodic ridge classifier, assigning FP32 or FP64 to separate solver stages without changing their exact-arithmetic definitions. Matched comparisons fix the model, input, and optimizer state; separately trained endpoints are evaluated through a common precision route. For SKA, we compare q_{110} with the higher-precision solver route q_{111}, a numerical reference rather than mathematical ground truth. Across 18 matched SKA validation comparisons from three training seeds, median relative discrepancies in logits, full-model gradients, and restored AdamW updates are 0.962%, 8.61%, and 9.88%, respectively, each normalized by its corresponding q_{111} reference norm. Prediction agreement can therefore miss differences in the learning computation. In a seven-seed matched classifier comparison, imposing externally prescribed ridge schedules shared across routes within each seed during training reduces the absolute between-route spread in post-training regularized-Gram conditioning by 97.4% relative to learning ridge separately under each route. This shows that the observed conditioning separation depends on the training-time ridge policy. Precision choices in these solvers therefore warrant checks of parameter updates alongside predictions, with endpoint conditioning and task outcomes assessed separately.

1 INTRODUCTION

Nearly unchanged logits or validation loss can make a lower-precision solver appear interchangeable with a higher-precision implementation. A solver inside an end-to-end learned model, however, also supplies derivatives used by the optimizer (Amos & Kolter, 2017; Agrawal et al., 2019; Seeger et al., 2017). Prediction agreement leaves the derivative computation untested: similar outputs do not establish that the two implementations produce similar parameter updates.

Numerical sensitivity of solvers and their derivatives is established (Higham & Mary, 2022; Hovland & Hückelheim, 2024). For training-time precision substitutions, however, derivative sensitivity alone does not quantify optimizer-update discrepancies or differences retained after training. We therefore separate two questions: how much do precision settings change updates at fixed parameters, inputs, and optimizer state, and which learned-state differences remain under a common evaluation implementation? Matched parameter changes assess *local training equivalence*, with full-model gradients as an intermediate diagnostic. Evaluating trained models through a common solver implementation separates training-history differences from immediate evaluation arithmetic.

We study a language model using controlled Spectral Koopman Attention (SKA) (Sridhar & Johansen, 2026) and an independent episodic ridge classifier (Bertinetto et al., 2019). Both form a matrix from learned feature inner products, add positive regularization, and solve a linear system. We define a *precision route* by assigning FP32 or FP64 separately to statistic construction, factorization, and solution-side computation. Within each solver, these assignments preserve the exact-arithmetic map while changing its numerical implementation.

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.

Across 18 matched SKA validation comparisons from three training seeds, a mixed-precision route yields median relative discrepancies of 0.962% for logits, 8.61% for full-model gradients, and 9.88% for restored AdamW updates against a higher-precision solver reference. Each quantity uses its own reference norm. A fixed-checkpoint control uses the same 20 training batches on an Apple M4 Max CPU and an NVIDIA A100. Raising only factorization precision reduces gradient and update disagreement against each environment's reference, with little change in logit discrepancy. Recomputing local solver derivatives at higher precision also improves agreement while retaining the forward outputs (Section 4.3).

Equivalent reformulation provides a complementary implementation control. At one classifier checkpoint, a comparison using five fixed episodes yields smaller mixed-precision discrepancies with the smaller, mathematically equivalent *dual formulation* than with the primal formulation, each measured against its own higher-precision route (Section 4.4). The fixed-state result shows that the observed sensitivity also depends on the algebraic implementation.

After training, we examine ridge—the regularization strength—and regularized-Gram *conditioning*, a measure of linear-system sensitivity to perturbations. Common-route evaluation reveals regime-dependent endpoint separation relative to seed variation and differences in loss (Section 4.4). In a seven-seed classifier comparison, an externally prescribed ridge schedule is shared across routes within each seed, following that seed's reference trajectory with a one-episode lag. Sharing the schedule reduces absolute between-route spread in post-training conditioning by 97.4% relative to route-specific ridge learning. A schedule fixed during development lowers the route-averaged condition number without establishing better or equivalent task performance (Section 4.6).

Precision placement can change parameter updates despite close predictions, while the fixed-state dual control shows that equivalent reformulation can reduce the measured sensitivity. Separately trained endpoints exhibit regime-dependent differences, and ridge interventions change conditioning without establishing a task benefit. The experiments distinguish the effects of numerical implementation on updates, learned-state statistics, and task outcomes: prediction agreement alone does not validate a precision substitution for training.

2 Background

Let $Z \in \mathbb{R}^{n\times r}$ contain learned features as rows and let $C \in \mathbb{R}^{r\times k}$ be the right-hand side of a linear system. The regularized Gram solve obtains $W \in \mathbb{R}^{r\times k}$ from $G_\lambda W = C$, where $G_\lambda = Z^\top Z + \lambda I_r$ and $\lambda > 0$. We call λ the ridge parameter. In exact arithmetic, it makes G_λ positive definite, allowing Cholesky factorization and triangular solves. Higher-precision factorization cannot reconstruct information already lost during Gram construction. Precision must therefore be specified by computational role (Higham & Mary, 2022).

Differentiating the exact system gives

$$\mathrm{d}W = G_\lambda^{-1}\big(\mathrm{d}C - (\mathrm{d}G_\lambda)W\big),$$

where d denotes a differential and inverse notation denotes a solve. This identity is not a finite-precision error bound (Appendix C). The model gradient depends on the gradient from the downstream loss, and the parameter update depends on the optimizer's stored state. Forward agreement alone does not establish agreement of these computations (Hovland & Hückelheim, 2024).

3 Method

We define the solvers and precision interventions below. Matched update comparisons assess local training equivalence, with full-model gradients as an intermediate diagnostic. Forward similarity concerns downstream outputs, not uniform agreement of internal solver quantities. Evaluation and optimizer protocols are specified in Section 4.1.

3.1 Differentiable Gram Solver Abstraction and Precision Stages

Both systems use the regularized Gram solve in Section 2, with a lower-triangular Cholesky factor L. A *precision route* specifies the arithmetic assigned to three computational stages:

$$\mathsf{q}_{p_G p_C p_S}, \qquad p_G, p_C, p_S \in \{0, 1\}, \qquad 0 = \mathrm{FP32}, \quad 1 = \mathrm{FP64}. \tag{1}$$

Here p_G controls statistic construction, p_C the Cholesky input cast and factorization, and p_S solution-side computation, including triangular solves. Routes preserve the exact-arithmetic map within each formulation and use ordinary automatic differentiation; the backward substitution below is a separate diagnostic.

3.2 SKA Solver Instantiation

Our controlled SKA (Sridhar & Johansen, 2026) uses projected keys and queries $k_t, q_t \in \mathbb{R}^r$ and values $v_t \in \mathbb{R}^{d_h}$. The first $P < T$ positions form the prefix. Suppressing batch and head indices, K and V stack prefix keys and values as rows; K_- and K_+ stack keys at positions $1, \dots, P-1$ and $2, \dots, P$. Thus $M = K_+^\top K_-$ uses $P-1$ lag-one pairs, whereas $C_v = V^\top K$ and the Gram use all P prefix positions. Projection and stacking details are in Appendix A.8.

$$\begin{aligned} \lambda = \exp(\theta_\lambda), \qquad & G_\lambda = K^\top K + \lambda I_r = LL^\top, \\ A_w = L^{-1} M L^{-\top}, \quad & B_v = L^{-\top} L^{-1} C_v^\top = G_\lambda^{-1} K^\top V. \end{aligned} \tag{2}$$

Ridge is optimized jointly with the model. Inverse notation denotes triangular solves; no explicit inverse is formed. The prefix-derived L, A_w, B_v are reused at every position with fixed two-step propagation:

$$s_t = \begin{cases} k_t, & t \le P, \\ q_t, & t > P, \end{cases} \qquad u_t = L A_w^2 L^{-1} s_t. \tag{3}$$

Here q_t is a projected query, distinct from the route label q. Each head outputs $o_t^{(h)} = (B_v^{(h)})^\top u_t^{(h)}$, giving

$$y_t^{\mathrm{SKA}} = \sigma(\alpha) W_O \, \mathrm{Concat}_h \, o_t^{(h)}. \tag{4}$$

This branch includes W_O and the learned sigmoid gate, but excludes the vocabulary projection. Token embeddings pass directly through SKA and then that projection; cross-entropy uses suffix positions only. We use $A_{\mathrm{used}} = A_w$, with no spectral rescaling, downstream residual connection, additional sequence-model block, or MLP. The Gram, Cholesky, and solve path, including learned ridge, remains differentiable.

3.3 Independent Ridge Solver Instantiation

The episodic classifier uses labeled support examples to fit a linear classifier and query examples to evaluate it. A shared encoder produces their features:

$$\begin{aligned} Z_s = f_\phi(X_s), \quad Z_q = f_\phi(X_q), \quad & G_c = Z_s^\top Z_s + \lambda_c I, \\ C_c = Z_s^\top Y_s, \qquad W = G_c^{-1} C_c, \quad & \ell_q = Z_q W, \quad \lambda_c > 0. \end{aligned} \tag{5}$$

Here Y_s contains episode-local five-way one-hot targets, and ℓ_q denotes query logits, distinct from scalar loss $\mathcal{L}$. Query cross-entropy trains the encoder and ridge $\lambda_c = \exp(\theta_c)$ jointly through Cholesky and triangular solves. The encoder and feature normalization are specified in Appendix A.8.

Endpoint and policy studies retain this *primal* implementation. A fixed-state control uses the *dual* form $W_d = Z_s^\top (Z_s Z_s^\top + \lambda_c I_{25})^{-1} Y_s$ instead: a 25×25 rather than 64×64 solve, with the same exact-arithmetic W for $\lambda_c > 0$ (Appendix C.1). Table 1 describes the primal stages; the dual configuration is discussed in Appendix A.11.

3.4 Precision Routes and Solver-Associated Backward Diagnostic

Table 1 specifies which computations each bit controls.

Writing FP32/FP64 as 32/64, in stage order (p_G, p_C, p_S):

$$\begin{aligned} &\mathsf{q}_{000} : (32, 32, 32), \quad \mathsf{q}_{100} : (64, 32, 32), \quad \mathsf{q}_{101} : (64, 32, 64), \\ &\mathsf{q}_{110} : (64, 64, 32), \quad \mathsf{q}_{111} : (64, 64, 64). \end{aligned}$$

In SKA, p_S includes M, C_v construction as well as the associated solves; in the primal classifier, C_c construction belongs to p_G. For example, SKA's q_{100} forms the regularized Gram in FP64 but casts

Table 1: Precision stages for SKA and the primal ridge classifier. Roles align across systems, but tensor assignments are solver specific.

Stage	Computational role	SKA	Primal ridge classifier
p_G	Routed-statistic construction	Gram-path key casting, $K^\top K$, ridge addition	G_c, C_c construction
p_C	Cholesky factorization	Cholesky input and factorization	Cholesky input and factorization
p_S	Solution-side routed computation	M, C_v construction and transition/readout solves	Triangular solves constructing W

it to FP32 before Cholesky. Thus $\mathsf{q}_{100} \to \mathsf{q}_{110}$ changes only Cholesky-stage precision; $\mathsf{q}_{110} \to \mathsf{q}_{111}$ changes solution-side precision. The q_{101}–q_{111} comparison instead isolates Cholesky precision with both other stages in FP64. The operational reference q_{111} uses FP64 in the three solver stages, not throughout the network or in exact arithmetic. Independent calibration covers selected outputs and directions (Appendix A.7).

Solver-associated backward diagnostic. We retain the outputs emitted by q_{110} but replace how the loss gradient is propagated through the solver. Let Q collect projected queries, F_{sel} denote the q_{110} head-output map, F_{full} the map recomputed through q_{111}, and D the downstream model. The substituted vector–Jacobian product (VJP) is

$$\begin{aligned} z &= (K, V, Q, \lambda), \quad h = F_{\text{sel}}(z), \\ \bar{h} &= \nabla_h \mathcal{L}(D(h)), \quad \bar{z} = J_{F_{\text{full}}}(z)^\top \bar{h}. \end{aligned} \tag{6}$$

Here h is the per-head output before concatenation, W_O, and gating. The incoming loss gradient $\bar{h}$ comes from the retained forward. Automatic differentiation through the recomputed map supplies the VJP; the rest of the model differentiates normally. The replacement covers the full solver head-producing submap, not only Cholesky backward or the entire model. Detached inputs, reconstruction, casts, and gradient paths are detailed in Appendix A.3.

4 Experiments

We evaluate precision substitutions at two levels: the computation of a single update and the outcomes of separate training runs. Matched comparisons hold model parameters, inputs, and optimizer state fixed; separately trained models are evaluated through a common solver route. Ridge interventions then examine how regularization affects conditioning and whether learning it online benefits the task, treating these as separate questions.

4.1 Experimental Setup

Systems and regimes. SKA uses WikiText-2 and TinyStories with primary sequence/prefix lengths $512/256$, four heads, operator rank 128, and initial learned ridge 10^{-3}. WikiText held-out records are an internal division of the filtered training split, not the official validation split. The five-way digits classifier uses 25 support and 50 query examples with 64-dimensional features. Its baseline starts learned ridge at 10^{-3}; the *lower-ridge-initialization regime* changes only this initialization to 10^{-4}. Other settings remain fixed (Appendices A.9 and A.1).

Matched computations. Each matched precision-route comparison fixes model parameters, input, algebraic formulation, ridge, and restored optimizer state. For a routed quantity x_q and its matched reference $x_\star$, relative disagreement is

$$E(x_q; x_\star) = \frac{\| \operatorname{vec}(x_q) - \operatorname{vec}(x_\star) \|_2}{\max(\| \operatorname{vec}(x_\star) \|_2, 10^{-12})}. \tag{7}$$

We compare full-model raw gradients and actual restored-step parameter changes $\Delta\theta_q = \theta_q^+ - \theta$, under the stated optimizer and clipping protocols. Discrepancies are continuous measurements, not a universal equivalence threshold. Main tables display relative discrepancies as $100E$ (%). Each quantity uses its own reference norm; cross-space percentages are not amplification factors. Vector and loss construction and sample manifests are detailed in Appendices A.5 and A.4.

Common-route endpoints. Step-5000 endpoints trained under $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$ retain their learned parameters, including ridge, and are evaluated under common q_{111}. This fixes immediate evaluation arithmetic while preserving differences left by training. TinyStories uses seeds 0–2; WikiText and both classifier regimes use seeds 0–4. Each endpoint averages validation IDs 0–15; task CE then reports the across-seed mean and sample standard deviation within each training route. Matched differences and full reductions are in Appendix B.7.

