FLASHRLS: EXACT DIFFERENTIABLE SECOND-ORDER ONLINE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Second-order online methods adapt to streaming data through sequential matrix updates that are costly on GPUs. Existing chunked implementations handle squared loss, where update weights are known in advance. Logistic and robust losses introduce response-dependent weights, making both batching and exact differentiation more difficult. We present FlashRLS, a block formulation that runs local update rules on a small matrix of observation interactions and batches the dense work into matrix products. It covers recursive least squares (RLS), online Newton, recursive Laplace, and robust updates, and we prove that it reproduces sequential execution and its exact gradients. With the forward rule held fixed, exact differentiation improves associative recall by 9.4 points and reduces outlier-regression MSE from 0.085 to 0.057. FlashRLS matches sequential estimators with 38.8×, 42.3×, and 53.3× speedups on Criteo, echo cancellation, and chaotic-system readouts.

1 INTRODUCTION

Second-order online methods adapt to streaming data using a dense matrix that accumulates the features seen so far and sets how strongly each observation corrects the predictor. Beyond adaptive filtering and forecasting, they are now used as layers inside neural networks and trained end-to-end through their updates, as in test-time training (TTT) (Sun et al., 2025).

Fast implementations batch these updates into matrix products, which requires each observation's weight to be known in advance. This holds for squared loss but not for logistic or robust losses, where each weight depends on the learner's own response. We introduce FlashRLS, which computes such response-dependent updates in blocks and differentiates through them exactly. Our method also differentiates through adaptive curvature, making non-quadratic online learners trainable at chunked-kernel cost.

1.1 THE ONLINE LEARNER

Recursive least squares (RLS) fits a linear predictor to a stream of observations, one at a time (Sayed & Kailath, 1994). It keeps a running record of which feature directions it has already seen, and makes larger corrections in directions it knows little about. When a neural network produces the features and targets, the RLS state becomes a learned summary of the stream (Sun et al., 2025; Wang et al., 2025). Online Newton, recursive Laplace, and proximal rules extend the same idea beyond squared loss (Hazan et al., 2007; Koyama et al., 2010; Kulis & Bartlett, 2010).

Each step has three parts: predict, decide how much to correct, and update. The learner holds a predictor $W_t \in \mathbb{R}^{m\times d}$ and an information matrix H_t, which accumulates the features seen so far; we write $P_t = H_t^{-1} \in \mathbb{R}^{d\times d}$. Given a feature k_t and target y_t, the learner first forms

$$p_t = W_{t-1}k_t, \qquad v_t = P_{t-1}k_t, \qquad s_t = k_t^\top v_t. \tag{1}$$

Here p_t is the prediction before the update, v_t is the direction of the correction, and s_t measures how uncertain the learner is along k_t. A local rule then chooses two scalars, $(u_t, w_t) = \mathcal{O}_t(p_t, s_t, y_t)$: the size of the correction u_t and the weight w_t that this observation gets in the information matrix. The state is updated as

$$W_t = W_{t-1} + u_t v_t^\top, \qquad H_t = H_{t-1} + w_t k_t k_t^\top. \tag{2}$$

For squared loss, the rule is fixed in advance: $w_t = 1$ and $u_t = (y_t - p_t)/(1 + s_t)$, which recovers standard RLS. For other losses, w_t depends on the response. For logistic loss, for example, w_t is the

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.

local curvature $\sigma(z_t)(1-\sigma(z_t))$, so confident predictions add little information. This dependence is what makes these rules harder to batch and to differentiate. Local rules may also carry auxiliary state between calls (Appendices A and C).

For response-dependent rules, an early observation changes both the predictor and the information matrix used by later updates. A robust rule, for example, can reduce an outlier's influence on both the current correction and the accumulated information. Exact differentiation must follow both paths. Stopping derivatives through adaptive weights and target corrections keeps the forward computation unchanged and defines a different gradient for training.

1.2 Related work

Each observation changes the dense matrix needed by the next, so direct implementations proceed one observation at a time with matrix–vector products and rank-one updates.

Factor-based methods. Square-root and innovations methods propagate triangular factors of the information matrix for numerical stability (Kailath, 1968; Bierman, 1977), and block Householder and GPU QR updates process several observations per step (Hsieh et al., 1993; Bojanczyk et al., 1993; Andrew & Dingle, 2014). Related factor-based computations enable parallel smoothing and square-root inference (Särkkä & García-Fernández, 2020; Yaghoobi et al., 2025), and Gramian differentials give derivatives of square-root filters (Corenflos, 2026). These methods target fixed observation weights, and their update schedules add dependent stages as the feature dimension grows. FlashRLS instead confines the sequential dependence to a $C \times C$ core whose weights may be set by the response, and differentiates through that choice (Section 4.3, Appendix N).

Learned online regression layers. DeltaNet and Gated DeltaNet compute delta-rule updates in chunks through a triangular (WY/UT) system (Yang et al., 2024; 2025); for fixed weights, our direction solve (Equation 28) has the same form. MesaNet and Gated KalmaNet maintain second-order statistics but read them with conjugate-gradient or Chebyshev iterations (von Oswald et al., 2026; Peng et al., 2026), whose accuracy depends on conditioning, iteration count, and precision (Higham, 2002). All these layers use squared loss. FlashRLS extends the chunked form to response-dependent weights and computes reads with exact direct solves (Sections 4.4 and L.5; Appendix B.8).

1.3 Contributions

- **Exact block execution.** FlashRLS reproduces every sequential prediction, state, and gradient of second-order online learners with response-dependent curvature, at the same asymptotic cost forward and backward. It extends to known diagonal process noise and transitions, with residual-based error bounds (Sections 2–3).
- **Exact gradients matter.** With the forward rule held fixed, the exact backward improves associative recall by 9.4 ± 1.1 points and reduces outlier-regression MSE from 0.085 to 0.057 (Section 4.1).
- **Fast exact online learning.** FlashRLS matches sequential estimators while running $38.8\times$ faster on Criteo, $42.3\times$ on echo cancellation, and $53.3\times$ on chaotic-system readouts (Section 4.6).

.

2 Setup and assumptions

Our goal is to compute every prediction and state in Equations (1)–(2) without updating a dense matrix after every observation. We process observations in blocks: the local rules still run in order, one per observation, but the dense operations are carried out once per block.

Notation. A block (or chunk) contains C consecutive observations. We stack its keys and queries as rows of $K, Q \in \mathbb{R}^{C\times d}$ and its targets as rows of $Y \in \mathbb{R}^{C\times m}$. A *read* is any product of a query with the state at a given point in the block, for example the prediction $W_{t-1}q_t$ or the vector $P_{t-1}q_t$. We write $\mathrm{tril}(D,-1)$ for the strictly lower-triangular part of a matrix D.

We make two assumptions. The first ensures that the block computation has well-defined inputs; the second ensures that every update is valid.

Assumption 2.1 (Local rule)**.** The incoming H_0 is positive definite. Keys, queries, and exogenous inputs of the block are known before its updates begin. Each local rule returns a unique output for given inputs; if it makes random choices, they are shared across implementations.

Assumption 2.2 (Feasible curvature)**.** Every pivot $h_t = 1 + w_t s_t$ is positive.

A positive pivot keeps H_t positive definite. The inverse then satisfies $P_t = P_{t-1} - \beta_t v_t v_t^\top$ with $\beta_t = w_t/h_t$, by the Sherman–Morrison formula. The assumption holds automatically whenever $w_t \geq 0$, since then $s_t \geq 0$ and $h_t \geq 1$. This covers squared loss and logistic and Poisson Laplace rules. Rules with negative weights must check the pivot explicitly (Appendix B).

The challenge is that each local rule needs the exact p_t and s_t that sequential execution would give, and both depend on every earlier update in the block. Section 3 computes them from a small $C \times C$ system and then reconstructs the full states.

3 FLASHRLS

Sequential execution updates a $d \times d$ matrix after every observation. FlashRLS instead tracks how the observations in a block interact with each other, which takes only a $C \times C$ table. The method has two stages: a small recurrence runs the local rules in order using this table, and a single triangular solve then recovers all update directions, from which matrix products give every prediction and the outgoing state.

3.1 COMPUTING THE LOCAL INPUTS

Each local rule needs the prediction p_j and uncertainty s_j that sequential execution would give. Both depend on the state after all earlier updates in the block, but only through its interaction with the block's keys. We therefore track the table $M_{ij} = k_i^\top P_{j-1} k_j$: how keys i and j interact under the state just before update j.

The table starts from the interactions under the incoming state. With $B_0 = KP_0$, $\Gamma = B_0 K^\top$, and $a = KW_0^\top$ (whose row j is $(W_0 k_j)^\top$), each earlier update ℓ subtracts its contribution:

$$M_{ij} = \Gamma_{ij} - \sum_{\ell<j} \beta_\ell M_{i\ell} M_{j\ell}, \qquad s_j = M_{jj}, \qquad p_j = a_j + \sum_{\ell<j} M_{j\ell} u_\ell. \tag{3}$$

Column j gives s_j and p_j, so the local rule $\mathcal{O}_j(p_j, s_j, y_j)$ can run, and its weight enters the next column.