Conditioning and route-to-seed separation. The *semantic condition* statistic measures regularized-Gram conditioning, not semantic quality: SKA uses $K_h^\top K_h + \lambda I$, and the classifier uses support-only $H_e = Z_s^\top Z_s + \lambda_c I$. SKA takes the linearly interpolated 95th percentile across four heads within each probe, then averages 16 probes; the classifier averages 16 episode conditions. Both use FP64 spectral measurements, with symmetrization and numerical floors specified in Appendix A.6. Because the classifier has 25 support examples and 64 features, $\mathrm{rank}(Z_s^\top Z_s) \leq 25 < 64$ and, in exact arithmetic, $\lambda_{\min}(H_e) = \lambda_c$ (Appendix C). Ridge and minimum eigenvalue are therefore not independent mechanism evidence.

We use $R_{\mathrm{route/seed}}(x;t) = \Delta_{\mathrm{route}}(x;t) / \max(\Delta_{\mathrm{seed}}(x;t), 10^{-12})$: the numerator averages absolute pairwise route differences within seeds; the denominator averages pairwise seed differences within routes. Both use raw-scale common-route statistics, including λ rather than $\log \lambda$. A large ratio can reflect a large route spread, a small seed spread, or both. It is descriptive, not a significance test or variance decomposition (Appendices A.6 and B.1).

4.2 Solver Fidelity Depends on Precision Placement

In the targeted 29-unit audit, each FP32-Gram route completes 21/29 units, with the same eight failing at Cholesky; each FP64-Gram route completes 29/29. These counts concern the specified forward, backward and restored-update path, not population failure rates or a certificate of finite values throughout (Appendix A.4). Table 2 separates Cholesky and solution-side interventions using unclipped restored AdamW updates.

In one failed unit, the symmetrized FP32 regularized Gram has $\lambda_{\min} \approx -1.5 \times 10^{-4}$: rebuilding from the same keys in FP64 restores positive definiteness, but FP64 Cholesky on the existing Gram does not. This example does not explain every failure. Stage effects are not additive, and successful factorization alone does not validate the learning update.

Table 2: Stagewise SKA comparisons on 29 frozen state–single-example units. Entries are median relative disagreements against q_{111}. Updates use checkpoint-restored AdamW.

Comparison	Changed role	N	Gradient	AdamW update
q_{101} vs. q_{111}	Cholesky	29	39.27%	41.02%
q_{110} vs. q_{111}	Solution side	29	13.45%	18.79%

4.3 Downstream Forward Similarity and Local Training-Step Disagreement

Cross-seed validation comparison. Six checkpoints from training seeds 1–3 at steps 100 and 1000, each crossed with three size-32 validation batches, give 18 matched q_{110}–q_{111} comparisons. All complete. Across these units, median logit, raw-gradient and clipped restored-AdamW update disagreements are 0.962%, 8.61% and 9.88%; median absolute loss difference is 3.19×10^{-4}. Gradient and clipped-update median cosines are 0.9961 and 0.9947: directions remain close despite vector disagreement. Table 3 gives six-unit medians for each seed; overall medians use all 18 units. These units come from three training seeds, not 18 independent training runs. Norm-1 clipping activates in both routes of one unit (Appendix A.5); auxiliary summaries and repeatability controls are in Appendix B.6.

Earlier seed-0 training-input comparisons with restored Adam likewise show close downstream outputs without a uniformly close internal A_w. Their nine-unit results and inactive-clipping follow-up are retained in Appendix B.8 and Figure 2. Different optimizers and inputs preclude attributing between-cohort magnitudes to seed alone.

Table 3: Cross-seed q_{110} versus q_{111} comparisons: six-unit medians per seed, in percent. Gradients precede clipping; restored AdamW updates include source-configuration norm-1 clipping.

Seed	Logit E (%)	Raw gradient E (%)	Clipped AdamW update E (%)
1	0.913	9.418	9.576
2	1.107	8.818	10.556
3	1.159	7.747	9.985

Fixed-checkpoint comparison across computing environments. Table 4 evaluates the same q_{111}-trained seed-0, step-1000 checkpoint and 20 saved training batches on Apple M4 Max CPU and NVIDIA A100-SXM4-40GB. This is an offline control at one saved training state, not additional training replication. Restoration and environment records are in Appendix A.10.

Table 4: Mean relative L_2 discrepancies in percent over the same 20 training batches, relative to each environment's q_{111}. Every execution restores model and Adam state. Norm-1 clipping is configured but inactive in both controls.

Environment	Route	Logit E (%)	Gradient E (%)	Update E (%)
Apple M4 Max CPU	q_{100}	1.65461	36.7520	41.5114
Apple M4 Max CPU	q_{110}	1.68202	14.0616	14.2051
A100-SXM4-40GB	q_{100}	2.74925	37.7789	42.1973
A100-SXM4-40GB	q_{110}	2.67472	17.9418	17.4266

Raising only Cholesky-stage precision ($\mathsf{q}_{100} \to \mathsf{q}_{110}$, $p_G = 1$, $p_S = 0$) reduces mean gradient/update discrepancy by 61.7%/65.8% on CPU and 52.5%/58.7% on A100, computed from the reported means. Reference-relative logit discrepancy changes little; the routes need not emit identical outputs. The pattern recurs in both environments, but hardware and software effects are not separated (Appendix B.11).

Separate custom-VJP diagnostic. On the same 29 frozen units as Table 2, the diagnostic retains q_{110} forward values and substitutes the solver-head Jacobian recomputed through q_{111} (Eq. (6)). Gradient and update agreement improve in all 29/29 cases (Table 5); this comparison uses unclipped restored Adam, not the stagewise audit's AdamW.

Table 5: Fixed-forward solver-derivative diagnostic on 29 frozen units. Entries are median relative disagreements against native q_{111}. Both columns emit the same q_{110} forward values; updates use unclipped checkpoint-restored Adam.

Quantity	Native q_{110}	Same forward + custom VJP
Full-model gradient E (%)	13.446	2.777
Restored Adam update E (%)	18.759	2.966

Improved agreement at unchanged prediction outputs exposes sensitivity in the differentiated solver pathway. The substitution covers the full solver-head map, so it neither isolates individual numerical error sources nor quantifies causal mediation (Appendix B.10).

On four fixed solver traces, selected outputs and one analytic directional derivative per trace show production q_{111} discrepancies of approximately 10^{-7}–10^{-6} against an 80-digit reference. Reference working-precision consistency is reported separately; these checks do not certify the full Jacobian, full-model gradients or optimizer updates (Appendix A.7).

4.4 Local Disagreement and Regime-Dependent Endpoint Separation

In baseline ridge, local q_{110}–q_{111} median gradient/update disagreements are 0.1821%/0.1099% across 36 matched units, alongside modest endpoint separation (Appendix B.9). These units are not independent training seeds, and the local and endpoint comparisons do not track one perturbation through training. Table 6 separately compares four trained regimes, with larger route-to-seed ratios in WikiText and the lower-ridge-initialization classifier. Local disagreement warrants a separate endpoint assessment; modest ratios do not establish model equality.

Table 6: Step-5000 endpoints under common q_{111} evaluation. Panel (a) gives raw-scale ridge and condition route-to-seed ratios. Panel (b) identifies training routes: each endpoint first averages IDs 0–15, then entries report mean ± sample SD across training seeds. Task CE is not a common cross-system performance scale; the two ridge regimes share the task and episode construction.

(a) Regularized-Gram endpoint separation

Regime	N_{seed}	R_λ	R_{cond}
TinyStories SKA	3	0.91	0.96
WikiText SKA	5	19.84	10.75
Ridge ($\lambda_0 = 10^{-3}$)	5	0.46	0.37
Ridge ($\lambda_0 = 10^{-4}$)	5	5.47	4.28

(b) Task cross-entropy by training route

Regime	N_{seed}	q_{100}	q_{110}	q_{111}
TinyStories SKA	3	2.259161 ± 0.006995	2.258135 ± 0.005344	2.258873 ± 0.005726
WikiText SKA	5	2.386080 ± 0.002150	2.386968 ± 0.001353	2.387161 ± 0.000976
Ridge ($\lambda_0 = 10^{-3}$)	5	0.889881 ± 0.015265	0.889756 ± 0.014979	0.889679 ± 0.014929
Ridge ($\lambda_0 = 10^{-4}$)	5	0.897302 ± 0.010427	0.889803 ± 0.013730	0.889559 ± 0.014046

Route-resolved task outcomes. In the lower-ridge-initialization classifier, the same-seed mean CE difference from q_{111} is $+0.007743$ for q_{100} (higher in 4/5 seeds), but only $+0.000245$ for q_{110}. WikiText instead has a q_{100} difference of -0.001081 (lower in 4/5 seeds). Thus q_{111} does not consistently attain the lowest observed task loss in these runs, and the classifier pattern does not extend to both selective routes. These observations establish neither statistical significance nor causation by local update discrepancies (Appendix B.7). Across four decreasing ridge initializations, local gradient/update discrepancy and R_λ increase, but R_{cond} peaks at 2×10^{-4} and then falls (Appendix B.2).

Algebraically equivalent ridge formulations. At one classifier checkpoint (q_{111}-trained, seed 0, step 5000; $\lambda_c \approx 1.4337369 \times 10^{-3}$), the CPU/Accelerate control compares 64×64 primal and 25×25 dual systems on validation episodes 0–4. All three measured discrepancies are smaller for dual (Table 7(a)); primal/dual update ratios are 26.0 and 8.9 for q_{100} and q_{110}.

Table 7: Fixed-state primal/dual control; all discrepancies are in percent. In (a), candidates use their own formulation's q_{111}. Logits are concatenated across five episodes; gradients and unclipped restored AdamW updates use one backward pass on the five-loss mean. In (b), dual-q_{111} is compared with primal-q_{111}: logit is the maximum episode error, whereas gradient/update use the single global five-loss comparison.

(a) Within-formulation precision discrepancies

Form	Route	Logit E (%)	Gradient E (%)	Update E (%)
Primal	q_{100}	0.11198	0.76070	0.12004
Dual	q_{100}	0.0052788	0.012503	0.0046255
Primal	q_{110}	0.047281	0.093731	0.021761
Dual	q_{110}	0.0017635	0.0049657	0.0024490

(b) Reference-implementation consistency

Observed E (%)	2.3819×10^{-5}	1.1740×10^{-4}	2.0194×10^{-4}

A standard equivalent formulation can therefore mitigate the observed local sensitivity without changing the mathematical ridge problem. The five specified reference-consistency criteria pass; the update observation has no prespecified threshold. These are implementation comparisons, not exact-error bounds; independent restore-repeatability and dual-trained endpoints are not established by this control (Appendices A.11 and B.12).

4.5 Ridge Dependence of Endpoint Statistics

Same-seed ridge transplantation. For seeds 0–4, step-5000 q_{100}/q_{110} recipients receive only their same-seed, step-5000 q_{111} donor's log-ridge. Non-ridge parameters remain fixed, no optimizer step occurs, and outputs are recomputed under common q_{111}. Closure is the fraction of the absolute

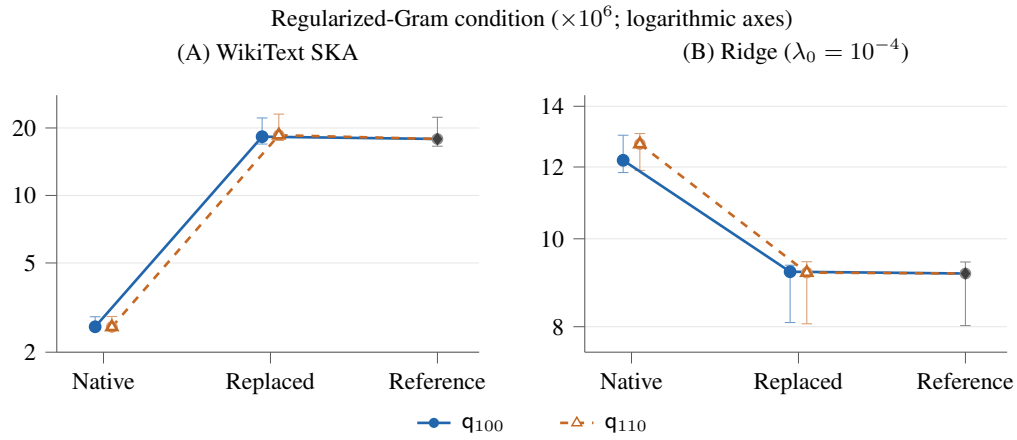

Figure 1: Endpoint ridge transplantation at step 5000. Native is the trained recipient; Replaced changes only its log-ridge to the same-seed q_{111}-trained endpoint's value; Reference is that endpoint. All configurations use q_{111} evaluation without optimization. Lines connect five-seed medians of means over IDs 0–15, not training trajectories; bars show min–max ranges, not confidence intervals. Replacement approaches the reference, increasing WikiText condition but decreasing ridge condition. Seed-level values and paired log-gap closures are in Appendix B.3.

log-conditioning gap to the reference closed by replacement. We report the median of five seed-level closures, computed from each endpoint's 16-evaluation mean (Appendix B.3).

The q_{100}/q_{110} closures are 98.8%/98.0% for WikiText and 98.1%/99.3% for the lower-ridge-initialization classifier (Figure 1). Approaching the reference raises WikiText condition but lowers classifier condition. This establishes strong conditioning dependence on ridge at fixed non-ridge parameters, not representation recovery or task improvement.

WikiText training-time ridge controls. With ridge externally fixed at 10^{-3} or replayed from the same seed's q_{111} trajectory, only $0/5$ and $1/5$ q_{100} runs complete, respectively; the remaining runs fail at Cholesky (Table 27 in Appendix B.14). Failures remain in the denominators. Ridge replay therefore does not resolve this training-time failure boundary or establish WikiText training-time ridge mediation, despite close endpoint agreement after replacement.

4.6 Training-Time Ridge Interventions

Policies use the classifier with $\lambda_0 = 10^{-4}$, seeds 5–11 and routes $q_{100}, q_{110}, q_{111}$. P1 learns log-ridge online. P2 replays each seed's saved P1–q_{111} trajectory across that seed's routes with the implementation's one-episode lag (Appendix B.13). P5 assigns one development-locked log-ridge schedule across all seeds and routes. P2/P5 do not optimize ridge. Initial ridge is shared, but subsequent P2 trajectories differ across seeds.

Matched seven-seed shared-ridge comparison. P1/P2 step-5000 endpoints use common q_{111} evaluation on identical episodes 0–15. Absolute condition route spread falls from 2,539,987.73 to 65,379.40 (97.4%), and R_{cond} from 5.6939 to 0.1384. Unlike frozen-endpoint replacement, P2 controls ridge throughout training while non-ridge parameters adapt under their assigned routes. This provides intervention evidence that the measured conditioning separation depends strongly on the training-time ridge policy; it does not decompose all route-dependent training effects.

Development-locked log-ridge schedule. P5 was constructed from development seeds 0–2 and locked before confirmatory training on seeds 5–11, without online route-specific ridge adaptation. Held-out refers to seeds relative to policy development, not an unseen dataset. Locked knots and interpolation are provided in Appendix B.5. Table 8 retains the distinct condition and task evaluators. P5 lowers route-averaged condition in 7/7 seeds, but query loss is lower in 4/7 and higher in 3/7 ($p = 0.375$). No equivalence margin was specified: the task comparison establishes neither benefit nor equivalence. Online route-specific ridge learning is not required for the observed lower route-averaged condition; these averages do not establish improvement on every route.