The correction term matters even when keys start out unrelated. With $P_0 = I$ and keys $k_1 = (1,0)^\top$, $k_2 = (1,1)^\top$, $k_3 = (1,-1)^\top$, keys 2 and 3 are orthogonal, so $\Gamma_{32} = 0$. After a unit-weight squared-loss update on k_1 ($\beta_1 = \frac{1}{2}$), they are not: $M_{32} = 0 - \beta_1 M_{31} M_{21} = -\frac{1}{2} = k_3^\top P_1 k_2$.

Proposition 3.1 (Exact local inputs)**.** *Under Assumptions 2.1–2.2, Equation* (3) *satisfies $M_{ij} = k_i^\top P_{j-1} k_j$ for every $i \geq j$, and returns exactly the predictions, variances, and local-rule outputs of the sequential learner.*

The proof is by induction: each update ℓ changes the inverse state by $-\beta_\ell v_\ell v_\ell^\top$ and the predictor by $u_\ell v_\ell^\top$, and projecting these onto the keys gives the two sums (Appendix B).

3.2 RECOVERING THE TRAJECTORY

Once all weights are known, the update directions follow in one step. Each direction $v_t = P_{t-1} k_t$ equals the initial direction $P_0 k_t$ minus the contributions of earlier updates, $\sum_{\ell<t} \beta_\ell M_{t\ell} v_\ell$. Stacking the directions as rows of A gives a unit lower-triangular system. With U the matrix of rows $u_t^\top$ and $\mathcal{B} = \mathrm{diag}(\beta_1, \ldots, \beta_C)$,

$$T = I + \mathrm{tril}(M, -1)\mathcal{B}, \qquad TA = B_0. \tag{4}$$

Every state in the block is the incoming state plus the rank-one corrections made so far, so it can be read off from the first rows of A and U.

Theorem 3.2 (Exact sequential states and predictions)**.** *Under Assumptions 2.1–2.2, let A_c, U_c contain the first c rows and $\mathcal{B}_c$ the corresponding diagonal block. Then, for every $0 \le c \le C$,*

$$P_c = P_0 - A_c^\top \mathcal{B}_c A_c, \qquad W_c = W_0 + U_c^\top A_c. \tag{5}$$

Predictions and inverse-state products before each update are

$$O = QW_0^\top + \mathrm{tril}(QA^\top, -1)U, \qquad Z = QP_0 - \mathrm{tril}(QA^\top, -1)\mathcal{B}A. \tag{6}$$

Row t of O is $(W_{t-1}q_t)^\top$ and row t of Z is $(P_{t-1}q_t)^\top$. The strictly lower-triangular mask selects the updates made before t; including the diagonal gives reads after update t, and any other prefix mask gives reads at that point. Setting $c = C$ gives the outgoing state.

Algorithm 1 FlashRLS with response-adaptive curvature

Require: P_0, W_0, K, Q, Y and local rules satisfying Assumptions 2.1–2.2

$B_0 \leftarrow KP_0$; $G \leftarrow B_0K^\top$; $b \leftarrow KW_0^\top$; $M \leftarrow 0$
for $t = 1, \dots, C$ **do**
  $M_{t:C,t} \leftarrow G_{t:C,t}$; $s_t \leftarrow G_{tt}$; $p_t \leftarrow b_t^\top$
  $(u_t, w_t) \leftarrow \mathcal{O}_t(p_t, s_t, y_t)$; $\beta_t \leftarrow w_t/(1 + w_t s_t)$; $U_{t,:} \leftarrow u_t^\top$
  $F \leftarrow \{t+1, \dots, C\}$; $g \leftarrow M_{F,t}$
  $G_{F,F} \leftarrow G_{F,F} - \beta_t g g^\top$; $b_F \leftarrow b_F + g u_t^\top$
end for
$\mathcal{B} \leftarrow \mathrm{diag}(\beta)$; solve $(I + \mathrm{tril}(M, -1)\mathcal{B})A = B_0$
Compute reads using Equation (6); return (P_C, W_C) using Equation (5)

Only the loop in Algorithm 1 is sequential, and it works on $C \times C$ quantities. Preparation, the triangular solve, reads, and state updates are batched matrix products. Any partition into consecutive blocks gives the same trajectory, and $C = 1$ recovers fully online execution.

3.3 Exact gradients through adaptive curvature

An early response influences later predictions in two ways: it changes the predictor, and it changes the information matrix used by later updates. The backward pass follows both paths. A bar denotes a derivative of the scalar training objective.

Assumption 3.3 (Differentiable local solutions)**.** The local rules are differentiable on the trajectory, pivots stay bounded away from zero, and each implicit local equation has a nonsingular response Jacobian. Local solutions and derivatives are exact.

For $z + cs\nabla\ell(z;\theta) = p$, the local reverse pass solves $[I + cs\nabla^2\ell(z;\theta)]^\top \lambda = \bar{z}$. It then differentiates the mean and curvature coefficients as well as the loss parameters. A Laplace weight $w = c\ell''(z)$ contributes $c\ell'''(z)\bar{w}$ to the response derivative. This term assigns credit to an early response for its effect on later updates.

Theorem 3.4 (Exact adaptive backward)**.** *Under Assumptions 2.1, 2.2, and 3.3, reversing Algorithm 1 through its local rules, interaction recurrence, direction solve, reads, and boundary update gives the exact gradient of the sequential learner for any differentiable objective of its emitted reads and outgoing state. The backward has the same dense arithmetic order as the forward, plus local derivative and chosen recomputation costs.*

Appendix G proves the result and gives the reverse operations. The direction solve reuses the existing factor: $T^\top V = \bar{A}$ gives $\bar{B}_0 \mathrel{+}= V$ and $\bar{T} \mathrel{+}= -VA^\top$. Reversing the interaction updates carries derivatives to earlier curvature choices. This carries losses on later reads back to the updates that created their state, using the same factorization. Appendix G.6 extends the backward to piecewise-smooth rules; Appendices G.7–G.8 detail kink conventions and pivot conditions.

A two-observation example. For scalar recursive Laplace with unit loss weights, let $s_1 = \Gamma_{11}$ and $b = \Gamma_{21}$. The first response determines $\beta_1 = \ell_1''(z_1)/[1 + s_1\ell_1''(z_1)]$, hence $s_2 = \Gamma_{22} - \beta_1 b^2$. The

second response equation is $F_2 = z_2 + b\ell_1'(z_1) + s_2\ell_2'(z_2) - a_2 = 0$, with

$$\frac{\partial F_2}{\partial z_1} = \underbrace{b\ell_1''(z_1)}_{\text{prediction path}} - \underbrace{\frac{b^2\ell_1'''(z_1)\ell_2'(z_2)}{[1+s_1\ell_1''(z_1)]^2}}_{\text{curvature path}}. \tag{7}$$

The second term measures how the first response changes uncertainty at the next observation. The exact backward includes both terms. Quadratic loss has $\ell_1''' = 0$, giving the null control in Section 4.1.

3.4 Squared loss and known dynamics

For unit-weight squared loss, the information weights are known in advance. The interaction recurrence is then represented by one Cholesky factor:

$$S = I + KP_0K^\top = LL^\top, \qquad X = L^{-1}KP_0, \qquad E = L^{-1}(Y - KW_0^\top). \tag{8}$$

Corollary 3.5 (Exact sequential regression)**.** *With $P_0 \succ 0$, the state after each update c satisfies*

$$P_c = P_0 - X_{1:c}^\top X_{1:c}, \qquad W_c = W_0 + E_{1:c}^\top X_{1:c}. \tag{9}$$

For a mask $\mathcal{M}$ selecting the updates used by each query, $O = QW_0^\top + [(QX^\top) \odot \mathcal{M}]E$.

The first c rows and columns of L factor the system for the first c observations. Positive observation weights are absorbed into scaled rows (Appendix A).

Known process noise and diagonal transitions. The same construction allows the state to evolve between observations. Known diagonal process noise gives $P_t^- = P_{t-1} + \mathrm{diag}(n_t)$. Let $\rho_t = \sum_{j\le t} n_j$, and let R have rows $\rho_t^\top$.

Proposition 3.6 (Execution with known dynamics)**.** *Under Assumptions 2.1–2.2 for the predicted state, replace the preparation by*

$$B = KP_0 + K \odot R, \qquad \Gamma_{ij}^{\mathrm{dyn}} = k_i^\top[P_0 + \mathrm{diag}(\rho_{\min(i,j)})]k_j. \tag{10}$$

The same interaction recurrence and solve $TA = B$ give the exact states $P_c = P_0 + \mathrm{diag}(\rho_c) - A_c^\top \mathcal{B}_c A_c$ and $W_c = W_0 + U_c^\top A_c$.

The minimum index selects the process noise shared by the observations. Invertible diagonal transitions known before block preparation are handled by a change of coordinates, with transformed keys and queries $\bar{k}_t = \Phi_t^\top k_t$, $\bar{q}_t = \Phi_t^\top q_t$, where $\Phi_t = F_t \cdots F_1$. Appendix H gives the state and noise transformations. Higher-rank observations and signed site updates are in Appendices E and I.

3.5 GPU execution and scaling

Within each block, the small interaction recurrence handles the chronological dependence, and the dense work uses batched matrix operations. The fp64 core evaluates the small system in double precision; the recurrent state uses fp32 in the GPU experiments.