Table 8: P5 versus native learned ridge on seeds 5–11. Condition uses 16 episodes and query loss uses 200 within each route, followed by a three-route average under common q_{111}. Entries are seven-seed means; final columns count lower P5 seed values and report unadjusted exact two-sided paired sign-flip p-values with seed as the paired unit. The loss test is not an equivalence test. Paired seed data and 95% bootstrap intervals appear in Appendix B.5 (Tables 17 and 18).

Metric	Native ridge	P5	Seeds lower	Paired p
Semantic condition	1.131×10^7	9.159×10^6	7/7	0.015625
Validation query loss	0.900022	0.899382	4/7	0.375

5 Related Work

Differentiable solvers and learned linear operators. Differentiable optimization and linear-algebra layers embed structured solves in end-to-end learning (Amos & Kolter, 2017; Agrawal et al., 2019; Blondel et al., 2022; Ionescu et al., 2015; Seeger et al., 2017). Bertinetto et al. (2019) combine episodic ridge regression with learned features and solver hyperparameters, use Woodbury's identity for smaller solves, and examine the task value of learning ridge. Our classifier studies precision sensitivity in this construction. Our controlled SKA derives from Echo (Sridhar & Johansen, 2026), within work on finite-dimensional Koopman approximations and learned dynamical representations (Williams et al., 2015; Takeishi et al., 2017; Lusch et al., 2018).

Finite-precision numerical linear algebra. Cholesky-based QR analysis and mixed-precision solvers address conditioning, roundoff, and precision allocation (Yamamoto et al., 2015; Carson & Higham, 2018; Higham & Pranesh, 2021; Higham & Mary, 2022). Mixed-precision training uses higher-precision accumulation, master weights, loss scaling, and tensor-level assignments (Micikevicius et al., 2018; Kalamkar et al., 2019; Lee et al., 2023). We compare solver-stage precision choices through predictions, full-model gradients, and restored updates at matched parameters, inputs, and optimizer state.

Forward–backward discrepancies in training. Automatic differentiation faces numerical pitfalls, including incompletely converged iterative solvers (Hückelheim et al., 2024; Akbarzadeh et al., 2020). Murray (2016) derives Cholesky differentiation rules; Hovland & Hückelheim (2024) compare differentiating iterative solver implementations with differentiating the underlying equations, primarily in forward mode. Matrix-decomposition methods also adjust backward computations for stability (Wang et al., 2019; 2022; Song et al., 2021). Our fixed-forward diagnostic tests solver-derivative sensitivity. Common-route evaluation separates endpoint differences from evaluation arithmetic; ridge interventions distinguish conditioning effects from the task value of online adaptation.

6 Conclusion

Forward agreement alone does not establish local training equivalence in differentiable Gram solvers. Matched SKA comparisons, including three additional training seeds, show close predictions alongside different gradients and actual parameter updates. The stage-specific pattern also appears in the fixed-checkpoint CPU/A100 control, while recomputing solver derivatives improves agreement without changing prediction outputs. In the separate classifier control, a standard equivalent dual formulation substantially reduces the measured discrepancies: their magnitude is not fixed by the mathematical problem alone.

After training, ridge and conditioning separation vary across regimes. Sharing a reference ridge schedule during classifier training substantially reduces the measured between-route conditioning spread; lower conditioning under a separate locked schedule does not establish better or equivalent task performance. Precision substitutions therefore need update checks in addition to prediction checks, and trained models require separate evaluation of conditioning and task outcomes. These findings concern the two tested Gram solvers and their evaluated regimes.

AI USE STATEMENT

We used ChatGPT and Codex to assist with drafting and revision, literature search and comparison, and code and artifact inspection. The authors are responsible for the manuscript's claims, citations, and experimental analyses.

REPRODUCIBILITY STATEMENT

Section 3 specifies both solvers, the precision assignments, and the custom-VJP intervention. Section 4.1 defines the comparison populations, metrics, and endpoint evaluators. Appendix A records experiment-specific optimizer and retry protocols, derivative reconstruction, the original nine comparisons and their Adam clipping follow-up, the additional 18 comparisons with restored AdamW and source-configured clipping, and estimator details. Appendix A.2 reports recovered computing-environment records, their historical coverage, and the dependency-manifest limitation. Appendix B.6 gives the additional local summaries, and Appendix B.7 reports task outcomes and same-seed differences under a common evaluator. Appendix A.7 describes the four-trace production-reference checks and pre-cast precision consistency of the reference. Appendix B also provides the descriptive endpoint criterion, ridge-sweep aggregation, seed-level transplant values, supplementary ridge-control protocols, and the P5 log-ridge knots. Appendix C gives the exact-arithmetic identities used to interpret the solver and ridge statistics. Evaluation units and aggregation orders remain distinct, and failed runs are retained in the completion counts. Cross-environment and primal/dual protocols and results are provided in Appendices A.10, B.11, A.11, and B.12.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane T. Barratt, Stephen P. Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9558–9570, 2019.

Siamak Akbarzadeh, Jan Hückelheim, and Jens-Dominik Müller. Consistent treatment of incompletely converged iterative linear solvers in reverse-mode algorithmic differentiation. *Computational Optimization and Applications*, 77:597–616, 2020. doi: 10.1007/s10589-020-00214-x.

E. Alpaydin and C. Kaynak. Optical recognition of handwritten digits. UCI Machine Learning Repository, 1998. URL https://doi.org/10.24432/C50P49. Dataset.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 136–145, 2017. URL https://proceedings.mlr.press/v70/amos17a.html.

Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7.

Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5230–5242, 2022.

Erin Carson and Nicholas J. Higham. Accelerating the solution of linear systems by iterative refinement in three precisions. *SIAM Journal on Scientific Computing*, 40(2):A817–A847, 2018. doi: 10.1137/17M1140819.

Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023. URL https://arxiv.org/abs/2305.07759.

Nicholas J. Higham and Theo Mary. Mixed precision algorithms in numerical linear algebra. *Acta Numerica*, 31:347–414, 2022. doi: 10.1017/S0962492922000022.

Nicholas J. Higham and Srikara Pranesh. Exploiting lower precision arithmetic in solving symmetric positive definite linear systems and least squares problems. *SIAM Journal on Scientific Computing*, 43(1):A258–A277, 2021. doi: 10.1137/19M1298263.

Paul Hovland and Jan Hückelheim. Differentiating through linear solvers. *arXiv preprint arXiv:2404.17039*, 2024. URL https://arxiv.org/abs/2404.17039.

Jan Hückelheim, Harshitha Menon, William Moses, Bruce Christianson, Paul Hovland, and Laurent Hascoët. A taxonomy of automatic differentiation pitfalls. *WIREs Data Mining and Knowledge Discovery*, 14(6):e1555, 2024. doi: 10.1002/widm.1555.

Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2965–2973, 2015.

Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019. URL https://arxiv.org/abs/1905.12322.

Wonyeol Lee, Rahul Sharma, and Alex Aiken. Training with mixed-precision floating-point assignments. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=ZoXi7n54OB.

Bethany Lusch, J. Nathan Kutz, and Steven L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9:4950, 2018. doi: 10.1038/s41467-018-07210-0.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016. URL https://arxiv.org/abs/1609.07843.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

Iain Murray. Differentiation of the Cholesky decomposition. *arXiv preprint arXiv:1602.07527*, 2016. URL https://arxiv.org/abs/1602.07527.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12 (85):2825–2830, 2011. URL https://www.jmlr.org/papers/v12/pedregosa11a.html.

Matthias Seeger, Asmus Hetzel, Zhenwen Dai, Eric Meissner, and Neil D. Lawrence. Auto-differentiating linear algebra. *arXiv preprint arXiv:1710.08717*, 2017. URL https://arxiv.org/abs/1710.08717.

Yue Song, Nicu Sebe, and Wei Wang. Why approximate matrix square root outperforms accurate SVD in global covariance pooling? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1115–1123, 2021. doi: 10.1109/ICCV48922.2021.00115.

Anupama Sridhar and Alexander Johansen. Echo: KV-cache-free associative recall with spectral Koopman operators. In *Proceedings of the ACM Conference on AI and Agentic Systems*, pp. 964–989, 2026. doi: 10.1145/3786335.3813146.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. In *Advances in Neural Information Processing Systems*, volume 32, pp. 3156–3164, 2019.

Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Robust differentiable SVD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5472–5487, 2022. doi: 10.1109/TPAMI.2021.3072422.

Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data-driven approximation of the Koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015. doi: 10.1007/s00332-015-9258-5.

Yusaku Yamamoto, Yuji Nakatsukasa, Yuka Yanagisawa, and Takeshi Fukaya. Roundoff error analysis of the CholeskyQR2 algorithm. *Electronic Transactions on Numerical Analysis*, 44:306–326, 2015.

A Additional Experimental Details

A.1 Experiment-Specific Optimizers and Retry Behavior

The training script that produced the original nine-comparison cohort's WikiText checkpoints uses `torch.optim.Adam`, learning rate 10^{-3}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and global gradient-norm clipping at 1. It uses neither AMP/a gradient scaler nor a learning-rate scheduler. The offline comparisons in this original cohort restore the checkpoint parameters and copy the Adam moments and hyperparameters for each route. Their original updates omit clipping; the follow-up in Appendix A.5 explicitly executes the source configuration's clipping operation. The compared inputs need not be the next inputs in the source checkpoint's shuffled training stream, so these measurements are matched restored updates rather than a replay of the actual next training step. Here, "norm-1 clipping" denotes global L_2-norm clipping at threshold 1.

The additional 18-unit cohort uses source-trained and restored AdamW with learning rate 0.001, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and zero weight decay. The evaluator restores the model parameters and checkpoint optimizer state separately for each route, then applies backward, global norm-1 clipping, and the optimizer step in that order. Updates are the actual parameter differences after this step. All six checkpoints have one parameter group containing 12 registered parameters and 11 established optimizer-moment entries. Their established state counters are 100 or 1000, matching the checkpoint step. The parameter `ska.log_gamma` is outside the effective gradient path and has no optimizer state; it remains in the metric vector with zero actual update. Parameter names, order, and shapes agree between the two routes. These details apply to this additional cohort only.

The 29-unit stagewise diagnostic instead restores AdamW with weight decay explicitly zero. The custom-VJP diagnostic on those same units restores Adam. Both original diagnostics omit clipping. These optimizer identities are specific to the corresponding evaluators; they do not identify the optimizer for every historical SKA training run. Independent ridge training uses its separate AdamW configuration: learning rate 3×10^{-4}, zero weight decay, and no gradient clipping.

The relevant title-level source training implementation contains a 10^{-4} Cholesky retry branch. The corresponding reported checkpoint rows show no retry activation. The offline title-level route implementation has no retry, and all nine comparisons complete without factorization failure. This observation does not imply that every historical training step was retry-free.

Ridge-control initialization and assignment. In the seven-seed policy study, all 63 step-0 checkpoints across P1/P2/P5, routes $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$, and seeds 5–11 store FP32 log-ridge `-9.21034049987793`, giving effective initial ridge `9.99999901978299e-05`. Within each seed, the nine policy–route initial model states agree. P1 learns log-ridge online and includes all seven parameter tensors in AdamW; P2 and P5 freeze and exclude log-ridge, leaving six tensors. All step-0 optimizer states are empty. The common hyperparameters are learning rate 3×10^{-4}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and zero weight decay.

P2 uses the same seed's saved P1–q_{111} trajectory with the implementation's one-episode lag. For zero-based episode j, let $s[0]$ be the initial ridge and $s[j+1]$ the P1 ridge before episode j. P2

episode j uses $s[j]$; saving or evaluating after t training steps assigns $s[t]$. Thus for $j \geq 1$, training uses the preceding P1 episode's pre-forward ridge. Exact replay refers to these stored values, not synchronous episode alignment. P5 instead uses one development-locked schedule across seeds and routes (Appendix B.5).

Sharing ridge across routes within each seed makes $R_\lambda = 0$ by construction in the matched seven-seed comparison; it is not independent evidence of mechanism. This does not make the P2 trajectories identical across seeds.

The WikiText fixed-ridge control stores FP32 log-ridge `-6.907755374908447`, giving effective ridge `0.0009999999310821295`. Before each forward pass, the fixed value is reassigned. Log-ridge remains in the optimizer group; its gradient is computed and included in global norm-1 clipping, then only `log_ridge.grad` is set to `None` before the optimizer step. The assigned value is restored after the step. The replay control starts from the same value. Its same-seed q_{111} schedule entry $s[t]$ represents ridge after exactly t optimizer updates: update t uses $s[t-1]$ and restores $s[t]$ afterward. This control has no P2-style one-episode lag. These assignments control ridge itself and are separate from Cholesky retry or jitter.

Fixed-checkpoint CPU/A100 restored-step protocol. For the fixed-checkpoint SKA control, each route and input starts from the same saved model parameters and checkpoint Adam state. The reported update is the actual clipped restored parameter update, not a gradient scaled by a learning rate. The CPU record reports no clipping activation in all 60 route executions: every global gradient norm is below 1, so clipped and unclipped restored updates coincide. The subsequently supplied A100 CSV records 0/60 clipping activations, with coefficient 1 in every row and pre-clipping global norms between 0.2513375 and 0.5102349. The wrapper calls global L_2 clipping at 1 before the Adam step; the environment export records Adam with learning rate 10^{-3}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, zero weight decay, and `amsgrad=False`. The A100 clipped and unclipped updates come from separate restored executions; their recorded discrepancies differ slightly, so inactive clipping does not establish bitwise agreement between those executions. This offline control does not continue training or produce a new training trajectory; its optimizer and clipping statements are separate from the original nine-unit, additional 18-unit, and 29-unit protocols above.

A.2 Computing Environment and Numerical Settings

Table 9 separates code-requested settings from run-recorded platform and software information. Coverage differs across the experiments; task identifiers retain the correspondence to the original records.

Numerical configuration. The routed FP32/FP64 assignments are specified in Table 1. The title-level SKA records report autocast disabled, and the Task 13 record specifies FP32 as the default dtype. Task 21 preflights record no CUDA, but differ in MPS availability between the offline and confirmatory stages. The inspected construction path selects CPU; accelerator availability is not treated as evidence of accelerator execution. Optimizer processing, clipping, and retry behavior are described in Appendix A.1.

Historical coverage. The records do not fully establish CPU model, memory, historical BLAS/LAPACK backend, thread counts, matrix-multiplication precision, TF32 and deterministic-algorithm flags, or numerical environment variables. The original nine-pair evaluation and the clipping follow-up are separate executions; the table does not supply run-linked environment metadata for the follow-up. CPU FP64 conversion for gradient and update metrics does not identify the training device. A separate current snapshot reports Apple Accelerate BLAS/LAPACK, but does not establish the backend used in the historical runs.

Dependency records. The inspected `pyproject.toml` leaves dependencies unpinned and specifies Python < 3.14, whereas the later run records report Python 3.14.0. No resolved dependency lockfile was found in the inspected repository. The recorded interpreter therefore conflicts with the manifest's Python constraint. The versions above document preserved run records; they do not constitute a tested installation recipe or establish that the current manifest reconstructs those environments.

Table 9: Recovered environment information by experiment. "CPU requested" identifies a code configuration; other platform entries come from preserved run records. Unlisted fields remain unestablished. Task identifiers link the reported experiments to their original records.

Experiment group	Execution and platform evidence	Recorded software
Title-level SKA source training and original nine-pair evaluation	CPU requested	Historical software and platform versions not recovered.
Stagewise and custom-VJP SKA audits (Tasks 11 and 9; Appendices A.4 and A.3)	CPU requested	No run-linked environment export recovered.
TinyStories source runs	CPU requested	Python 3.14.0; PyTorch 2.11.0.
Independent ridge (Task 13; Section 3.3)	CPU; macOS 26.3; arm64	Python 3.14.0; PyTorch 2.11.0; NumPy 2.4.4; scikit-learn 1.8.0.
Five-seed shared-ridge evaluation (Task 19; Appendix B.4)	CPU; macOS 26.3; arm64	Python 3.14.0; PyTorch 2.11.0.
WikiText training-time ridge controls (Task 20; Table 27)	CPU; macOS 26.3; arm64	Python 3.14.0; PyTorch 2.11.0.
Seven-seed shared-ridge and locked-policy comparisons (Task 21; Section 4.6)	CPU requested by the construction path; macOS; arm64	Python 3.14.0; PyTorch 2.11.0; NumPy 2.4.4; SciPy 1.17.1.

The additional 18-unit evaluation has its own environment export dated 2026-09-11: CPU, macOS 26.3 arm64, Python 3.14.0, PyTorch 2.11.0, NumPy 2.4.4, pandas 3.0.2, and ten PyTorch threads. The separate pre-cast reference follow-up records CPU, the same listed OS and software versions, and mpmath 1.3.0. These exports document those offline executions, not earlier checkpoint training or the original nine-pair and 29-unit audits. The primal/dual control has an independent CPU/Accelerate export with macOS 26.3 arm64, Python 3.14.0, PyTorch 2.11.0, and NumPy 2.4.4.

Reported environments for the new offline control. The recovered CPU environment export identifies Apple M4 Max, arm64, macOS 26.3, Python 3.14.0, PyTorch 2.11.0, no CUDA, and Apple Accelerate BLAS/LAPACK. The subsequently supplied A100 environment JSON is exported by the evaluator at the end of the run recorded on 2026-09-14. It identifies NVIDIA A100-SXM4-40GB with 40 GB memory, driver 580.82.07, device `cuda:0`, and MIG `Disabled`. The `nvidia-smi` CUDA field is 13.0, whereas PyTorch is 2.11.0+cu128 with CUDA build 12.8; the raw cuDNN version field is 91900. The displayed CUDA field is not the PyTorch build version or a kernel-precision specification. The host record specifies Linux kernel `6.6.122+`, `x86_64`, glibc 2.39, and Python 3.13.15.

For this A100 execution, the export records default dtype FP32 and FP32 matrix-multiplication precision `highest`. CUDA matmul TF32 is disabled, whereas cuDNN TF32 is enabled; cuDNN benchmark, cuDNN deterministic mode, deterministic algorithms, and deterministic warn-only mode are all disabled. The linear-algebra backend preference is `_LinalgBackend.Default`. The supplied wrapper contains no autocast or gradient-scaler call and declares `amp_or_scaler=false` in its export. Recovered dependency candidates permit static inspection of the model path, but are not a complete historical import manifest. The A100 host CPU model and NumPy version, and both executions' full thread configurations, remain unestablished. These two exports do not establish the environments of earlier training, the primal/dual control, or the other offline evaluations above.

The new A100 CSV contains 60 unique completed route–batch rows, covering routes $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$ and batch IDs 0–19. Its log records all 20 batch completions and a final 60-row message. The original summary and manifest nevertheless retain `CPU COMPLETE; A100 PENDING EXTERNAL A100 EXECUTION`. The supplied script initializes this string and changes it to the both-complete status only when the same CSV contains both CPU and A100 rows; this CSV contains only A100 rows. The retained status therefore does not negate A100 completion. The sepa-

Table 10: Compact manifest of the original nine-comparison cohort. Every row contributes three matched q_{110}–q_{111} comparisons, each with batch size 32. The clipping follow-up reuses this entire set.

Checkpoint source	Seed	Step	Training-loader batch IDs
q_{111}-trained WikiText	0	100	0, 9, 19
q_{111}-trained WikiText	0	1000	0, 9, 19
Original-baseline WikiText	0	100	0, 9, 19

rately recovered CPU CSV and log establish its own 60 completed route–batch rows. Original files and status strings are preserved; checking these saved artifacts is distinct from a new execution.

A.3 Solver Derivative Reconstruction

The custom backward in Eq. (6) saves detached K, V, Q, λ and reconstructs the complete map to the per-head solver output using the reference solver-stage route. In backward it rebuilds this FP64 solver graph and calls `autograd.grad`; it is distinct from the earlier hand-written VJP feasibility implementation. This reconstruction includes Gram construction and ridge addition, both cross-moments, Cholesky factorization, operator and readout solves, whitening, fixed two-step propagation, unwhitening, value readout, and the required route output casts. The forward values emitted to the downstream model remain those of the selective route. The downstream graph supplies the incoming cotangent through ordinary differentiation; the substituted Jacobian is confined to this head-producing submap.

The head-producing submap ends before head concatenation, W_O, and gating, so its output h differs from the SKA branch output in Eq. (4). The incoming cotangent follows ordinary differentiation through head concatenation, W_O, the sigmoid gate, vocabulary projection and suffix-only cross-entropy.

A.4 Local Comparison Populations

The stagewise and custom-VJP evaluations share 29 fixed state–single-example units: 21 WikiText and eight TinyStories units from 14 source run/seed trajectories. Several trajectories contribute more than one checkpoint. The targeted historical categories comprise captured WikiText FP32-Gram failures, boundary failures, pre-failure checkpoints, healthy FP64-solver-trained WikiText checkpoints, and healthy TinyStories checkpoints. The configurations include both $r = 128, T = 512$ and $r = 256, T = 256$. These fixed single-example inputs differ from the batch-size-32 title-level inputs. The manifest was fixed before the grouped factorial analysis, but the historical categories already existed; this does not establish selection before all result inspection.

For this audit, completion means exception-free execution of the routed forward computation and loss, backward pass, gradient extraction, restored optimizer step, update extraction, and post-step diagnostics. The implementation does not separately apply explicit finite-value gates to every intermediate tensor. The same eight units terminate at Cholesky for each FP32-Gram route; each such route completes 21/29 units, whereas every FP64-Gram route completes 29/29. These are within-audit counts rather than estimated failure probabilities.

Table 10 specifies the separate nine-comparison population. Each checkpoint is crossed with all three listed non-shuffled training-loader batches. The three checkpoints are script-fixed and use seed 0; neither the checkpoints nor the inputs constitute a random or held-out sample. There is no exact checkpoint-and-input overlap with the 29-unit audit.

The separate baseline ridge local evaluation contains 36 selected state–episode entries from nine pilot q_{111}-trained checkpoints: seeds 0–2, each at steps 100, 2000, and 5000. At each checkpoint, 200 validation episodes are sorted by $(\kappa, \text{episode ID})$; quantiles $u \in \{0, 0.5, 0.95, 1\}$ select zero-based ranks `int(round(`$u(200-1)$`))`. The same checkpoint–episode pairs are reused across all eight routes. The full 200-item rankings were not separately saved. These pilot states differ from the non-pilot primal/dual checkpoint, and the 36 entries are not independent training replicates. The corresponding TinyStories local-result files contain headers only, and the conditional Stage 3

Table 11: Original local measurements and the clipping follow-up on the same nine title-level comparisons. Disagreements are raw relative fractions, not percentages; ranges are observed minima and maxima.

Quantity	Median E	Minimum E	Maximum E	Median cosine
Raw gradient	0.0561775	0.0488635	0.134304	0.998424
Original restored update	0.0585428	0.0547923	0.138429	0.998300
Restored update with clipping	0.0585428	0.0547923	0.138429	0.998300

transfer analysis was not carried out. They do not support four completed local units. This branch is separate from the eight TinyStories units in the 29-unit audit and from the long-run endpoints.

The additional cross-seed comparison contains 18 matched validation units from source seeds 1–3: each seed contributes checkpoints at steps 100 and 1000, and each checkpoint is crossed with three validation batches of size 32. Thus there are six q_{111}-trained checkpoints and three source training seeds, rather than 18 independent training runs. The recovered source records identify Task 12 training and the Task 19 redirection of seed 3 to a new checkpoint directory while calling the same `task12.train_run` entry point. The evaluator uses the first three complete non-shuffled batches of `texts[8000:9000]`, the internal validation partition of filtered WikiText training text. These are batch IDs 0–2, with matching input hashes across seeds and steps; they are not the official WikiText validation split. These units are reported separately from the original nine comparisons and the 29-unit audit.

A.5 Metric Construction and Clipping Follow-Up

For the original nine-comparison SKA diagnostic, let $\mathcal{P}$ index all registered model-parameter tensors in registration order. Define

$$\begin{aligned} g_q &= \mathrm{concat}_{i\in\mathcal{P}}\,\mathrm{vec}[\mathrm{FP64}(\nabla_{\theta_i}\mathcal{L}_q)]\,, \\ \Delta\theta_q &= \mathrm{concat}_{i\in\mathcal{P}}\,\mathrm{vec}\big[\mathrm{FP64}\big(\theta^+_{i,q}-\theta_i\big)\big]\,. \end{aligned} \tag{8}$$

Here $\theta^+_{i,q}$ is the parameter after the restored optimizer step. A zero tensor substitutes for an absent gradient, and zero updates remain in the update vector. FP64 conversion is for metric construction, not whole-network FP64 gradient computation. Relative disagreement and cosine similarity use these same vectors.

In this original diagnostic, the parameter difference is formed in the parameter dtype before FP64 conversion. The baseline ridge and primal/dual controls likewise subtract FP32 parameters before converting the difference to FP64. The additional 18-unit SKA evaluation instead uses $\mathrm{FP64}(\theta^+_{i,q}) - \mathrm{FP64}(\theta_i)$ inside the update vector, converting each endpoint before subtraction. These cohort-specific metric implementations are not interchangeable descriptions of rounding.

For the original nine comparisons, the branch comparison uses the full $32 \times 512 \times 256$ tensor, including every sequence position, W_O, and the sigmoid gate. It excludes the vocabulary projection. Loss is mean cross-entropy over the 32×256 suffix tokens, with signed difference

$$\delta\mathcal{L} = \mathcal{L}_{\mathsf{q}_{110}} - \mathcal{L}_{\mathsf{q}_{111}}. \tag{9}$$

The clipping follow-up evaluates exactly the nine pairs in Table 10. All 18 route evaluations have pre-clipping global gradient norm below 1. The norm-1 clipping operation is executed, every clipping coefficient is 1.0, and every optimizer step completes. Clipped and unclipped restored updates are identical in every pair. Table 11 gives the reported raw-fraction summaries; the local main-text percentages round these relative fractions. This check establishes inactive clipping on these nine comparisons only.

For the additional 18-unit cohort, gradients are measured before clipping. The main update measurement is the actual restored AdamW parameter change after the source configuration's norm-1 clipping; unclipped updates are an auxiliary diagnostic. Clipping activates only for seed 3, step 100, validation batch 0, in both compared routes (Table 12). This is one matched unit and two route executions. Its relative update disagreement is 0.062974 without clipping and 0.061128 with clipping.

Table 12: The only clipping-active matched unit in the additional 18-unit cohort: seed 3, step 100, validation batch 0. Norms are global parameter-gradient norms before and after norm-1 clipping. Both route executions activate clipping.

Route	Pre-clipping norm	Post-clipping norm	Coefficient
q_{110}	1.054906	0.999999	0.947951
q_{111}	1.050154	0.999999	0.952240

Both routes remain below the clipping threshold in the other 17 units, where clipped and unclipped updates coincide. Six same-route repeatability controls have zero checked differences in loss, gradients, and updates; this control does not establish absolute determinism of every backend operation. Logit disagreement in this cohort covers all 32 examples, all 512 sequence positions, and the full vocabulary; the loss averages cross-entropy over suffix positions only (zero-based indices 256–511).

A.6 Endpoint Reductions

For SKA, each probe and head define an FP64 regularized Gram matrix

$$H_h = K_h^\top K_h + \lambda I, \qquad \widetilde{H}_h = \tfrac{1}{2}(H_h + H_h^\top). \tag{10}$$

For each head, the condition statistic is

$$\kappa_h = \frac{\lambda_{\max}(\widetilde{H}_h)}{\max(\lambda_{\min}(\widetilde{H}_h), 10^{-30})}. \tag{11}$$

Each probe's condition statistic is the linearly interpolated 95th percentile across its four head conditions. If the sorted head conditions are $\kappa_{(1)}, \ldots, \kappa_{(4)}$, this is $0.15\kappa_{(3)} + 0.85\kappa_{(4)}$. Endpoint semantic condition is the arithmetic mean of these probe statistics over 16 fixed single-sequence evaluations, not a percentile across probes. The minimum-eigenvalue statistic takes the minimum across heads within each probe and then the mean over these 16 probes.

For independent ridge, each validation episode defines

$$H_e = Z_s^\top Z_s + \lambda_c I, \qquad \kappa_e = \frac{\lambda_{\max}(H_e)}{\max(\lambda_{\min}(H_e), 10^{-30})}, \tag{12}$$

with H_e constructed in FP64. Only support embeddings enter this matrix. The standard endpoint condition statistic is the arithmetic mean of κ_e over fixed episodes 0–15. A separate full-validation evaluator uses the 95th percentile across all 200 validation episodes. We keep these evaluation sets and aggregations distinct.

For the four-regime task comparison, each step-5000 endpoint is evaluated under common q_{111} execution, and its task CE first averages evaluation IDs 0–15. SKA uses only the first sequence of each selected validation batch and suffix-only loss; ridge uses the 16 validation episodes. This is not full-corpus loss or the 200-episode evaluator. Within each training route, the endpoint values are then summarized by their mean and sample standard deviation across training seeds, using the $N_{\text{seed}} - 1$ denominator. Same-seed route differences are formed before computing their across-seed mean and sample standard deviation (Appendix B.7). The grand task mean, which equally weights route–seed endpoint means, is retained only to reconcile the earlier aggregate results. The different task objectives are not a common cross-system performance scale.