Proposition 3.7 (Bounded-block execution)**.** *Under Assumptions 2.1–2.2, chronological numerical blocks of size at most b preserve every sequential state. For N observations with $O(1)$ reads each, the work is*

$$O\big(N[d^2 + dm + b(d+m) + b^2] + T_{\mathrm{local}}\big), \tag{11}$$

with $O(d^2 + dm)$ persistent state and $O(b^2 + b(d+m))$ reusable forward workspace. T_{local} includes local-solver and auxiliary-state work.

Each block costs $O(b^3)$ in the core, $O(b^2(d+m))$ in reconstruction and reads, and $O(b(d^2+dm))$ in preparation and state updates. Choosing $b \le d$ gives $O(N(d^2+dm))$ dense work; b can stay fixed as context length grows (Appendix B.7). Storing corrections for backward uses $O(N(d+m))$ entries, plus local-rule data for backward (Appendix J). Table 9 compares execution structures. Fixed b bounds the dense work per block; the incoming state summarizes all preceding updates.

Numerical error control. Residuals provide checks for both solves. A monotone scalar response equation with derivative at least one has solution error bounded by its absolute residual. For a fixed interaction core, $R = B_0 - T\widehat{A}$ gives $\|\widehat{A} - A\|_2 \le \|T^{-1}\|_2\|R\|_2$. Appendices F and J.5 propagate these defects through later responses and state reconstruction for fixed and adaptive geometry.

4 NUMERICAL EXPERIMENTS

The nonlinear experiments test whether training through adaptive curvature improves learning and whether the corresponding online rule can be executed efficiently. Linear regression, filtering, and forecasting test the full-covariance foundation and its application-level cost. The appendices give the protocols, numerical checks, and per-seed results.

4.1 TRAINING THROUGH ADAPTIVE CURVATURE

We test if the curvature path improves learning by changing only the backward rule. The arms share the adaptive forward computation, initialization, data order, and learning rate. Stopped curvature blocks derivatives through information weights and target corrections. CG-30 uses 30 conjugate-gradient iterations for the implicit backward solve (Golub & Van Loan, 2013; Blondel et al., 2022). Frozen memory trains projections through the direct path and updates memory in the forward pass.

Logistic MQAR evaluates associative recall (Arora et al., 2024); Huber regression tests residual-dependent information updates with heavy-tailed noise and outliers (Huber, 1964). Table 1(a) summarizes the ten-seed studies. Appendix P lists their per-seed results; Appendix L.6 gives the separate three-seed study and its timing measurements.

Table 1: (a) Ten-seed backward studies, mean $\pm$ standard deviation. MQAR: steps to 99% accuracy and recall at 224 pairs after 3,000 steps (B1). Huber: outlier MSE at 256 points (B2). CG steps cover the eight runs reaching the criterion; recall covers all ten. Batch Huber is the fixed offline reference. Dashes denote unreported or inapplicable quantities. (b) Separate three-seed B1 timing study: time to 99% is estimated as mean steps times mean step latency within this study.

(a) Ten-seed backward studies

	MQAR (B1)		Huber (B2)
Backward	Steps ↓	Recall (%) ↑	MSE ↓
FlashRLS (exact backward)	$1{,}110 \pm 91$	89.1 ± 1.7	0.057 ± 0.004
Stopped curvature	$1{,}997 \pm 168$	79.7 ± 2.6	0.085 ± 0.008
CG-30 implicit	$2{,}314 \pm 417$ (8/10)	71.2 ± 3.5	0.098 ± 0.029
Frozen memory	–	50.6 ± 6.0	0.158 ± 0.039
Squared loss, exact	992 ± 100	72.8 ± 3.5	0.161 ± 0.022
Batch Huber, offline	–	–	0.051

(b) Three-seed MQAR timing

Backward	Steps to 99% ↓	ms/step ↓	Est. time to 99% (s) ↓
FlashRLS (exact backward)	$1{,}130 \pm 90$	214 ± 3	242
Stopped curvature	$2{,}050 \pm 180$	171 ± 3	351

The complete backward improves recall by 9.4 points (paired 95% CI ± 1.1) and reaches 99% training accuracy in $1{,}110 \pm 91$ steps. The stopped-curvature arm needs $1{,}997 \pm 168$ steps. Outlier-regression MSE falls from 0.085 ± 0.008 to 0.057 ± 0.004. These comparisons measure the benefit of differentiating the same adaptive forward rule. In the separate three-seed B1 study, the exact backward takes 214 ± 3 versus 171 ± 3 ms per step and needs $1{,}130 \pm 90$ versus $2{,}050 \pm 180$ steps. Their products give 242 versus 351 s to 99%, an estimated $1.45\times$ speedup (Table 1(b)).

Local rules and controls. The logistic information weight is $\mathrm{sigmoid}(z_t)[1 - \mathrm{sigmoid}(z_t)]$. Huber uses $w_t = \min\{1, \delta/|r_t|\}$ for local residual r_t, with $w_t = 1$ at zero and $\delta = 0.5$. The backward differentiates these weights and target corrections. Quadratic exact and stopped gradients coincide. In the three-seed Gaussian control, exact and stopped Huber MSEs are 0.042 ± 0.004 and 0.043 ± 0.004 (Appendix L.6). Appendix G.7 details the threshold derivatives.

Exact gradients help as curvature varies. Response-dependent weights let early responses influence later updates; stopping their derivatives omits this training signal. In the contamination sweep (Table 2), increasing outliers from 5% to 20% raises $\mathrm{CV}^2(w)$ from 0.45 to 1.6. The stopped-minus-exact MSE gap grows from 0.012 to 0.051 & gradient cosine falls from 0.90 to 0.68. The Gaussian control has nearly constant weights and a 0.001 MSE gap; constant quadratic curvature gives zero gap. These results support the benefit of differentiating through response-dependent curvature.

Table 2: MSE reduction is stopped-curvature MSE minus FlashRLS (exact backward) MSE; positive values favor FlashRLS.; gradient cosine compares the two backwards. $\mathrm{CV}^2(w) = \mathrm{Var}_t(w_t)/\mathrm{E}_t[w_t]^2$. The Gaussian control and contaminated distributions are evaluated separately (Appendix L.6).

Condition	$\mathrm{CV}^2(w)$	FlashRLS MSE reduction	Gradient cosine
Gaussian control	0.02 ± 0.01	$+0.001 \pm 0.003$	0.995 ± 0.003
5% outliers	0.45 ± 0.05	$+0.012 \pm 0.004$	0.90 ± 0.02
10% outliers	0.9 ± 0.1	$+0.028 \pm 0.006$	0.80 ± 0.03
20% outliers	1.6 ± 0.2	$+0.051 \pm 0.010$	0.68 ± 0.05
Quadratic memory	0	0	1.000

4.2 Streaming classification

We next evaluate the adaptive method on Criteo, with 45.8M examples hashed to $d = 4{,}096$ (Criteo Labs, 2014). Each example is predicted before its update. Sequential online Newton and FlashRLS-GLM use the same local rule and fading $1 - 10^{-6}$. FlashRLS uses $C = 64$, fp32 recurrent state, and an fp64 core. FTRL and Adagrad provide first-order references (McMahan et al., 2013; Duchi et al., 2011). Progressive logloss and AUC use the final ten million rows.

Table 3: Streaming Criteo prediction. Logloss is mean $\pm$ standard deviation over ten hashing seeds; AUC uses three seeds. Time is seconds per million rows on one H100. Bold marks the best displayed value in each column, including ties.

Method	Logloss ↓	AUC ↑	Time (s) ↓
FTRL-Proximal	0.4716 ± 0.0006	0.7795 ± 0.0010	4
Adagrad-SGD	0.4698 ± 0.0008	0.7810 ± 0.0010	**3**
Sequential ONS, fp64	$\mathbf{0.4630 \pm 0.0003}$	$\mathbf{0.7885 \pm 0.0008}$	620
FlashRLS-GLM	$\mathbf{0.4630 \pm 0.0003}$	$\mathbf{0.7885 \pm 0.0008}$	16

On the same H100, FlashRLS matches sequential fp64 ONS logloss and AUC and reduces runtime from 620 to 16 seconds per million rows, a $38.8\times$ speedup (Table 3). The logloss improvement over Adagrad is 0.0068 (paired 95% CI ± 0.0006).

4.3 FlashRLS speedups over square-root QR updating

Square-root QRD-RLS stores R with $H = R^\top R$ and updates it as observations arrive. Table 4 compares custom GPU Householder, Givens, and column-pivoted QR updates with block Woodbury, including reads at every update and backward. Here B is batch size and n_h counts heads.

Table 4: FlashRLS speedup over tested GPU QR updates (↑): 3.9–$61.2\times$, increasing with feature width. Ratios are QR-update time divided by FlashRLS block-Woodbury time. Forward plus backward, reads at every update, $C = 64$, $B = n_\mathrm{h} = 8$, $N = 2{,}048$, and QR panel width $p = 32$. Token-sequential Sherman–Morrison costs $14.1\times$ at $d = 128$.

QR update / feature width d	64	128	256	512	1,024
Householder	$3.9\times$	$7.4\times$	$14.6\times$	$29.0\times$	$57.7\times$
Givens	$7.2\times$	$10.6\times$	$17.6\times$	$31.5\times$	$59.4\times$
Column-pivoted	$4.1\times$	$7.9\times$	$15.5\times$	$30.7\times$	$61.2\times$

FlashRLS's speedup over the tested Householder implementation grows from $7.4\times$ at $d = 128$ to $57.7\times$ at $d = 1{,}024$. The tested QR schedules add dependent stages as d grows; FlashRLS groups dense work around a fixed $C \times C$ core. Both use $O(C(d^2 + dm))$ dense work for $C \le d$. Appendix N explains the schedules and reports numerical stability under a separate protocol. QR updates also support adaptive differentiation.