Route-to-seed separation. Let $\mathcal{Q} = \{\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}\}$ and let $\mathcal{S}$ be the matched seed set, with $|\mathcal{S}| \geq 2$. Write $x_{q,s,t}$ for the common-route scalar statistic of route q, seed s and training step t, after averaging the 16 probe or episode values where applicable. We define

$$\begin{aligned}
\Delta_{\text{route}}(x;t) &= \frac{1}{|\mathcal{S}|\binom{3}{2}} \sum_{s\in\mathcal{S}} \sum_{\{q,q'\}\in\binom{\mathcal{Q}}{2}} |x_{q,s,t} - x_{q',s,t}|, \\
\Delta_{\text{seed}}(x;t) &= \frac{1}{3\binom{|\mathcal{S}|}{2}} \sum_{q\in\mathcal{Q}} \sum_{\{s,s'\}\in\binom{\mathcal{S}}{2}} |x_{q,s,t} - x_{q,s',t}|, \\
R_{\text{route/seed}}(x;t) &= \frac{\Delta_{\text{route}}(x;t)}{\max(\Delta_{\text{seed}}(x;t), 10^{-12})}.
\end{aligned} \tag{13}$$

Table 13: Pre-cast consistency of the 50- and 80-digit references on the four fixed traces. Output differences use the Euclidean norm; directional differences concern the scalar analytic derivative for the recorded direction and cotangent. Relative differences normalize by the 80-digit output norm or absolute directional value, respectively. All differences are evaluated at 120 decimal digits before binary64 conversion. The classifier labels identify initialization regimes.

	(a) Output consistency		(b) Analytic directional consistency	
Trace	Absolute	Relative	Absolute	Relative
SKA validation batch 0	1.560606×10^{-48}	1.189587×10^{-48}	2.787950×10^{-47}	2.592224×10^{-49}
SKA validation batch 1	2.420707×10^{-48}	2.697793×10^{-49}	4.572866×10^{-47}	2.764588×10^{-49}
Classifier, $\lambda_0 = 10^{-3}$	6.746417×10^{-49}	4.017353×10^{-50}	8.102818×10^{-50}	1.740314×10^{-49}
Classifier, $\lambda_0 = 10^{-4}$	3.724996×10^{-48}	2.268016×10^{-49}	2.417442×10^{-48}	1.846631×10^{-47}

The inner sums range over unordered pairs. Both spreads are raw-scale mean pairwise absolute differences; learned-ridge comparisons use λ, not $\log \lambda$. A large ratio can reflect a large route spread, a small seed spread, or both. It is descriptive, not a significance test or variance decomposition. The secondary four-part route-dominance criterion is documented in Appendix B.1; the main analysis uses continuous ratios and, for the matched intervention, absolute route spreads.

A.7 Independent Reference Calibration

Inputs and checked quantities. The calibration uses four fixed solver-input traces: SKA seed 1, step 100, validation batches 0 and 1, and seed-0, step-5000 validation episode 0 from each classifier regime initialized at $\lambda_0 = 10^{-3}$ and 10^{-4}. These initialization values identify the regimes; the references use each saved trace's actual ridge value. For SKA, the checked quantity is the first head's 64-dimensional first query output for example 0, not a prefix output or full-model logits. Each reference uses the saved inputs and mathematical map, with one recorded direction and cotangent per trace.

Production-route agreement with the independent reference. The corrected production-calibration results compare q_{111} with the 80-digit reference. Reported forward relative discrepancies are approximately 1.27×10^{-7}–9.38×10^{-7} for the two SKA traces, and approximately 2.30×10^{-7} and 2.48×10^{-7} for the two classifier traces. The checked directional-derivative relative discrepancies are approximately 10^{-7}–10^{-6}. These summary values do not resolve the contribution of interface casts separately from solver computation, and are not attributed solely to FP32 conversion. They compare production execution with its independent reference, rather than two precision routes or optimizer updates.

Precision consistency of the reference. The reference is computed separately at 50 and 80 decimal digits of working precision, and the resulting arbitrary-precision outputs and analytic directional derivatives are compared at 120 digits before any Python or NumPy binary64 conversion. Table 13 reports these reference-to-reference differences; their maximum relative output and directional differences are approximately 1.19×10^{-48} and 1.85×10^{-47}, respectively. The earlier zero differences described agreement after binary64 conversion; the pre-cast differences are nonzero. This checks the working-precision stability of the same `mpmath` implementation, separately from the production-route comparison above. These pre-cast relative differences have no denominator floor and are undefined for a zero reference denominator.

Scope and provenance. These checks cover four fixed inputs, one direction and cotangent per input, and the selected solver outputs and analytic directional derivatives. They do not establish strict error bounds, full-Jacobian or full-model gradient and update accuracy, or task optimality of q_{111}. The corrected four-trace v2 results are the direct production-calibration artifacts; the canonical table is a derived copy. Seven earlier attempts (three SKA and four ridge records) completed but were archived and excluded following one query-index correction. They are invalid earlier outputs, not seven solver exceptions or seven independent bugs. The corrected SKA index is

`heads[0,0,K.shape[0]]`. The pre-cast follow-up recomputed only the high-precision references, without rerunning the cross-seed comparisons, endpoint evaluations, or production q_{111} computations.

Reference evidence coverage. *Production* q_{111} *versus mp80:* four fixed solver-input traces; selected outputs and one analytic directional derivative with its recorded direction and cotangent per trace. This supports numerical comparisons on those inputs, outputs, and directions, not certification of the full Jacobian, full-model gradient, or optimizer update, nor a matched reference-error bound for the five-episode dual control. *mp50 versus mp80:* the same four traces; arbitrary-precision outputs and analytic directional derivatives compared pre-cast at 120 digits. This checks the reference's working-precision consistency, not production q_{111} accuracy at 10^{-48}. *Primal-*q_{111} *versus dual-*q_{111}*:* the five-episode control; maxima of per-episode internal W, logit and loss discrepancies, and one global gradient and restored-update comparison after averaging the five losses. The first five gate criteria pass; the observed update relative discrepancy is 2.019428×10^{-6}, with no gate threshold or independent restore-repeatability record. This checks consistency between implementations, not error against mathematical truth. See Table 26 and Appendix A.11.

A.8 Solver Construction Details

Shared regularized Gram form. The exact-arithmetic Gram system underlying Section 3.1 is

$$G_\lambda = Z^\top Z + \lambda I = LL^\top, \qquad \lambda > 0, \tag{14}$$

Here L is the lower-triangular Cholesky factor. A finite-precision construction may fail to be positive definite; solver-specific quantities are obtained through differentiable triangular solves.

SKA projections and prefix construction. Let $x_t \in \mathbb{R}^d$ denote a sequence representation supplied to the SKA module. For head h,

$$k_t^{(h)} = W_K^{(h)} \,\mathrm{LN}(x_t) \in \mathbb{R}^r, \quad q_t^{(h)} = W_Q^{(h)} \,\mathrm{LN}(x_t) \in \mathbb{R}^r, \quad v_t^{(h)} = W_V^{(h)} \,\mathrm{LN}(x_t) \in \mathbb{R}^{d_h}.$$

Here r and d_h are the per-head operator and value dimensions; $q_t^{(h)}$ denotes a projected query, distinct from the precision-route label q.

For prefix length $P < T$, suppressing head and batch indices, define

$$K = [k_1, \ldots, k_P]^\top, \quad V = [v_1, \ldots, v_P]^\top, \quad K_- = [k_1, \ldots, k_{P-1}]^\top,$$
$$K_+ = [k_2, \ldots, k_P]^\top, \quad M = K_+^\top K_-, \qquad C_v = V^\top K.$$

The Gram matrix uses all P prefix keys, whereas M uses the $P - 1$ observed lag-one pairs.

Independent classifier encoder. The encoder maps 64-dimensional inputs through a 128-unit linear layer, GELU, layer normalization, and a 64-unit output layer. Output features are RMS-normalized per example.

A.9 Training Configuration

The controlled character-language-model SKA instantiation uses sequence length $T = 512$, prefix length $P = 256$, embedding dimension $d = 256$, four heads, value dimension $d_h = 64$ and per-head operator dimension $r = 128$. WikiText-2 (Merity et al., 2016) uses 8,000 training records and 1,000 held-out records drawn from the filtered training split; the latter are not the official WikiText validation split. TinyStories (Eldan & Li, 2023) uses 8,000 training and 1,000 validation records. Training uses batch size 32 for at most 5,000 steps, learning rate 10^{-3} and zero weight decay. Learned ridge is initialized at 10^{-3} and the initial sigmoid gate is approximately 0.9. The source and diagnostic optimizer protocols, clipping, and retry behavior are specified separately in Appendix A.1.

The independent ridge classifier uses scikit-learn digits (Pedregosa et al., 2011; Alpaydin & Kaynak, 1998): flattened 8×8 images with pixels divided by 16 and a stratified 60/20/20 train/validation/test split. Each episode draws five of the ten classes, with five support and ten query examples per class, giving 25 support and 50 query examples. Training uses one deterministic episode per step for 5,000

steps, AdamW with learning rate 3×10^{-4}, zero weight decay and no gradient clipping. The baseline initializes learned ridge at 10^{-3}. The *lower-ridge-initialization regime* ($\lambda_0 = 10^{-4}$) changes only this initialization; ridge remains trainable. Architecture, feature dimension, data manifests, episodes, optimizer, learning rate, training duration, routes and seeds are otherwise fixed.

A.10 Fixed-Checkpoint Cross-Environment Protocol

The SKA control uses one q_{111}-trained checkpoint at seed 0, step 1000, with operator rank 128, sequence length 512, prefix length 256, and batch size 32. Its learned ridge is `0.001112196064965833`; this is the saved checkpoint value, not an initialization or a ridge held fixed throughout training. The saved input bundle contains batch IDs 0–19; both x and y have shape $(32, 512)$ and dtype `int64`. The recovered loader uses the `wikitext-2-raw-v1` train split, retaining texts whose stripped length exceeds 10. Its vocabulary uses the 50 most frequent characters in the first 5,000 retained texts plus two special tokens. Training uses `texts[:8000]`; internal validation uses `texts[8000:9000]`. Sequences use length $T + 1$ and stride $T/2$. The fixed bundle contains the first 20 non-shuffled training batches, not independent validation inputs. Exact tensor overlap with the older title-level batches remains unresolved because their saved inputs or hashes are unavailable. Fixing this bundle does not establish selection before all historical result inspection.

Both environment exports identify the same saved checkpoint and input bundle. The preserved package audit records matching byte hashes for both artifacts and the original `SHA256SUMS` entries. These identities are separate from checking the saved scalar results and do not imply a new execution of the evaluator. Line-break opportunities do not insert characters into the hashes.

Checkpoint SHA-256:

`01d54221e347e67766fa905cbf2965f94457431853b90e97f3ae6e8a5fed9b99`

Input-bundle SHA-256:

`7ee682ebb4040f83dbbc826712a19bd3e48c9280178e3b45188865655979dd05`

For each platform, q_{100} and q_{110} are compared with that platform's own q_{111} execution. Each quantity uses its corresponding reference vector and normalization. The 20 matched inputs yield $20 \times 3 = 60$ route executions per environment, not 60 independent training replicates; the two environments are not pooled with either earlier cohort. The main table reports arithmetic means of the per-batch relative discrepancies, whereas the original nine-unit and additional 18-unit main results report medians. Appendix B.11 retains the supplied mean, median, 95th-percentile, minimum, and maximum summaries. These compare routes within platforms, not CPU outputs against GPU outputs.

Supplied artifacts and metric implementation. The evidence index links the separate CPU and A100 CSVs, summaries, logs, manifests and environment exports, together with the evaluator and checkpoint/input identities. The evaluator computes each relative L_2 error after flattening the supplied tensors and casting them to FP64, using the corresponding platform-local q_{111} norm floored at 10^{-12} as denominator. It obtains full-model raw gradients through its imported helper and measures the parameter vector after the optimizer step minus its restored pre-step value. Aggregation follows per-batch error calculation: arithmetic mean, median, NumPy's 0.95 quantile, minimum and maximum over 20 batches. Re-aggregation of each platform's saved CSV checks the 210 summary entries across three routes, 14 metrics and five statistics without rerunning a solver.

Each recorded route–batch row includes one clipping-enabled restored step and an auxiliary, separately restored unclipped step. The wrapper also performs one excluded warm-up per route. Thus the 60 recorded rows are not a count of all internal step executions. Model and optimizer construction, gradient-vector ordering, and internal precision casts are delegated to an imported repository module. Source candidates for the seven previously missing repository dependencies have been recovered. Their availability permits static protocol inspection; historical file agreement does not establish every runtime import or bind the Mac repository commit to Colab. The original A100 Git and installation binding remain unestablished. Original A100 logits, gradient and update vectors were not supplied; their saved scalar discrepancies can be reaggregated but do not reconstruct those vectors.

A.11 PRIMAL/DUAL CONTROL PROTOCOL

The independent classifier control uses one non-pilot q_{111}-trained checkpoint, seed 0, step 5000, initialized at $\lambda_0 = 10^{-3}$, and fixed validation episode IDs 0–4. Each five-way episode has 25 support and 50 query examples with disjoint support/query indices. The underlying train/validation/test split contains 1,078/359/360 examples. Both formulations use the same support features $Z_s \in \mathbb{R}^{25\times 64}$, targets $Y_s \in \mathbb{R}^{25\times 5}$, positive ridge, mathematical output definition W, and task definition. The primal and dual systems have dimensions 64×64 and 25×25, respectively. This is a fixed-state implementation control with five inputs, not five training seeds or a dual retraining experiment.

Checkpoint SHA-256:

`2554860c02b3dd4ed6d563a38cc45f5da5aeeb7d6e5ca63d7bca3ae7a7bdf2c1`

Episode-manifest SHA-256:

`d95b7a8766f6d140504b008ede4e36c8219f99848d8eaa533d7850c8b1e98ec1`

Reference, reduction and restored update. Each formulation's mixed-precision routes are compared with its own q_{111}; only the consistency gate compares dual q_{111} with primal q_{111}. For the control, logits are flattened and concatenated in episode order 0–4 before computing relative error. The five FP32 cross-entropy losses are averaged, followed by one backward pass and one restored AdamW step. Thus the control CSV contains six formulation–route aggregates; the gradient/update comparison repeated in the five-row equivalence CSV is one global measurement, not five independent samples.

Each cell separately deep-copies the saved model and optimizer state and resets seed 0; this is not independent verification of checkpoint RNG restoration. AdamW uses learning rate 3×10^{-4}, $\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$ and zero weight decay, without clipping, AMP/scaling, retry, jitter or fallback. Gradients follow `named_parameters` order: log-ridge, first linear weight/bias, normalization weight/bias, and second linear weight/bias. Absent gradients become zeros before FP64 flattening and concatenation. Updates subtract post-step and pre-step parameters in FP32, then convert the difference to FP64. Relative L_2 errors use the corresponding reference norm floored at 10^{-12} and FP64 post-processing.

Dual precision stages and effective ridge. All three dual routes construct the Gram matrix in FP64. q_{100} uses FP32 Cholesky, solve and $Z_s^\top \alpha_d$; q_{110} raises Cholesky to FP64 but keeps the solve and final multiplication in FP32; q_{111} uses FP64 for all three. Every route casts W to FP32 for the surrounding classifier. The checkpoint stores FP32 log-ridge `-6.547471046447754`. Its binary64 exponential is `0.0014337368713159366`; the model instead exponentiates the FP32 parameter first, giving effective ridge `0.0014337368775159121`, and then converts it to the Gram dtype. The two values are compatible representations of this checkpoint; q_{111} does not recompute the exponential in FP64.

Consistency gate and residuals. Table 26 uses five-episode maxima for internal FP64 W, logit and loss discrepancies, and the single global gradient comparison and cosine for the averaged loss. Its five specified criteria pass. The global restored-update relative discrepancy is $2.0194277280644333 \times 10^{-6}$, but no update threshold was set and this observation did not enter the gate. Independent restore-repeatability is not evidenced. This implementation-consistency check is distinct from an arbitrary-precision reference or the exact-arithmetic identity.

Residuals are computed in FP64 as $\|H_{\text{factor}} W_{\text{internal}} - B_{\text{acc}}\|_F / \max(\|B_{\text{acc}}\|_F, 10^{-12})$ for primal and $\|H_{\text{factor}} \alpha_d - Y_{\text{acc}}\|_F / \max(\|Y_{\text{acc}}\|_F, 10^{-12})$ for dual. H_{factor} already includes the route's factorization-input rounding. Each cell reports the NumPy 0.95 quantile of its five residuals. The two systems have different dimensions and right-hand sides, so these residuals do not directly rank functional fidelity. The five per-episode q_{111} residuals are saved; the mixed routes retain only their p95 aggregates.

Evidence index. The accompanying `claim_evidence_map.csv` maps experiment groups and manuscript labels to reported input/checkpoint identifiers and the availability of result artifacts, evaluator scripts, and environment records. It distinguishes directly available result tables and scalar

aggregates, historical source candidates, and locally retained binary artifacts represented only by path/size/hash records. Neither a hash nor a later environment snapshot establishes complete execution provenance. Missing vectors and selected large audit tables limit what can be independently reaggregated; they are not evidence that those experiments were never run.

B Additional Results

B.1 Descriptive Route-Dominance Criterion

The Task 19/21 secondary criterion used in the final analysis applies to each scalar statistic separately and requires all four conditions below, using the pairwise spreads in Equation 13:

1. $R_{\mathrm{route/seed}}(x; 5000) > 2$.
2. $\Delta_{\mathrm{route}}(x; 5000) > \Delta_{\mathrm{route}}(x; 0)$.
3. At least two seeds have the same complete ascending ordering of the three endpoint values for $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$.
4. $\Delta_{\mathrm{route}}(x; 500) > \Delta_{\mathrm{route}}(x; 0)$ or $\Delta_{\mathrm{route}}(x; 2000) > \Delta_{\mathrm{route}}(x; 0)$.

Matching one pairwise direction does not satisfy the full-ordering condition. Task-performance similarity is not part of the criterion. The rule is descriptive and is not treated as preregistered; the ratio threshold alone is not the complete rule. Continuous endpoint statistics provide the primary evidence. Task 19 skips scalar ordering for vector-valued statistics. Earlier Task 14 and Task 15 analyses use different descriptive rules; their classifications are not interchangeable with this four-condition rule.

The raw learned-ridge spreads illustrate the denominator's role. For TinyStories, $\Delta_{\mathrm{route}} = 0.0020472726060284507$ and $\Delta_{\mathrm{seed}} = 0.002254387156830894$, giving $R = 0.9081282$. For WikiText, they are 0.012821787750969331 and 0.0006462138456602892, giving $R = 19.8414005$. A large ratio can reflect the route spread, the seed denominator, or both; it is not a variance decomposition or a significance test.

B.2 Four-Level Ridge-Initialization Evaluation

Table 14 retains the four tested initializations. Only initial ridge changes between these regimes, and ridge remains trainable. For each initialization, source training uses $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$ and seeds 0–4 for 5000 training episodes. Local checkpoints are saved at steps 500, 2000, and 5000; the table reports the step-5000 summaries on fixed validation episodes 0–15.

For each source checkpoint and evaluated route, the evaluator first averages the gradient vector or restored-update vector over the 16 episodes, then computes its relative disagreement with the corresponding q_{111} mean vector. The reported local value is the arithmetic mean of these disagreements over three source routes, two candidate routes ($\mathsf{q}_{100}, \mathsf{q}_{110}$), and five seeds, giving 30 rows. Thus it uses $E(\mathrm{mean}_e\, v_{q,e}; \mathrm{mean}_e\, v_{111,e})$, not $\mathrm{mean}_e\, E(v_{q,e}; v_{111,e})$. For updates, each $v_{q,e}$ is the parameter-change vector from its restored per-episode optimizer step; these update vectors are averaged after the steps have been evaluated, rather than taking one step on an averaged gradient. These local means are distinct from the 36-entry baseline audit medians.

The endpoint ratios use the raw-scale pairwise spreads in Equation 13. With five seeds and three routes, Δ_{route} averages 15 matched-seed route-pair differences and Δ_{seed} averages 30 within-route seed-pair differences.

The local means and R_λ increase as initialization decreases across these four settings. The minimum-eigenvalue ratios match R_λ at the reported precision, as expected from the rank-deficient ridge design. In contrast, R_{cond} rises through 2×10^{-4} and then falls at 10^{-4}. Absolute ridge and minimum-eigenvalue route spreads are not monotonic. Changing ridge initialization also changes regularization and training dynamics; these four values do not establish a continuous numerical law or a universal threshold. Reused historical runs are not new independent replications.

Table 14: Four-level ridge-initialization results. Local gradient and update disagreements are raw fractions; the two right columns are endpoint route-to-seed ratios.

λ_0	Mean local gradient E	Mean local update E	R_λ	R_{cond}
10^{-3}	0.00352110	0.000421454	0.456240	0.367052
5×10^{-4}	0.00780849	0.000922505	2.078835	1.662756
2×10^{-4}	0.02567498	0.003105570	5.157735	4.992702
10^{-4}	0.05562609	0.006692950	5.472349	4.280485

B.3 Seed-Level Ridge Transplantation

Table 15 records all 20 seed-level native/replaced/reference triples underlying Figure 1. Each value is already an arithmetic mean over matched IDs 0–15, evaluated under common q_{111}. The recipient and the separately trained reference have the same seed and step 5000. Only the recipient's log-ridge tensor is replaced; non-ridge recipient parameters are retained, features and outputs are recomputed, and no optimizer step is taken. The reference column is the same-seed q_{111} endpoint, not a later point in the recipient's training trajectory.

For this intervention only, let θ denote the non-ridge parameters, with ridge written separately, and let $\bar{m}_s(\theta, \lambda)$ be the mean regularized-Gram condition over the 16 matched evaluations for seed s. For a fixed recipient route q, define the native, ridge-replaced and reference statistics as

$$a_s = \bar{m}_s(\theta_q, \lambda_q), \quad b_s = \bar{m}_s(\theta_q, \lambda_{111}), \quad c_s = \bar{m}_s(\theta_{111}, \lambda_{111}). \tag{15}$$

With $\psi(v) = \log \max(v, 10^{-30})$, define

$$\begin{aligned} d_s^{\text{native}} &= |\psi(a_s) - \psi(c_s)|, \qquad d_s^{\text{repl}} = |\psi(b_s) - \psi(c_s)|, \\ C_s &= 1 - \frac{d_s^{\text{repl}}}{d_s^{\text{native}}}, \qquad d_s^{\text{native}} > 10^{-30}. \end{aligned} \tag{16}$$

Closure is undefined otherwise and is not clipped. Zero denotes unchanged distance to the reference, while one denotes zero remaining distance; defined closure may be negative but cannot exceed one. We report the median of the five seed-level closures. Thus the 16 measurements are averaged before taking logarithms, and seed aggregation follows closure calculation.

Ridge enters the measured matrix directly. For fixed feature matrix Z,

$$\lambda_{\min}(Z^\top Z + \lambda I) = \lambda_{\min}(Z^\top Z) + \lambda.$$

Replacing ridge can therefore change regularized-Gram conditioning without changing the representation-generating parameters.

The WikiText median closures are 98.8% and 98.0% for q_{100} and q_{110}; the lower-initialization ridge values are 98.1% and 99.3%. These percentages concern regularized-Gram conditioning specifically. The reference is approached by increasing condition in WikiText and decreasing it in ridge. Thus closure measures reference agreement rather than improvement or recovery of a representation.

B.4 Preliminary Five-Seed Shared-Ridge Evaluation

In the earlier independent-ridge control, each seed's q_{111} ridge schedule is replayed across $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$ while the non-ridge parameters continue training under their assigned routes. Seeds 0–4 use a one-episode lag: schedule entry 0 is the initial ridge, entry $j+1$ stores native episode j's pre-forward ridge, and controlled episode j uses entry j. Log-ridge is frozen and excluded from AdamW. The 15 runs all complete 5000 episodes; the 165 saved rows comprise 150 trajectory evaluations and 15 final endpoints. These endpoints are evaluated at step 5000 under common q_{111} using the separate 200-validation-episode evaluator. Its route-to-seed ratios are $R(\lambda_c) = 0$, $R(\lambda_{\min}) = 2.55 \times 10^{-9}$, and $R(\kappa) = 0.0472$. All three ratios belong to this evaluator, whose condition statistic uses the 95th percentile over 200 episodes. The minimum-eigenvalue statistic is the evaluator's minimum route-input positive-definiteness margin (`route_input_pd_margin_min`), not the later 16-episode semantic mean. Final endpoints apply schedule entry 5000, whereas the last

Table 15: Seed-level semantic conditions used in Figure 1. All values retain the precision of the supplied figure data. Reference values are repeated where both recipient routes share the same donor.

System	Recipient	Seed	Native	Replaced	Reference
WikiText	q_{100}	0	2537582.41	22165006.84	22317241.28
WikiText	q_{100}	1	2594777.11	18283704.32	17860652.15
WikiText	q_{100}	2	2878470.54	17747692.48	17190931.29
WikiText	q_{100}	3	2706498.03	18844594.36	19017070.82
WikiText	q_{100}	4	2596890.89	16955994.60	16582048.39
WikiText	q_{110}	0	2622686.57	23063225.46	22317241.28
WikiText	q_{110}	1	2521156.26	18564659.00	17860652.15
WikiText	q_{110}	2	2885258.59	18092166.07	17190931.29
WikiText	q_{110}	3	2591961.01	19196551.53	19017070.82
WikiText	q_{110}	4	2584996.41	17593189.62	16582048.39
Ridge	q_{100}	0	12629430.77	9214027.64	9156837.80
Ridge	q_{100}	1	13007999.51	9197929.14	9250547.20
Ridge	q_{100}	2	11831824.47	9351954.29	9428435.98
Ridge	q_{100}	3	12206242.12	8646966.50	8577933.03
Ridge	q_{100}	4	12077389.31	8085442.03	8023193.56
Ridge	q_{110}	0	12719053.64	9177403.42	9156837.80
Ridge	q_{110}	1	12891597.53	9242541.37	9250547.20
Ridge	q_{110}	2	13064010.67	9435551.74	9428435.98
Ridge	q_{110}	3	11896899.14	8552931.83	8577933.03
Ridge	q_{110}	4	12641498.09	8059907.77	8023193.56

trajectory evaluation precedes the next schedule assignment; their final recorded ridge and loss need not coincide.

No matching uncontrolled common-route baseline was retained for that evaluator. Consequently these ratios do not support a reduction percentage computed against a baseline from another episode set. The zero ridge ratio is imposed by the intervention; near-zero minimum-eigenvalue separation is also structurally expected in this rank-deficient design. This preliminary evaluation is distinct from the matched seven-seed, 16-episode intervention in the main text.

B.5 Locked Policy Construction and Paired Aggregation

The development-locked log-ridge schedule, internally labeled P5, uses q_{111} development trajectories from seeds 0–2. Table 16 lists its locked knots. Let $\theta_s^{\mathrm{dev}}(t)$ be development log-ridge and let t_j denote the knot steps. Each knot is the pointwise median across the three development seeds. The schedule interpolates linearly in log-ridge:

$$m_j = \mathrm{median}_{s\in\{0,1,2\}}\, \theta_s^{\mathrm{dev}}(t_j),$$
$$\theta_{\mathrm{P5}}(t) = (1-a)m_j + a m_{j+1}, \qquad a = \frac{t-t_j}{t_{j+1}-t_j}, \quad t_j \le t \le t_{j+1}.$$

The ridge used at training step t is $\lambda(t) = \exp(\theta_{\mathrm{P5}}(t))$; interpolation is not performed directly in λ. P5 was selected and locked after development, before confirmatory training on seeds 5–11. It is frozen and non-trainable during that evaluation, with no route-specific online ridge adaptation. This single development-derived schedule differs from replay of each confirmatory seed's own reference trajectory.

For condition, the seed-level statistic first averages the 16 episode conditions within each source route and then averages the three source routes. Task loss instead first averages 200 validation episodes within each route and then the same three routes. Both evaluations use common q_{111} execution, and each reported grand mean averages the seven resulting seed values. In symbols, for

Table 16: Knots of the development-locked log-ridge schedule (P5). Each value is the pointwise median of q_{111} development log-ridge across seeds 0, 1, and 2. Interpolation between these knots is linear in log-ridge.

Step	Locked log-ridge
0	−9.210340469997888
500	−9.154973978956892
1000	−9.107165324471913
2000	−9.03595063096627
3000	−8.949161556929168
4000	−8.88258076605998
5000	−8.813025494781678

policy p,

$$\bar{\kappa}_{s,p} = \frac{1}{3}\sum_{q\in\mathcal{Q}}\left(\frac{1}{16}\sum_{e=0}^{15}\kappa_{q,s,p,e}\right), \qquad \bar{\ell}_{s,p} = \frac{1}{3}\sum_{q\in\mathcal{Q}}\left(\frac{1}{200}\sum_{e\in\mathcal{E}_{200}}\ell_{q,s,p,e}\right), \qquad |\mathcal{E}_{200}| = 200.$$

The paired tests use the seven native-versus-P5 seed differences and exact two-sided sign flips. Neither the 21 route–seed rows nor the individual episodes serve as independent paired units. With $d_s = x_{s,\text{left}} - x_{s,\text{right}}$, the statistic is $T_{\text{obs}} = |\frac{1}{7}\sum_{s=5}^{11} d_s|$. All $2^7 = 128$ sign vectors are enumerated, including the original assignment and its global reversal. A permutation is counted when $T_{\text{perm}} \geq T_{\text{obs}} - \tau$, with $\tau = 10^{-15}\max(1, T_{\text{obs}})$; the p-value is the count divided by 128, with no plus-one correction and zero differences retained. This is not the mean absolute difference. The sign-flip test assumes that, under the null, the joint distribution of paired seed differences is invariant under independent sign reversals.