4.4 Direct reads gain a speed advantage as conditioning worsens

At matched output accuracy, FlashRLS direct reads become faster than conjugate gradient (CG) as the information system becomes more ill-conditioned. We sweep the condition number κ with $d = m = 128$, $C = 64$, $B = n_\mathrm{h} = 8$, and $N = 2{,}048$. CG uses fp32 inputs and iterations and targets 10^{-4} relative output error. CG takes $0.56\times$ the direct-kernel time at $\kappa = 10^2$, but $1.1\times$ at 10^3

and $3.1\times$ at 10^4 (Figure 1). Both methods meet the accuracy target at these three condition numbers: CG is faster at 10^2, while direct reads are faster at 10^3 and 10^4.

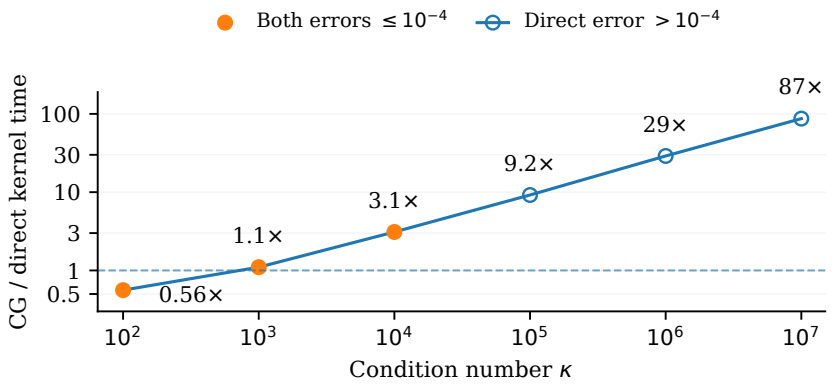

Figure 1: CG cost relative to FlashRLS direct reads as conditioning worsens. Ratios divide CG time to 10^{-4} relative output error by the 34.1 ms direct-kernel time; values above one favor FlashRLS in runtime (Table 44). Filled points meet the accuracy target for both methods. At open points, direct reads miss the target, so the ratios are not equal-accuracy speedups. At $\kappa = 10^8$, CG diverges and no ratio is plotted.

At $\kappa = 10^6$ and 10^7, direct reads exceed the 10^{-4} error target, with relative output errors of 6×10^{-4} and 2×10^{-3}, respectively. The corresponding CG/direct time ratios are $29\times$ and $87\times$, but these are not equal-accuracy speedups. Figure 1 marks these cases with open points. Appendix O gives the full error and recovery diagnostics.

Direct reads distinguish near-duplicate keys. Each group in this recall task contains eight nearly identical keys with distinct values, so successful retrieval requires resolving small differences between keys. At $\kappa \approx 10^6$, FlashRLS direct reads achieve $98 \pm 1\%$ recall, compared with $22 \pm 5\%$ for 30 iterations of CG (Table 46).

4.5 Read accuracy and throughput under weaker regularization

We compare read solvers in 50M Gated KalmaNet (Peng et al., 2026) as regularization weakens, holding architecture, gates, initialization, data order, and learning rate fixed. All rows train on one billion SlimPajama tokens (Soboleva et al., 2023), with $C = 64$ and shared chunk-frozen ridge $\lambda_c = a\|H_{[c]}\|_F$; smaller a means weaker regularization. CH-k denotes k Chebyshev iterations. Table 5 reports three paired seeds; Appendix L.5 gives the full solver study.

Table 5: 50M Gated KalmaNet: FlashRLS improves read accuracy; CH-30 has about $5\times$ higher throughput. Three paired seeds; quality and throughput are mean $\pm$ standard deviation. Rows share chunk-frozen ridge, bf16 inputs, and fp32 state; direct solves use an fp64 core. Read error compares identical inputs against fp64. Throughput is thousands of tokens/s.

Read solver	a	Slim PPL ↓	MQAR (%) ↑	Read error ↓	k tokens/s ↑
CH-30, bf16	0.01	24.48 ± 0.15	84 ± 3	5×10^{-3}	226 ± 4
FlashRLS, direct	0.01	24.41 ± 0.14	86 ± 3	5×10^{-5}	45 ± 1
CH-30, bf16	0.005	24.62 ± 0.20	83 ± 4	1.6×10^{-2}	226 ± 4
FlashRLS, direct + guard	0.005	24.33 ± 0.14	89 ± 2	5×10^{-5}	44 ± 1

At the stronger displayed ridge $a = 0.01$, CH-30 has 5×10^{-3} relative read error and about $5\times$ higher throughput. At $a = 0.005$, its error grows to 1.6×10^{-2}; exact solves recover 6.4 ± 1.2 MQAR points at 44 ± 1 versus 226 ± 4 thousand tokens/s. The guard rebuilds from fp64 statistics to correct roundoff, 2.7 ± 1.5 times per million chunks at this ridge. Table 26 also reports fp32 CH-100, which reaches similar accuracy at comparable cost.

4.6 ONLINE FILTERING AND FORECASTING

We measure two application workloads with 4,096-dimensional linear readouts using the squared-loss specialization. A numerical *guard* checks the inverse-state residual and rebuilds from fp64 statistics when needed. In the tables, core64/core32 denote the precision of the small factorization; recurrent states use fp32.

Echo cancellation: matching quality at $42.3\times$ speed. FlashRLS matches symmetrized Sherman–Morrison's 20.3 ± 0.7 dB steady echo return loss enhancement (ERLE) at 260 versus 11,000 ms/audio-s. We use 200 ten-second Microsoft AEC Challenge clips, 16 kHz audio, and a 4,096-tap filter (Sridhar et al., 2021). Guarding gives 20.6 ± 0.6 dB at 280 ms/audio-s, reducing failed diagnostic windows from 4 ± 3 to zero. FDKF is much faster at 0.3 ms/audio-s, but achieves lower ERLE (15.61 dB). Timings include the numerical policy and ridge maintenance: zero-target pseudo-observations replenish regularization lost through fading (Appendix M.1).

Table 6: Echo cancellation: FlashRLS matches sequential ERLE at $42.3\times$ speed; guarding eliminates observed diagnostic failures. Timings include ridge maintenance. Steady ERLE uses the final two seconds; $\pm$ values use three clip splits. FDKF: frequency-domain adaptive Kalman filter (Enzner & Vary, 2006).

Method	Policy	ERLE (dB) ↑	Failures ↓	ms/audio-s ↓
FlashRLS, core64	Free	20.3 ± 0.7	4 ± 3	260
FlashRLS, core64	Guarded	20.6 ± 0.6	0	280
FlashRLS, core32	Free	20.2 ± 0.7	12 ± 5	255
Symmetrized SM	Free	20.3 ± 0.7	6 ± 4	11,000
Joseph form	Free	20.3 ± 0.7	2 ± 2	12,000
FDKF	Gated	15.61	n/a	0.3

Chaotic-system readouts: $53.3\times$ faster fitting. FlashRLS matches sequential fp64 RLS forecast quality at 30 versus 1,600 s (Table 7). Batch ridge takes 2 s with slightly higher mean forecast quality, but fits all training data together; RLS retains predictors after each update. With teacher signals during rollout, FlashRLS reaches 5.1 ± 0.5 and 6.4 ± 0.6 Lyapunov times, versus batch ridge's 4.3 and 5.6. Appendix M.6 gives the full protocol.

Table 7: FlashRLS matches sequential RLS forecast quality with $53.3\times$ faster fitting. Readouts have 4,096 dimensions; fit times use 10^5 observations. Forecast quality is valid prediction time in Lyapunov units. Rollout updates use teacher signals when available.

Readout	Lorenz–96 ↑	Kuramoto–Sivashinsky ↑	Fit time (s) ↓
Batch ridge	4.3 ± 0.4	5.6 ± 0.5	2
Sequential RLS fp64	4.2 ± 0.4	5.5 ± 0.5	1,600
FlashRLS	4.2 ± 0.4	5.5 ± 0.5	30
FlashRLS, rollout updates	5.1 ± 0.5	6.4 ± 0.6	32

Discussion and conclusion. FlashRLS uses one block representation of the updates for exact execution and adaptive differentiation. The nonlinear experiments show the benefit of learning through curvature: the full backward improves associative recall and robust regression with the same adaptive forward rule. Streaming classification establishes acceleration of nonlinear online updates. Linear filtering and forecasting give application-level speedups at matched sequential prediction quality, and conditioning studies identify when accurate full-covariance updates matter. The resulting operator supports trainable online memories with response-dependent curvature and access to the state after every update.

Neural features, targets, and local-rule parameters can therefore be trained jointly through the complete online learner. Block size changes the execution schedule and preserves the trajectory.

Limitations. The matched-shape FlashRLS kernel runs slower than GDN. Dense state costs $O(d^2 + dm)$ and FD style sketch methods are left as future work. Though FlashRLS is faster, QR updates have lower inverse drift.