The primary policy comparison is P1 versus P5 across five continuous metrics: validation and test query loss, semantic minimum eigenvalue, semantic condition, and final ridge. P1 versus P2 and P2 versus P5 each use the same five metrics as secondary descriptive comparisons. All 15 continuous contrasts are unadjusted for multiplicity. Survival has no discordant seed pairs and no sign-flip p-value. The validation-loss and condition contrasts have 48/128 and 2/128 extreme assignments, respectively. The other primary contrasts have $p = 0.09375$ for test loss and $p = 0.015625$ for each of minimum eigenvalue and final ridge; the latter two are structurally related in this rank-deficient classifier. The 16 condition episodes are a subset of the 200 validation episodes. Individual losses for all 200 episodes were not retained, so the first loss aggregation is supported by saved run aggregates.

Seven-seed paired outcomes and descriptive intervals. Table 17 gives the saved seed-level P1 and P5 outcomes, using the episode and route aggregation above. We retain the original contrast orientation, $d_s = x_{s,\text{P1}} - x_{s,\text{P5}}$: positive differences indicate lower P5 condition or query loss. Differences and summaries are calculated before display rounding.

Table 18 reproduces the original descriptive percentile-bootstrap intervals from the seven paired differences. For each metric, 100,000 resamples draw seven seed differences with replacement; the interval endpoints are the 2.5th and 97.5th percentiles of the resampled means, using linear quantile interpolation. The original NumPy generator, `default_rng(210021)`, is reset for each metric. The resampling unit is the seed, not an episode or route. With only seven pairs, these intervals are descriptive and unadjusted for multiplicity; they are neither exact sign-flip-test inversions nor equivalence bounds.

Lower route-averaged condition in 7/7 seeds does not imply that every individual route improves. The loss result is lower in 4/7 seeds and higher in 3/7, with $p = 0.375$ and no specified equivalence margin; the task evidence does not establish equivalence.

B.6 Cross-Seed Local Comparisons

For the additional 18 matched validation units, median relative A_{used} disagreement is 33.78%, gated-branch disagreement is 1.58%, and absolute loss difference is 3.19×10^{-4}. Median gradient and clipped-update cosine similarities are 0.9961 and 0.9947, respectively. Table 19 gives

Table 17: P1 and P5 outcomes for all seven confirmatory seeds. Each entry first averages the specified episodes within each route and then the three routes under common q_{111} evaluation. Panel (a) uses 16 condition episodes and displays values in units of 10^6; panel (b) uses 200 validation episodes and unscaled query cross-entropy. Positive $d_s = \mathrm{P1} - \mathrm{P5}$ favors P5 for these lower-is-better outcomes.

(a) Regularized-Gram condition ($\times 10^6$)

Seed	P1	P5	d_s
5	11.553702	9.373051	+2.180650
6	11.039621	8.727209	+2.312412
7	11.124503	9.172382	+1.952122
8	11.533823	9.078508	+2.455315
9	11.170487	9.240831	+1.929656
10	11.650213	9.349679	+2.300534
11	11.124865	9.169809	+1.955056
Mean	11.313888	9.158781	+2.155106

(b) Validation query loss

Seed	P1	P5	d_s
5	0.900651166	0.899611967	+0.001039199
6	0.918703179	0.915158812	+0.003544367
7	0.890117036	0.891515187	−0.001398151
8	0.900043733	0.900772159	−0.000728425
9	0.894646508	0.894858813	−0.000212305
10	0.903396749	0.902022109	+0.001374639
11	0.892595649	0.891738230	+0.000857419
Mean	0.900022003	0.899382468	+0.000639535

Table 18: P1-minus-P5 paired differences over seeds 5–11. Sample SD uses the seven paired differences with denominator six. Intervals are 95% paired percentile-bootstrap intervals with 100,000 resamples. The final column reproduces the unadjusted exact two-sided sign-flip p-values from all 128 sign assignments; these tests and intervals are separate procedures. Condition values in this table are scaled by 10^{-6} for readability; query loss is unscaled.

Metric	Mean d	Sample SD	95% bootstrap interval	Paired p
Condition ($\times 10^6$)	2.155106	0.211630	[2.012254, 2.300154]	0.015625
Query loss	0.000639535	0.001629660	[−0.000414732, 0.001812570]	0.375

Table 19: Additional forward summaries for the cross-seed local cohort. Each entry is a median over six matched units for the indicated seed. Relative disagreements are raw fractions, not percentages; absolute loss differences are not normalized.

Seed	A_{used} E	Gated-branch E	$\lvert\Delta\mathcal{L}\rvert$
1	0.33658	0.01547	3.19×10^{-4}
2	0.35780	0.01801	2.76×10^{-4}
3	0.33569	0.01705	2.52×10^{-4}

Table 20: Observed minimum–maximum disagreements and median cosine similarities over the six units of each source seed. Disagreements are raw fractions, not percentages. Gradients are measured before clipping; updates use the source training configuration's norm-1 clipping and restored AdamW state.

Seed	Quantity	Minimum E	Maximum E	Median cosine
1	Raw gradient	0.06194	0.14871	0.99524
2	Raw gradient	0.05920	0.12482	0.99578
3	Raw gradient	0.04333	0.12800	0.99663
1	Clipped update	0.05446	0.13450	0.99485
2	Clipped update	0.06302	0.15135	0.99374
3	Clipped update	0.06113	0.15166	0.99447

the supplementary forward summaries by source seed, and Table 20 gives observed gradient and clipped-update ranges and cosine summaries. Each seed contributes six units from two checkpoint steps and three validation batches per checkpoint. These are descriptive within-seed summaries, not independent training replicates or confidence intervals.

The auxiliary unclipped-update medians are 0.09576, 0.10556, and 0.09985 for seeds 1, 2, and 3, respectively. Each equals its corresponding clipped-update median at the reported precision, although the individual clipping-active unit differs as documented in Appendix A.5. Gradient and clipped-update median disagreements are 5.95% and 6.14% at step 100, versus 12.12% and 14.44% at step 1000. These are observed differences between the two checkpoint steps; they do not establish monotonic accumulation of error during training. Optimizer restoration, clipping, and the six same-route repeatability controls are described in Appendices A.1 and A.5.

The high median cosine similarities do not support an interpretation as gradient reversal.

B.7 Route-Resolved Endpoint Task Results

The task summaries use 54 existing step-5000 endpoints across training routes $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$: seeds 0–2 for TinyStories SKA and seeds 0–4 for WikiText SKA and each ridge regime. All endpoints use common q_{111} evaluation, and each endpoint task CE is the arithmetic mean over evaluation IDs 0–15. SKA uses only the first example in each selected validation batch and evaluates loss on the query suffix; ridge uses the 16 selected validation episodes. These are not full-corpus or 200-episode task losses. The 54/54 completed endpoint records represent a disaggregation of the existing results, not 54 new training runs or independent seeds. These records are separate from the WikiText fixed-ridge and schedule-replay controls in Table 27, whose failed runs remain in their denominators. They also differ from the seeds 5–11 policy study, whose task loss averages 200 validation episodes.

For training route q and seed s, define the paired task difference

$$d_{q,s} = \mathrm{CE}(q, s; \mathrm{eval} = \mathsf{q}_{111}) - \mathrm{CE}(\mathsf{q}_{111}, s; \mathrm{eval} = \mathsf{q}_{111}).$$

Negative values indicate lower task CE for route q. Each route mean and sample standard deviation in Table 6 is computed across endpoint seed values. For the paired summaries below, subtraction precedes the across-seed reduction, and the sample standard deviation uses the $N_{\text{seed}} - 1$ denominator. It is computed from the paired differences, not by combining two marginal route standard deviations. The displayed route means and paired summaries are rounded separately.

Table 21: Same-seed task CE differences relative to endpoints trained under q_{111}. All endpoints are evaluated under common q_{111} at step 5000, after averaging IDs 0–15 within each endpoint. Means and sample standard deviations summarize paired seed differences. The final column counts lower/equal/higher CE for the listed training route. These are descriptive summaries.

Regime	Route	N_{seed}	Mean d	Sample SD	Lower/equal/higher
TinyStories SKA	q_{100}	3	$+0.000288$	0.001421	1/0/2
	q_{110}	3	-0.000738	0.000796	2/0/1
WikiText SKA	q_{100}	5	-0.001081	0.001658	4/0/1
	q_{110}	5	-0.000193	0.000683	3/0/2
Ridge ($\lambda_0 = 10^{-3}$)	q_{100}	5	$+0.000201$	0.001540	2/0/3
	q_{110}	5	$+0.000077$	0.000071	1/0/4
Ridge ($\lambda_0 = 10^{-4}$)	q_{100}	5	$+0.007743$	0.005214	1/0/4
	q_{110}	5	$+0.000245$	0.001209	2/0/3

For the lower-ridge-initialization q_{100} comparison, the paired differences in source-seed order are

$$[0.010590,\ 0.007071,\ -0.000983,\ 0.009888,\ 0.012151].$$

Thus four seeds have higher CE and one has lower CE. The summaries do not establish task-performance equivalence, statistical significance, or a route that is optimal across regimes.

For reconciliation with the earlier task aggregates, equal weighting of the route–seed endpoint means gives 2.258723 for TinyStories SKA, 2.386736 for WikiText SKA, 0.889772 for ridge initialized at 10^{-3}, and 0.892221 for ridge initialized at 10^{-4}. These correspond to the earlier four-decimal entries 2.2587, 2.3867, 0.8898, and 0.8922, respectively; the route-resolved summaries retain differences hidden by those grand means.

B.8 Original Nine-Comparison Results

Table 10 and Appendix A.4 specify the original nine q_{110}–q_{111} comparisons, using script-fixed seed-0 checkpoints and non-shuffled training inputs. All nine complete. Restored-update and retry behavior are given in Appendix A.1; metric construction and the clipping follow-up are in Appendix A.5.

Internal solver quantities are not uniformly similar. The measured operator A_{used} has median relative disagreement of 20.929%. Downstream, branch disagreement has median 1.107% and maximum 2.531%, while logit disagreement has median 0.882% and range 0.772–1.686%. The median signed loss difference $\delta\mathcal{L}$ is $+3.24726 \times 10^{-4}$; the median absolute difference is 3.61681×10^{-4}, with range 1.9312×10^{-5} to 1.056×10^{-3}.

At the same matched states, median gradient disagreement is 5.618% with range 4.886–13.430%, and median gradient cosine similarity is 0.998424. Median restored Adam-update disagreement is 5.854% with range 5.479–13.843%, and median update cosine similarity is 0.998300. Figure 2 summarizes the downstream logit and local update measurements.

Executing the source training configuration's norm-1 clipping leaves the updates unchanged on these same nine comparisons because clipping is inactive throughout this set (Appendix A.5). Thus, close downstream predictions coexist with route-dependent gradients and restored updates at the same model state; the forward comparison alone leaves this learning-level discrepancy untested.

B.9 Local Controls in Modest-Separation Regimes

These local diagnostics and the multi-seed endpoints in Table 6 use distinct sampling populations; they do not track a particular local perturbation along a training trajectory.

The historical TinyStories local functional, gradient and update tables contain headers only. Their report records that the conditional local stage was not carried out after no Cholesky boundary was found in the bounded search. We therefore do not retain numerical claims about four TinyStories

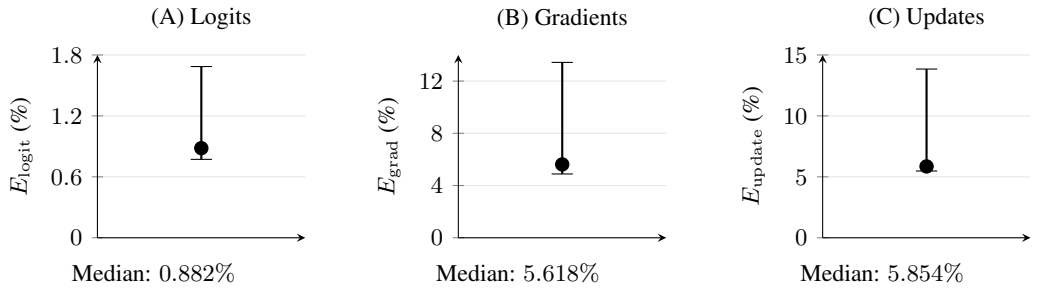

Figure 2: Nine matched q_{110} versus q_{111} comparisons from three seed-0 WikiText-2 checkpoints and training batches 0, 9 and 19. Points show medians; segments show observed minimum–maximum ranges, not confidence intervals. Updates use restored Adam; reproducing norm-1 clipping leaves them unchanged because clipping is inactive in all 18 route evaluations. Each quantity uses its own reference norm and vertical scale; cross-panel ratios are not amplification factors.

Table 22: Local q_{110} versus q_{111} diagnostics in baseline ridge with modest endpoint route separation. Entries are median relative disagreements; N counts matched local units, not independent training replicates. Magnitudes are not a common cross-system effect-size scale. Endpoint results and seed counts appear in Table 6.

Regime	N	Gradient	Update
Ridge baseline	36	0.1821%	0.1099%

local units. The eight TinyStories states in the separate 29-unit stagewise audit and the long-run endpoints in Table 6 do not replace these unavailable local records.

Appendix A.4 gives the nine-checkpoint and conditioning-based episode selection for the 36-entry independent-ridge local audit; the selected checkpoint–episode pairs are reused across routes. All eight routes complete all 36 entries, so hard Cholesky failure is not required for the detectable baseline-ridge local effect; these entries are not independent training replicates. The ridge audit shows local route disagreement alongside modest endpoint ridge and conditioning separation relative to between-seed variation.

B.10 Separate Custom-VJP Results

On the separate 29 frozen state–single-example units, we compare native selective-forward execution and selective-forward execution with the custom VJP against the full-route native-autograd reference. Median gradient disagreement decreases from 13.446% to 2.777%, while median restored Adam-update disagreement decreases from 18.759% to 2.966%. The reductions calculated from these medians are 79.35% and 84.19%, respectively, and all 29/29 cases improve. This analysis uses the same units as Table 2 with a distinct backward and optimizer protocol.

Retaining the selective forward outputs while substituting the recomputed solver-submap Jacobian substantially improves gradient and update agreement. This identifies sensitivity in the solver-associated backward pathway that is not exposed by inspecting forward values alone. The substituted Jacobian covers the complete solver submap; the diagnostic does not isolate every backward numerical effect, and its reductions are not causal mediation fractions.

This custom VJP saves semantic inputs, recomputes the solver submap with all solver stages in FP64, and calls `autograd.grad`; it is distinct from the separate hand-written VJP feasibility study. All nine saved synthetic checks report an overall pass, but five explicitly skip native full-route gradcheck. These records therefore do not establish nine executed native gradchecks. The historical overall no-go decision also required component-error, runtime and memory criteria that were not met; the local fidelity improvement does not establish a computationally advantageous replacement.

Table 23: Apple M4 Max CPU distributions across the 20 fixed batches. Relative L_2 errors use the platform-local q_{111} reference; $|\Delta\mathcal{L}|$ is the absolute loss difference. Updates are clipped restored Adam updates. Entries are recomputed from the saved scalar CSV and rounded to six significant digits.

Route	Metric	Mean	Median	p_{95}	Min	Max
q_{100}	E_{logit}	0.0165461	0.0165594	0.0170860	0.0156941	0.0172604
q_{110}	E_{logit}	0.0168202	0.0168068	0.0174741	0.0159067	0.0176722
q_{100}	$\lvert\Delta\mathcal{L}\rvert$	0.000861394	0.000831246	0.00161269	1.3113×10^{-5}	0.00198913
q_{110}	$\lvert\Delta\mathcal{L}\rvert$	0.000905108	0.000815272	0.00157380	0.000119448	0.00230765
q_{100}	E_{gradient}	0.367520	0.353094	0.473401	0.240584	0.477646
q_{110}	E_{gradient}	0.140616	0.139540	0.162707	0.112533	0.176625
q_{100}	E_{update}	0.415114	0.406302	0.481910	0.354073	0.510092
q_{110}	E_{update}	0.142051	0.139378	0.156327	0.127634	0.158783

Table 24: A100-SXM4-40GB distributions across the same 20 fixed batches. Relative L_2 errors use the A100-local q_{111} reference; $|\Delta\mathcal{L}|$ is the absolute loss difference. Updates are clipped restored Adam updates. Entries are recomputed from the saved scalar CSV and rounded to six significant digits.

Route	Metric	Mean	Median	p_{95}	Min	Max
q_{100}	E_{logit}	0.0274925	0.0276125	0.0283178	0.0261905	0.0289932
q_{110}	E_{logit}	0.0267472	0.0266414	0.0277291	0.0259735	0.0281922
q_{100}	$\lvert\Delta\mathcal{L}\rvert$	0.00193456	0.00203276	0.00289102	0.000291109	0.00293994
q_{110}	$\lvert\Delta\mathcal{L}\rvert$	0.00186611	0.00187945	0.00274930	0.000660658	0.00297308
q_{100}	E_{gradient}	0.377789	0.373711	0.474872	0.282989	0.518496
q_{110}	E_{gradient}	0.179418	0.177831	0.216740	0.137398	0.229581
q_{100}	E_{update}	0.421973	0.415508	0.521355	0.346328	0.526931
q_{110}	E_{update}	0.174266	0.175669	0.195955	0.155849	0.198414

B.11 Cross-Environment Local Results

Tables 23 and 24 retain the complete core distributions from the supplied CPU and A100 scalar CSVs. Each platform has 60/60 completed route executions at one fixed SKA checkpoint on the same 20 saved sequential training batches, with each candidate compared with that environment's own q_{111} execution. Appendix A.10 specifies the saved state, inputs and reference. These are summaries over batches at one state, not independent training replicates, and are not pooled with the original nine or additional 18 comparisons. The relative L_2 values are raw fractions; loss differences are absolute. Each mean averages 20 saved batch-level scalars; it is not an error computed after averaging vectors. Medians, linearly interpolated 95th percentiles, minima and maxima use the same rows. These inputs come from the training split, not an independent validation cohort. No standard deviation, confidence interval or significance test is added.

The tables now use the preserved scalar CSVs with one stated rounding rule. This supersedes four last-digit entries in the earlier A100 report transcription; the historical CSV, JSON and report are unchanged. The original generation chain for those four earlier entries remains unrecovered. No original logits, gradients or updates are reconstructed from these scalar summaries.

The CPU CSV records no clipping activation in 60 route executions: all global gradient norms are below one, and clipped and unclipped restored updates coincide for that CPU control. The supplied A100 CSV also records 0/60 clipping activations, coefficient 1 throughout, and identical pre-/post-clipping gradient norms within every row. Its clipped and unclipped updates are obtained from separate restored executions, whose recorded norm and discrepancy values differ slightly; inactive clipping alone does not identify the source of those differences. This A100 result does not change the separately recorded clipping-active unit in the 18-comparison cohort.

The gradient and update ranges are disjoint between the two candidates in both environments. On CPU, the q_{110} maxima 0.176625 and 0.158783 are below the corresponding q_{100} minima 0.240584 and 0.354073; on A100 the comparisons are $0.229581 < 0.282989$ and $0.198414 < 0.346328$. Both

Table 25: Primal/dual relative-L_2 values from one classifier checkpoint and validation episodes 0–4. Values are raw fractions against each formulation's own q_{111}. Logits are concatenated before comparison; gradients and restored AdamW updates use one backward pass on the five-loss mean, without clipping.

Formulation	Route	Logit	Gradient	Restored update
Primal	q_{100}	1.1198×10^{-3}	7.6070×10^{-3}	1.2004×10^{-3}
Dual	q_{100}	5.2788×10^{-5}	1.2503×10^{-4}	4.6255×10^{-5}
Primal	q_{110}	4.7281×10^{-4}	9.3731×10^{-4}	2.1761×10^{-4}
Dual	q_{110}	1.7635×10^{-5}	4.9657×10^{-5}	2.4490×10^{-5}

Table 26: q_{111} implementation consistency: dual is compared with primal. The first three observations are maxima over five episodes; gradient and update observations use the single global five-loss comparison. The original five gate criteria pass. An update observation exists, but no update threshold was specified and it was not included in the gate decision.

Metric	Observed	Threshold	Decision
Internal FP64 W relative L_2	6.4404×10^{-12}	$\leq 10^{-9}$	Pass
Logit relative L_2	2.3819×10^{-7}	$\leq 2 \times 10^{-6}$	Pass
Absolute loss difference	1.1921×10^{-7}	$\leq 2 \times 10^{-6}$	Pass
All-parameter gradient relative L_2	1.1740×10^{-6}	$\leq 2 \times 10^{-5}$	Pass
Gradient cosine	0.9999999999993	≥ 0.999999	Pass
Restored AdamW update relative L_2	2.0194×10^{-6}	Not specified	Not gated

supplied CSVs also show lower q_{110} gradient and update discrepancy in all 20 matched batches. This row check is not a significance test. The measurements compare routes within each environment; they are not CPU-versus-GPU vector discrepancies or a hardware-only intervention.

B.12 Primal/Dual Local Results

Formulation-specific reference comparisons. The additional classifier control uses one q_{111}-trained seed-0 checkpoint at step 5000 and validation episodes 0–4, with initialization $\lambda_0 = 10^{-3}$ and no dual retraining. Table 25 retains the four candidate rows from the six formulation–route aggregate records. Each candidate is compared with its own formulation's q_{111}, not a common primal reference. Logits are flattened and concatenated in episode order before computing relative L_2 error. The five FP32 query losses are averaged for one backward pass and one independently restored AdamW step, without clipping. The gradient and update comparisons are therefore global five-episode quantities, not five independent samples. Relative errors use the reference-vector norm with a 10^{-12} floor and FP64 postprocessing. Appendix A.11 gives the parameter order, FP32 update subtraction, state restoration and precision-stage assignments.

Sensitivity reduction under the equivalent formulation. Dividing the saved primal value by the saved dual value within each route and metric gives, in logit/gradient/restored-update order, approximately 21.2, 60.8, and 26.0 for q_{100}, and 26.8, 18.9, and 8.9 for q_{110}. These descriptive ratios use the unrounded saved scalars and are then rounded; they are not averages of per-episode ratios, task benefits or speedups.

The dual formulation substantially attenuates the measured mixed-precision discrepancies for both candidate routes at this fixed checkpoint and episode set. An algebraically equivalent formulation can therefore mitigate the observed sensitivity, and the primal magnitude is not formulation invariant. The control does not establish that dual is always more accurate or faster. Residual discrepancies remain nonzero, but no practical equivalence threshold or full dual training trajectory was tested. Nonzero values alone do not establish a meaningful training impact. This local control neither replaces the original primal endpoint and policy experiments nor tests an equivalent dual implementation of SKA. It changes the algebraic computation path as well as system size, so system dimension is not isolated as the cause.

The gate is an implementation-consistency check on tested inputs, distinct from the exact-arithmetic identity in Appendix C.1 and the earlier independent high-precision calibration. It does not certify q_{111} as mathematical ground truth. Appendix A.7 distinguishes the coverage of the production–mp80, pre-cast mp50–mp80 and primal–dual checks. The observed update relative L_2 is $2.0194277280644333 \times 10^{-6}$; its absolute L_2 is $2.0713025625865157 \times 10^{-8}$ and cosine is 0.9999999999979617. These values do not supply the unspecified update threshold or an independent historical restore-repeatability check. Neither the gradient gate nor a reference floor from another experiment supplies that missing update criterion.

System-specific solve residuals. The primal and dual residuals use their own equations and right-hand sides:

$$r_p = \frac{\|H^p_{\mathrm{factor}} W_{\mathrm{internal}} - B_{\mathrm{acc}}\|_F}{\max(\|B_{\mathrm{acc}}\|_F, 10^{-12})}, \qquad r_d = \frac{\|H^d_{\mathrm{factor}} \alpha - Y_{\mathrm{acc}}\|_F}{\max(\|Y_{\mathrm{acc}}\|_F, 10^{-12})}.$$

Postprocessing uses FP64, but H_{factor} can already contain FP32 rounding under q_{100}. Each saved p_{95} is a linearly interpolated quantile of five episode residuals. For q_{111}, these are $2.054421808223476 \times 10^{-15}$ for primal and $1.6057547159356287 \times 10^{-13}$ for dual. The five q_{111} residuals are preserved; the candidate-route records retain only the residual p_{95}, so their episode-level reductions cannot be independently checked. The primal system has dimension 64 and the dual system dimension 25. Their different equations and right-hand sides preclude ranking functional fidelity by comparing these residual magnitudes directly.

B.13 Matched Seven-Seed Shared-Ridge Comparison

The matched classifier comparison uses $\lambda_0 = 10^{-4}$, seeds 5–11, and routes $\mathsf{q}_{100}, \mathsf{q}_{110}, \mathsf{q}_{111}$. P1 learns log-ridge online. P2 replays the saved same-seed P1-q_{111} trajectory with a one-episode lag across that seed's routes: entry 0 is the common initial ridge, native episode j stores its pre-forward ridge in entry $j+1$, and P2 episode j uses entry j. Thus for $j \geq 1$ it uses the preceding native episode's pre-forward ridge; saved step-s evaluation assigns entry s. Exact replay refers to the stored trajectory values, not same-episode synchronization. Trajectories may differ across seeds. Non-ridge parameters continue training under their assigned routes, and P2 does not update ridge through the optimizer. Initialization and optimizer protocols are given in Appendix A.1. Step-5000 endpoints are evaluated under common q_{111} on the same validation episodes 0–15.

Using the spreads in Equation 13, the absolute condition route spread decreases from $2{,}539{,}987.73$ under P1 to $65{,}379.40$ under P2. The reported reduction is 97.4%, calculated as $1 - \Delta_{\mathrm{route}}(P2)/\Delta_{\mathrm{route}}(P1)$, and R_{cond} changes from 5.6939 to 0.1384. The percentage is the relative decrease in absolute between-route spread, not a reduction in mean condition, a fraction of the full representation explained, or a causal mediation fraction. The earlier five-seed, 200-episode p_{95} results in Appendix B.4 are a separate evaluation and cannot serve as this comparison's baseline.

B.14 WikiText Training-Time Ridge Controls

Table 27 retains all five attempts per route for fixed nominal ridge 10^{-3} and same-seed q_{111} schedule replay. The replay starts from the same ridge value. Every incomplete q_{100} run fails at Cholesky; both other routes complete under each control. Assignment, optimizer, and clipping details are in Appendix A.1. These controls are separate from the complete four-regime endpoints in Table 6.

Table 27: WikiText training-time ridge controls. Entries are completed runs out of five attempts per route; failed runs remain in the denominator.

Intervention	q_{100}	q_{110}	q_{111}
Fixed ridge ($\lambda_{\mathrm{fixed}} = 10^{-3}$)	0/5	5/5	5/5
Matched q_{111} ridge-schedule replay	1/5	5/5	5/5

C Additional Derivations

Differential of the regularized solve. In exact arithmetic, differentiating $G_\lambda W = C$ gives $(\mathrm{d}G_\lambda)W + G_\lambda \mathrm{d}W = \mathrm{d}C$. Since $\lambda > 0$ makes the exact regularized Gram matrix invertible, left multiplication by G_λ^{-1} yields $\mathrm{d}W = G_\lambda^{-1}(\mathrm{d}C - (\mathrm{d}G_\lambda)W)$. This identity describes the exact map; it is neither a bound on floating-point differentiation error nor a prediction of observed optimizer disagreement.

Minimum eigenvalue in the independent ridge design. With 25 support examples and 64 features, $Z_s^\top Z_s$ has rank at most 25 and hence at least 39 zero eigenvalues in exact arithmetic. Adding $\lambda_c I$ shifts every eigenvalue by λ_c, so $\lambda_{\min}(H_e) = \lambda_c$ and $\lambda_{\max}(H_e) = \sigma_{\max}(Z_s)^2 + \lambda_c$. The unfloored spectral condition number is consequently $1 + \sigma_{\max}(Z_s)^2/\lambda_c$. This explains why ridge and minimum eigenvalue are not independent mechanistic evidence, while condition also depends on the feature matrix. It does not remove finite-precision differences in the measured eigenspectrum.

Closure bounds and interpretation. With a strictly positive native absolute log gap, the replacement gap is nonnegative. Their ratio is therefore nonnegative, so defined closure $1 - d_{\mathrm{replaced}}/d_{\mathrm{native}}$ cannot exceed one. It can be negative if replacement moves the measured statistic farther from the reference. No clipping is applied. The condition statistic depends directly on ridge, so closure of this gap alone is not full training-time mediation.

C.1 Primal–Dual Identity for Positive Ridge

For $Z = Z_s \in \mathbb{R}^{25\times 64}$, $Y = Y_s \in \mathbb{R}^{25\times 5}$ and $\lambda = \lambda_c > 0$, let $B_p = Z^\top Y$, $H_p = Z^\top Z + \lambda I_{64}$ and $H_d = ZZ^\top + \lambda I_{25}$. Both matrices are positive definite in exact arithmetic and hence invertible. Direct multiplication gives

$$(Z^\top Z + \lambda I_{64})Z^\top = Z^\top(ZZ^\top + \lambda I_{25}). \tag{17}$$

Multiplying $H_p Z^\top = Z^\top H_d$ on the left by H_p^{-1} and on the right by $H_d^{-1}Y$ yields

$$W_p = H_p^{-1}B_p = Z^\top H_d^{-1}Y = Z^\top \alpha_d = W_d, \qquad \alpha_d = H_d^{-1}Y.$$

The inverse notation denotes the mathematical solve, not explicit matrix inversion in the implementation. This identity equates the exact maps at fixed features, targets and positive ridge. It is not a finite-precision stability theorem or a guarantee for every input, and does not specify either implementation's precision-stage assignments.