GENERATIVE AI USE STATEMENT

Generative AI tools were used to assist with manuscript editing and writing, primarily in the Introduction and Discussion/Conclusion, including improving clarity, organization, and phrasing. Generative AI was also used to assist with the presentation of figures and plots, including refining chart layout, labels, legends, and overall visual readability. All technical content, mathematical claims, experimental results, numerical values, and scientific conclusions were made by human authors.

ETHICS STATEMENT

This work studies the execution and differentiation of second-order online estimators. Its experiments use publicly released datasets as distributed and without modification: the Criteo display-advertising challenge logs (Criteo Labs, 2014), the Microsoft AEC Challenge speech recordings (Sridhar et al., 2021), SlimPajama (Soboleva et al., 2023), and synthetic data for the MQAR, Huber-regression, and chaotic-system studies. It involves no human subjects, no new data collection, and no personally identifying information beyond what those public corpora already contain.

REPRODUCIBILITY STATEMENT

Section 3 specifies the method: the transported interaction table and the local inputs it produces (Equation (3)), the triangular reconstruction of the update directions (Equation (4)), the prefix states and the two read maps (Equation (5) and Equation (6)), and the adaptive block procedure (Algorithm 1). Assumptions 2.1–2.2 state the conditions under which these hold, and Appendix G.8 treats their relaxation. Proposition 3.1 and Theorem 3.2 are proved in Appendix B; the backward and its exactness are given in Section 3.3 and Appendix G, with the nonsmooth threshold derivatives in Appendix G.7 and the piecewise case in Appendix G.6. Section 2 and Appendix K give the protocol for every study: the local rule and its parameters ($\delta = 0.5$ for Huber, the logistic information weight $\mathrm{sigmoid}(z_t)[1 - \mathrm{sigmoid}(z_t)]$), the block size, the fading factor, the ridge policy, and the arithmetic precision of the recurrence, the core, and the state. Per-study protocols are in Appendix M (MQAR and Huber), Appendix L.1 (Criteo), Appendix M.1 (echo cancellation), Appendix M.6 (chaotic readouts), and Appendix L.5 (learned memory), with the solver calibration in Appendix K.5 and the hardware for every timing in Appendix K. Seed counts are stated per experiment: the backward studies use ten paired seeds (Appendix P lists per-seed results), the separate timing study and the learned-memory comparison use three paired seeds (Appendix L.6), and each comparison pairs initialization and data order across arms. Appendix J.6 itemizes the storage terms behind each memory figure and Appendix J the layouts they assume. Numerical behavior is characterized separately: Appendix K.2 gives the conditioning and error-control analysis, Appendix J.5 the residual certificate, Appendix O the conditioning study, and Appendix K the complete result tables.

REFERENCES

American Statistical Association. Data expo 2009: Airline on-time performance. Joint Statistical Computing and Statistical Graphics Section Data Expo, 2009. URL https://community.amstat.org/jointscsg-section/dataexpo/dataexpo2009.

Robert Andrew and Nicholas J. Dingle. Implementing QR factorization updating algorithms on GPUs. *Parallel Computing*, 40(7):161–172, 2014. doi: 10.1016/j.parco.2014.03.003.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2312.04927.

Pierre Baldi, Peter Sadowski, and Daniel Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5:4308, 2014. doi: 10.1038/ncomms5308.

Gerald J. Bierman. *Factorization Methods for Discrete Sequential Estimation*. Academic Press, New York, 1977.

Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 443–448, 2007. doi: 10.1137/1.9781611972771.42.

Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive online analysis. *Journal of Machine Learning Research*, 11(52):1601–1604, 2010.

Jock Blackard. Covertype. UCI Machine Learning Repository, 1998.

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, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/228b9279ecf9bbafe582406850c57115-Abstract-Conference.html.

Adam W. Bojanczyk, James G. Nagy, and Robert J. Plemmons. Block RLS using row householder reflections. *Linear Algebra and its Applications*, 188–189:31–61, 1993. doi: 10.1016/0024-3795(93)90464-Y.

Jérôme Bolte, Tam Le, Edouard Pauwels, and Antonio Silveti-Falls. Nonsmooth implicit differentiation for machine learning and optimization. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://arxiv.org/abs/2106.04350.

Daniele Calandriello, Alessandro Lazaric, and Michal Valko. Second-order kernel online convex optimization with adaptive sketching. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 645–653, 2017. URL https://proceedings.mlr.press/v70/calandriello17a.html.

Adrien Corenflos. Robust automatic differentiation of square-root Kalman filters via Gramian differentials. arXiv:2603.13559, 2026. URL https://arxiv.org/abs/2603.13559.

Robert M. Corless, Gaston H. Gonnet, David E. G. Hare, David J. Jeffrey, and Donald E. Knuth. On the Lambert W function. *Advances in Computational Mathematics*, 5:329–359, 1996. doi: 10.1007/BF02124750.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006. URL https://jmlr.org/papers/v7/crammer06a.html.

Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems*, volume 22, pp. 414–422, 2009.

Criteo Labs. Criteo display advertising challenge dataset. Kaggle Display Advertising Challenge, 2014. URL https://www.kaggle.com/c/criteo-display-ad-challenge.

Lehel Csató and Manfred Opper. Sparse on-line Gaussian processes. *Neural Computation*, 14(3):641–668, 2002. doi: 10.1162/089976602317250933.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011. URL https://jmlr.org/papers/v12/duchi11a.html.

Yaakov Engel, Shie Mannor, and Ron Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, 2004. doi: 10.1109/TSP.2004.830985.

Gerald Enzner and Peter Vary. Frequency-domain adaptive Kalman filter for acoustic echo control in hands-free telephones. *Signal Processing*, 86(6):1140–1156, 2006. doi: 10.1016/j.sigpro.2005.09.013.

Sarah Filippi, Olivier Cappé, Aurélien Garivier, and Csaba Szepesvári. Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 4 edition, 2013. URL https://www.press.jhu.edu/books/title/10678/matrix-computations.

Michael Harries. Splice-2 comparative evaluation: Electricity pricing. Technical report, University of New South Wales, 1999.

Babak Hassibi, Ali H. Sayed, and Thomas Kailath. *Indefinite-Quadratic Estimation and Control: A Unified Approach to H^2 and H^∞ Theories*. SIAM, 1999. doi: 10.1137/1.9781611970760.

Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69:169–192, 2007. doi: 10.1007/s10994-007-5016-8.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2 edition, 2002.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024. doi: 10.48550/arXiv.2404.06654.

S. F. Hsieh, K. J. Ray Liu, and K. Yao. A unified square-root-free approach for QRD-based recursive least-squares estimation. *IEEE Transactions on Signal Processing*, 41(3):1405–1409, 1993. doi: 10.1109/78.205742.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35 (1):73–101, 1964. doi: 10.1214/aoms/1177703732.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

Thomas Kailath. An innovations approach to least-squares estimation: Part I: Linear filtering in additive white noise. *IEEE Transactions on Automatic Control*, 13(6):646–655, 1968. doi: 10.1109/TAC.1968.1099025.

Shinsuke Koyama, Lucia Castellanos Pérez-Bolde, Cosma Rohilla Shalizi, and Robert E. Kass. Approximate methods for state-space models. *Journal of the American Statistical Association*, 105 (489):170–180, 2010. doi: 10.1198/jasa.2009.tm08326.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Brian Kulis and Peter L. Bartlett. Implicit online learning. In *Proceedings of the 27th International Conference on Machine Learning*, pp. 575–582, 2010. URL https://icml.cc/Conferences/2010/papers/429.pdf.

Yoshiki Kuramoto and Toshio Tsuzuki. Persistent propagation of concentration waves in dissipative media far from thermal equilibrium. *Progress of Theoretical Physics*, 55(2):356–369, 1976. doi: 10.1143/PTP.55.356.

LAPACK. DGEQRF: Qr factorization of a general rectangular matrix. LAPACK 3.12.1 documentation, 2025. URL https://netlib.org/lapack/explore-html/d0/da1/group__geqrf_gade26961283814bb4e62183d9133d8bf5.html.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Edward N. Lorenz. Predictability: A problem partly solved. In *Proceedings of the Seminar on Predictability*, volume 1, pp. 1–18, Reading, UK, 1996. ECMWF.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1711.05101.

H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1222–1230, 2013. doi: 10.1145/2487575.2488200.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1609.07843.

Thomas P Minka. Expectation propagation for approximate bayesian inference. *arXiv preprint arXiv:1301.2294*, 2013.

Yann Ollivier. Online natural gradient as a Kalman filter. *Electronic Journal of Statistics*, 12(2): 2930–2961, 2018. doi: 10.1214/18-EJS1468.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3): 127–239, 2014. doi: 10.1561/2400000003.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0970.

Liangzu Peng, Aditya Chattopadhyay, Luca Zancato, Elvis Nunez, Wei Xia, and Stefano Soatto. Gated KalmaNet: A fading memory layer through test-time ridge regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026. URL https://arxiv.org/abs/2511.21016.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 20, pp. 1177–1184, 2007.

Simo Särkkä and Ángel F García-Fernández. Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2020.

Ali H. Sayed and Thomas Kailath. A state-space approach to adaptive RLS filtering. *IEEE Signal Processing Magazine*, 11(3):18–60, 1994.

Noam Shazeer. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. URL https://arxiv.org/abs/2002.05202.

G. I. Sivashinsky. Nonlinear analysis of hydrodynamic instability in laminar flames. i. derivation of basic equations. *Acta Astronautica*, 4(11–12):1177–1206, 1977. doi: 10.1016/0094-5765(77)90096-0.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R. Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627b token cleaned and deduplicated version of RedPajama. Cerebras Systems dataset release, 2023. URL https://www.cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama.

Kusha Sridhar, Ross Cutler, Ando Saabas, Tanel Parnamaa, Markus Loide, Hannes Gamper, Sebastian Braun, Robert Aichner, and Sriram Srinivasan. ICASSP 2021 acoustic echo cancellation challenge: Datasets, testing framework, and results. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 151–155, 2021. doi: 10.1109/ICASSP39728.2021.9413457.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 57503–57522, 2025. URL https://proceedings.mlr.press/v267/sun25h.html.

Kevin Tracy. A square-root Kalman filter using only QR decompositions. arXiv:2208.06452, 2022. URL https://arxiv.org/abs/2208.06452.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Sarthak Mittal, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Rif A. Saurous, Guillaume Lajoie, Charlotte Frenkel, Razvan Pascanu, Blaise Agüera y Arcas, and João Sacramento. MesaNet: Sequence modeling by locally optimal test-time training. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2506.05233.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: A unifying framework for designing sequence models with associative memory. *arXiv preprint arXiv:2501.12352*, 2025.

Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

Fatemeh Yaghoobi, Adrien Corenflos, Sakira Hassan, and Simo Särkkä. Parallel square-root statistical linear regression for inference in nonlinear state space models. *SIAM Journal on Scientific Computing*, 47(2):B454–B476, 2025. doi: 10.1137/23M156121X.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522, 2024.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 12381–12392, 2019.

Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings. Mathematical, physical, and engineering sciences*, 473(2205):20170457, 2017.

A RECURSIVE LEAST SQUARES AND FIXED WEIGHTED GEOMETRY

We now derive the squared-loss specialization and prove the state identity in Corollary 3.5. Throughout, $H_0 \succ 0$, $P_0 = H_0^{-1}$, and the observation order is fixed.

A.1 QUADRATIC OBJECTIVE AND SEQUENTIAL COEFFICIENTS

For unit observation noise, define

$$\mathcal{J}_c(W) = \frac{1}{2}\operatorname{tr}[(W - W_0)H_0(W - W_0)^\top] + \frac{1}{2}\sum_{t\le c}\|y_t - Wk_t\|^2. \tag{12}$$

Let Y have rows $y_t^\top$, and set

$$H_c = H_0 + K_c^\top K_c, \qquad B_c = W_0 H_0 + Y_c^\top K_c.$$

The derivative of Equation (12) is $WH_c - B_c$, so its unique minimizer is $W_c = B_c P_c$. With $r_t = y_t - p_t$ and $d_t = 1 + s_t$, the local coefficients are

$$w_t = 1, \quad \beta_t = d_t^{-1}, \quad u_t = r_t/d_t, \quad g_t = v_t/d_t. \tag{13}$$

Thus $W_t = W_{t-1} + r_t g_t^\top$ and $P_t = P_{t-1} - d_t g_t g_t^\top$. This is the conventional full-matrix RLS recurrence (Sayed & Kailath, 1994).

A.2 ONE FACTORIZATION RECOVERS EVERY STATE

Set

$$S = I + KP_0K^\top = TDT^\top, \qquad D = \operatorname{diag}(d_1, \dots, d_C), \tag{14}$$

with T unit lower triangular and D positive diagonal. Since $S \succeq I$, unpivoted $LDL^\top$ exists. The leading $c \times c$ block satisfies

$$S_c = I + K_c P_0 K_c^\top = T_c D_c T_c^\top.$$

This follows directly by multiplying the leading blocks of a triangular product. Likewise, the leading rows of a lower-triangular solve depend only on the corresponding leading equations.

For completeness, the pivot at time t is the Schur complement

$$d_t = 1 + k_t^\top P_0 k_t - k_t^\top P_0 K_{t-1}^\top S_{t-1}^{-1} K_{t-1} P_0 k_t.$$

The matrix identity

$$(H_0 + K_c^\top K_c)^{-1} = P_0 - P_0 K_c^\top (I + K_c P_0 K_c^\top)^{-1} K_c P_0 \tag{15}$$

is verified by multiplication with $H_0 + K_c^\top K_c$. Inserting it into the Schur complement gives $d_t = 1 + k_t^\top P_{t-1} k_t$.

Define the gain and innovation-residual arrays

$$G = D^{-1}T^{-1}KP_0, \qquad R = T^{-1}(Y - KW_0^\top). \tag{16}$$

The row recurrence for the first solve is Equation (28), so $G_{t,:} = g_t^\top$. The second solve gives

$$R_t = y_t - W_0 k_t - \sum_{j<t}(k_t^\top g_j) R_j.$$

Induction through the mean update identifies $R_t = r_t$. Consequently,

$$P_c = P_0 - G_c^\top D_c G_c, \qquad W_c = W_0 + R_c^\top G_c, \tag{17}$$

$$O_{\text{pre}} = QW_0^\top + \operatorname{tril}(QG^\top, -1)R. \tag{18}$$

This derivation makes the relationship to the general interface explicit: $A = DG$ and $U = D^{-1}R$.

A.3 THE UNIT-TRIANGULAR TRANSFORM AND SELF-QUERIES

Equation (31) becomes

$$T = I + \mathrm{tril}(KG^\top, -1), \qquad \mathrm{diag}(KG^\top) = \mathbf{1} - D^{-1}\mathbf{1}. \tag{19}$$

The off-diagonal identity follows from $T_{ij} = \beta_j M_{ij}$. The diagonal follows from $k_t^\top g_t = s_t/(1+s_t)$. It is the triangular structure used by the residual solve. Delta-rule chunk algorithms motivate this implementation interface (Yang et al., 2024).

For $Q = K$, the outputs simplify to

$$O_{\mathrm{pre}}(K) = Y - R, \qquad O_{\mathrm{post}}(K) = Y - D^{-1}R. \tag{20}$$

The same arrays give $k_t^\top P_{t-1} = d_t g_t^\top$, $k_t^\top P_t = g_t^\top$, and the two variances $s_t = d_t - 1$ and $s_t/(1+s_t) = 1 - d_t^{-1}$.

A.4 KNOWN POSITIVE OBSERVATION WEIGHTS

For information weights $\omega_t \geq 0$, the objective replaces each squared loss by $\omega_t \|y_t - W k_t\|^2/2$. The coefficients are

$$w_t = \omega_t, \qquad u_t = \frac{\omega_t(y_t - p_t)}{1 + \omega_t s_t}. \tag{21}$$

For $\omega_t > 0$, the observation-noise variance is $1/\omega_t$. Zero weights leave the state unchanged for this quadratic rule. Rescaling $\widetilde{k}_t = \sqrt{\omega_t} k_t$ and $\widetilde{y}_t = \sqrt{\omega_t} y_t$ reduces prescribed nonnegative weights to Equation (14). The general online interface also handles weights selected from the current local inputs. Such adaptive weights specify an online reweighted regression trajectory.

For nonzero weights of either sign, define $J = \mathrm{diag}(w_t)$ and $S_J = J^{-1} + \Gamma$. Its ordered factorization has pivots $\delta_t = w_t^{-1} + s_t$, with $\beta_t = \delta_t^{-1}$. The zero-safe representation in Appendix B supplies the implementation at vanishing weights. Appendix I covers signed feasibility and inference.

A.5 BALANCED CHOLESKY COORDINATES

The sequential innovation variance, gain, and residual satisfy

$$L_{tt}^2 = d_t, \qquad X_{t,:} = \sqrt{d_t}\, g_t^\top, \qquad E_{t,:} = r_t^\top/\sqrt{d_t}. \tag{22}$$

A Cholesky factor L of S satisfies $L = TD^{1/2}$. Let

$$X = L^{-1}KP_0 = D^{1/2}G, \qquad E = L^{-1}(Y - KW_0^\top) = D^{-1/2}R.$$

Then $P_C = P_0 - X^\top X$ and $W_C = W_0 + E^\top X$. These balanced factors distribute the innovation scaling between the two arrays. Both $LDL^\top$ and Cholesky implementations compute the same exact states. Their finite-precision behavior depends on factors, scaling, and backend (Higham, 2002).

The diagonal row scaling commutes with selection by any update-selection mask Π:

$$\big[(QG^\top) \odot \Pi\big]R = \big[(QX^\top D^{-1/2}) \odot \Pi\big]D^{1/2}E = \big[(QX^\top) \odot \Pi\big]E.$$

The same cancellation applies to each intermediate state correction. Together with $L_{tt}^2 = d_t$, this proves Corollary 3.5 and the row interpretation in Equation (22) in the main text's balanced coordinates.

A.6 A TWO-OBSERVATION EXAMPLE

Set $P_0 = 1$, $W_0 = 0$, $K = (1,1)^\top$, and $Y = (2,2)^\top$. Then

$$L = \begin{pmatrix} \sqrt{2} & 0 \\ 1/\sqrt{2} & \sqrt{3/2} \end{pmatrix}, \qquad X = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{6} \end{pmatrix}, \qquad E = \begin{pmatrix} \sqrt{2} \\ \sqrt{2/3} \end{pmatrix}.$$

The two mean corrections are 1 and $1/3$. For self-queries, the strict mask emits $(0,1)^\top$, and the inclusive mask emits $(1, 4/3)^\top$. The terminal mean is $4/3$, and the terminal covariance is $1/3$. The second target has been converted into the residual left after the first update.

B SEQUENTIAL UPDATE IDENTITIES

Proposition 3.1, Theorem 3.2, and Proposition 3.7 are restated below with their detailed proofs. In this appendix we derive the response-adaptive transport of Section 3.1 and prove agreement with the sequential update. Coordinates are fixed within each chunk. Learned keys and queries are evaluated before the online updates, and any local randomness is coupled across implementations.

B.1 NOTATION AND THE SEQUENTIAL ALGORITHM

Symbol	Shape	Meaning
K, Q	$C \times d$	Keys and one query per update time
H_t, P_t	$d \times d$	Information matrix and its inverse
W_t	$m \times d$	Predictor after update t
v_t	d	Adapted direction $P_{t-1}k_t$
p_t, u_t	m	Pre-update prediction and mean coefficient
s_t, w_t, β_t	scalar	Uncertainty, information weight, covariance coefficient
Γ, M, T	$C \times C$	Entry Gram, ordered interactions, transport factor
A, U	$C \times d,\ C \times m$	Adapted directions and mean coefficients

Let $H_0 \succ 0$, $P_0 = H_0^{-1}$, and $W_0 \in \mathbb{R}^{m\times d}$. The sequential recurrence is

$$v_t = P_{t-1}k_t, \qquad s_t = k_t^\top v_t, \qquad p_t = W_{t-1}k_t, \tag{23}$$

$$(u_t, w_t, \xi_t) = \mathcal{F}_t(p_t, s_t, y_t, \xi_{t-1}), \tag{24}$$

$$H_t = H_{t-1} + w_t k_t k_t^\top, \qquad W_t = W_{t-1} + u_t v_t^\top. \tag{25}$$

Here ξ_t is optional auxiliary state whose size and update cost are included in the local rule. Its inputs must be available from the local quantities and specified exogenous data. Assume

$$h_t := 1 + w_t s_t > 0, \qquad \beta_t = \frac{w_t}{h_t}. \tag{26}$$

Here h_t is the scalar feasibility pivot, and H_t denotes the information matrix.

B.2 POSITIVE DEFINITENESS AND THE ADAPTED DIRECTION

Write $r = H_{t-1}^{-1/2}k_t$. Congruence gives

$$H_t = H_{t-1}^{1/2}(I + w_t r r^\top)H_{t-1}^{1/2}.$$

The matrix in parentheses has eigenvalue $1 + w_t\|r\|^2 = h_t$ in the direction r and eigenvalue one on its orthogonal complement. Thus Equation (26) is exactly the condition that preserves positive definiteness at this step. Direct multiplication verifies

$$P_t = P_{t-1} - \beta_t v_t v_t^\top. \tag{27}$$

At $w_t = 0$, the coefficient β_t is zero and the metric is unchanged. The mean rule permits a nonzero u_t in this fixed-metric case.

Repeated substitution in Equation (27) gives

$$v_j = P_0 k_j - \sum_{\ell<j} \beta_\ell v_\ell (v_\ell^\top k_j). \tag{28}$$

The coefficient of an earlier direction depends on its interaction with the current key. This is the quantity stored in the interaction table.

B.3 THE EXAMPLE-SPACE RECURRENCE

Define $\Gamma = KP_0K^\top$ and $a = KW_0^\top$, with rows $a_t^\top$. For $i \geq j$, set

$$M_{ij} = \Gamma_{ij} - \sum_{\ell<j} \beta_\ell M_{i\ell} M_{j\ell}. \tag{29}$$

Set entries above the diagonal to zero. At stage j, compute

$$s_j = M_{jj}, \qquad p_j = a_j + \sum_{\ell<j} M_{j\ell} u_\ell, \tag{30}$$

and call Equation (24). The column $M_{j:C,j}$ uses only earlier information weights. The current weight w_j first enters subsequent columns.

Proposition B.1 (Exact local inputs)**.** *Equations* (29)–(30) *give* $M_{ij} = k_i^\top P_{j-1} k_j$ *for* $i \geq j$*, and reproduce all inputs and outputs of the sequential local rules.*

Proof. At $j = 1$, the asserted interaction is Γ_{i1}. Suppose the preceding columns and local outputs agree. Taking the inner product of Equation (28) with k_i gives Equation (29). Its diagonal is $k_j^\top P_{j-1} k_j$. Telescoping the mean update gives $W_{j-1} k_j = W_0 k_j + \sum_{\ell<j} u_\ell v_\ell^\top k_j$, which is Equation (30). A well-defined local rule now returns the same (u_j, w_j, ξ_j) in both computations. Induction completes the argument. □

B.4 Recovering the directions and states

Let $\mathcal{B} = \mathrm{diag}(\beta_1, \ldots, \beta_C)$ and let U contain rows $u_t^\top$. Define

$$T = I + \mathrm{tril}(M, -1)\mathcal{B}, \qquad TA = KP_0. \tag{31}$$

The factor T is unit lower triangular. The transposed row recurrence in Equation (28) is exactly this triangular system. Row t of A is therefore $v_t^\top$.

Theorem B.2 (States and products at every update)**.** *For each update index* c*, let* $A_c = A_{1:c,:}$*,* $U_c = U_{1:c,:}$*, and* $\mathcal{B}_c = \mathrm{diag}(\beta_1, \ldots, \beta_c)$*. Then*

$$P_c = P_0 - A_c^\top \mathcal{B}_c A_c, \tag{32}$$

$$W_c = W_0 + U_c^\top A_c. \tag{33}$$

For queries with rows $q_t^\top$*, the pre-update outputs and inverse actions are*

$$O_{\mathrm{pre}} = QW_0^\top + \mathrm{tril}(QA^\top, -1)U, \tag{34}$$

$$Z_{\mathrm{pre}} = QP_0 - \mathrm{tril}(QA^\top, -1)\mathcal{B}A. \tag{35}$$

Including the diagonal of the mask gives the corresponding post-update quantities.

Proof. Summing Equations (25) and (27) gives Equations (32)–(33). For a query at time t, use the state after update $t-1$ into these identities and multiply by q_t. Stacking the resulting sums gives Equations (34)–(35). Every sum contains precisely the permitted earlier indices. □

Queries can also be assigned to any update index $c(i)$. Replacing the triangular mask by $L_{ij} = \mathbf{1}\{j \leq c(i)\}$ returns $W_{c(i)} q_i$ for any fixed collection of such queries. The query vectors themselves must use only information available at their assigned update.

B.5 An implementation using trailing updates

The following recurrence implements Equation (29) with matrix updates. Initialize $G^{(0)} = \Gamma$ and $b^{(0)} = a$. At stage t, copy

$$c = G^{(t-1)}_{t:C,t}, \qquad s_t = c_1, \qquad p_t = b_t^{(t-1)}.$$

Store $M_{t:C,t} = c$, call the local rule, and update the remaining indices $F = \{t+1, \ldots, C\}$:

$$G^{(t)}_{F,F} = G^{(t-1)}_{F,F} - \beta_t c_{2:} c_{2:}^\top, \tag{36}$$

$$b_F^{(t)} = b_F^{(t-1)} + c_{2:} u_t^\top. \tag{37}$$

For static coordinates, $G^{(t)}_{F,F} = K_F P_t K_F^\top$. This invariant also provides a direct implementation check. The intermediate states in Theorem B.2 are materialized only for verification or an application that explicitly requests them.

B.6 Arithmetic, storage, and ordering

Preparing KP_0 and $KW_0^\top$ costs $O(Cd^2 + Cdm)$. Forming Γ costs $O(C^2 d)$. The trailing updates cost $O(C^3 + C^2 m)$ and require C local-rule calls. Recovering A costs $O(C^2 d)$. Generic query contractions cost $O(Cdm + C^2(d+m))$, and committing the boundary costs $O(Cd^2 + Cdm)$. These counts use one $C \times C$ interaction table; bounded blocks give Equation (11).

The interaction core uses $O(C^2 + Cm)$ workspace, plus local-solver storage. The full forward pass uses $O(C^2 + C(d+m))$ temporary storage and $O(d^2 + md)$ persistent state. This bound concerns the representation in a conventional dense implementation. Actual memory traffic depends on tiling, caching, and kernel fusion. Learned features and auxiliary neural modules add their own costs.

Chunk partitioning preserves the observation order and the incoming state of each chunk. Proposition B.1 therefore proves partition invariance. Permuting observations changes the ordered local problems and generally changes the online trajectory, including in exact arithmetic.

B.7 Exact bounded-block execution and context-length scaling

The chunk size in an algebraic identity and the block size used by its implementation are distinct choices. Let C denote a logical chunk and partition its chronological observations into blocks of sizes $b_j \le b$, with $\sum_j b_j = C$. A numerical block here includes a boundary-state commit. A storage tile used inside a monolithic factorization only partitions memory traffic; it has no such state transition.

Proposition B.3 (Exact bounded-block cost)**.** *Assume the local rule is well-defined, the required pivots are positive, and every block receives the committed state of its predecessor. With a constant number of reads per observation, each using only information available at that step, the partitioned evaluation reproduces the original sequential trajectory. Its arithmetic work is*

$$O\big(C(d^2 + dm + b(d+m) + b^2) + T_{\mathrm{local}}\big)\,, \tag{38}$$

where T_{local} is the sum of local-rule costs. Persistent state has $O(d^2 + md)$ entries and reusable forward workspace has $O(b^2 + b(d+m))$ entries.

Proof. The first block starts from the original state, so Proposition B.1 gives identical local responses and states throughout that block. Suppose all preceding blocks agree. The next block then receives the same incoming state and the same observations, giving agreement after every update. Induction proves agreement throughout the logical chunk. The read formula contracts each query against the state after its assigned update.

A block of length b_j costs

$$O\big(b_j(d^2 + dm) + b_j^2(d+m) + b_j^3 + T_{\mathrm{local},j}\big)\,.$$

The inequalities $\sum_j b_j^2 \le bC$ and $\sum_j b_j^3 \le b^2 C$ give Equation (38). The forward computation reuses one block's workspace and the committed boundary state. Differentiating the composed block maps has the same dense arithmetic order, together with the local derivatives and the selected checkpoint/recomputation cost. □

At $b \le d$, both $b(d+m)$ and b^2 are bounded by a constant multiple of $d^2 + dm$. Squared-loss execution therefore costs $O(C(d^2 + dm))$ for a logical chunk and $O(N(d^2 + dm))$ for a sequence of length N. Internal block length can remain fixed as N increases. The chunk sweeps in Appendix K use $b = C$. Holding b fixed preserves linear work in context length.

A structured view of the same innovation factor. The innovation matrix itself also has a structured evaluation. For fixed squared-loss inputs, put $v_t = P_{t-1}k_t$ and $a_t = \sqrt{1 + k_t^\top v_t}$. The Cholesky entries are

$$L_{tt} = a_t, \qquad L_{it} = k_i^\top v_t / a_t \quad (i > t).$$

Indeed, eliminating observations $1, \dots, t-1$ leaves the Schur complement $I + K_{t:C} P_{t-1} K_{t:C}^\top$. Its first pivot and column give the displayed expressions. Computing the v_t through feature-space rank-one updates costs $O(Cd^2)$; explicitly emitting the remaining factor entries costs $O(C^2 d)$. Thus the structured factor can be formed in $O(Cd^2 + C^2 d)$, which is quadratic in C for fixed d. The

normalized directions $X_{t,:} = v_t^\top / a_t$ and residuals $E_{t,:} = (y_t - W_{t-1}k_t)^\top / a_t$ can also be emitted directly. This rowwise construction exposes the chronological dependency. Bounded-block FlashRLS groups the exact work into matrix products and local factors without storing a context-scale factor.

B.8 WHERE THE QUADRATIC WORK IS GAINED

Here C is the number of observations in a chunk, d is the per-head feature width, and m is the response width. Table 8 separates the costs of alternative execution routes. Counts exclude feature generation and include only the output or subproblem named in the table. They are algebraic operation counts, not new timing measurements.

Table 8: **Arithmetic for one chunk and for its subproblems.** T_{local} includes the iterations used by all local response solvers. A numerical block of length b includes a state commit.

Computation	Arithmetic work
Fresh dense feature-space factorization and fit after every update	$O(C(d^3 + d^2 m))$, given accumulated normal equations
Tokenwise rank-one RLS	$O(C(d^2 + dm))$
Shared dense innovation block	$O(C(d^2 + dm) + C^2(d + m) + C^3)$
Exact bounded-block execution	$O(C(d^2 + dm + b(d + m) + b^2) + T_{\mathrm{local}})$
Structured explicit innovation factor	$O(Cd^2 + C^2 d)$, retaining chronological direction updates
Fixed-geometry scalar Newton direction or implicit adjoint	$O(C^2)$, after the geometry has been formed
Fully response-adaptive interaction core	$O(C^3 + C^2 m) + T_{\mathrm{local}}$

The first two rows compare refitting a dense system with reusing its rank-one change. This is the classical reduction from cubic feature-width work to quadratic update work. FlashRLS preserves the rank-update arithmetic order and reorganizes the computation into shared factors and matrix products.

For the shared block, multiplying a $C \times d$ array by a dense $d \times d$ state costs $O(Cd^2)$. Forming the interaction table costs $O(C^2 d)$, its Cholesky factor costs $O(C^3)$, and solving for $d + m$ right-hand sides costs $O(C^2(d + m))$. Reads and commits give the remaining displayed terms. Dividing the block cost by C gives a factor contribution of $O(C^2)$ per observation and $O(C^3)$ per block. The inequalities in Proposition B.3 give the bounded-block row and show that $b \leq d$ recovers $O(d^2 + dm)$ per observation.

The scalar nonlinear system in Appendix F has a lower-triangular Jacobian when geometry is fixed. One Newton direction uses forward substitution and an implicit adjoint uses backward substitution, each touching $O(C^2)$ coefficients. The iteration count for obtaining a nonlinear solution is an additional cost. Response-adaptive geometry requires the extra interaction updates of Appendix B; their dense total is cubic in C and is included above. The local nonlinear equations have response dimension m, so their solve cost is determined in response space.

The measured speedups in Section 4.6 establish the execution benefit for the tested implementations.

Related work. Classical RLS and square-root filtering maintain the predictor and its accumulated information through rank-one updates or triangular factors (Kailath, 1968; Bierman, 1977; Sayed & Kailath, 1994). Block Householder RLS and GPU QR updating organize factor operations into larger matrix computations (Hsieh et al., 1993; Bojanczyk et al., 1993; Andrew & Dingle, 2014). Appendix N derives their correspondence with the states used here, including access to predictions at intermediate updates. These methods establish the factor-based foundations for the squared-loss case. FlashRLS represents the ordered update directions in observation space and uses the same representation for response-dependent information weights.

Online Newton, recursive Laplace, and implicit online learning supply local rules beyond squared loss (Hazan et al., 2007; Koyama et al., 2010; Kulis & Bartlett, 2010). Appendix C derives the response-space proximal problem, and Appendices D and E specify scalar and vector rules. The present construction preserves the selected rule, including the effect of an early response on later information weights. Its backward differentiates these dependencies through the interaction recurrence and local solutions.

Learned regression memories use features and targets produced by a surrounding neural model (Sun et al., 2025; Wang et al., 2025). DeltaNet and Gated DeltaNet organize delta-rule updates into chunked matrix operations (Yang et al., 2024; 2025). MesaNet and Gated KalmaNet use conjugate-gradient and Chebyshev reads (von Oswald et al., 2026; Peng et al., 2026). Table 9 distinguishes their update targets, solver gradients, ordered operations, and stored states. The matched-ridge study in Section 4.5 compares the read solvers within one model. The language-model comparisons in Appendix L report the modeling and execution tradeoffs of the tested memories.

Parallel smoothing and square-root inference also organize inference through factor-based computations (Särkkä & García-Fernández, 2020; Yaghoobi et al., 2025). Gramian-based differentiation gives derivatives of square-root filter outputs in physical state coordinates (Corenflos, 2026). Appendix N explains the corresponding state-coordinate chain rule for the operator studied here. The exact-gradient claim in Theorem 3.4 concerns the full sequential learner, including response-dependent curvature and local-solver derivatives.

Table 9: Recurrence and execution structure. Exactness is in real arithmetic for the named target; C is block length. The QRD row describes the measured squared-loss implementation; adaptive factor updates are discussed in Appendix N.

Method	Trajectory	Curvature / gradient	Sequential dependence per block	State
Tokenwise rule	Exact dense	Adaptive; exact reverse	C ordered dense rank-one updates	$O(d^2 + dm)$
Block QRD-RLS	Exact RLS	Fixed; QR backward	Block QR; ordered state solves	$O(d^2 + dm)$
Chunked DeltaNet	Exact delta rule	First-order; exact reverse	Chunked triangular solve	$O(dm)$
CG / Chebyshev reads	Approximate dense solve	Finite-iteration gradient	Solver iterations per read	$O(d^2 + dm)$
FlashRLS	**Exact dense**	**Adaptive; exact reverse**	**C local calls + core updates; batched dense work**	$O(d^2 + dm)$

C Nonlinear proximal learning in innovation coordinates

This appendix reduces a parameter-space proximal update to an equation in the current prediction. We then state the ordered system, its well-posedness conditions, and its relation to the batch objective.

The local problem. A proximal learner chooses

$$W^+ = \arg\min_W \left\{ c\ell(Wk; y) + \tfrac{1}{2} \operatorname{tr}[(W - W^-)H(W - W^-)^\top] \right\}. \tag{39}$$

Put $p = W^- k$ and $s = k^\top P k$. Minimizing first over the response gives, for $s > 0$,

$$z = \operatorname{prox}_{cs\ell}(p) := \arg\min_z \{ \tfrac{1}{2}\|z - p\|^2 + cs\ell(z) \}, \qquad u = (z - p)/s. \tag{40}$$

The change is $u(Pk)^\top$, so the nonlinear problem has response dimension m. Proper closed convex losses have a unique proximal solution; for a ρ-weakly convex loss it suffices that $cs\rho < 1$ (Section C.4). If $s = 0$, then $k = 0$ and the state is unchanged.

For scalar logistic loss, $z + cs[\operatorname{sigmoid}(z) - y] = p$ is a monotone scalar equation. A recursive Laplace rule uses $u = -c[\operatorname{sigmoid}(z) - y]$ and $w = c \operatorname{sigmoid}(z)[1 - \operatorname{sigmoid}(z)]$. The response therefore determines the curvature used in later updates. Online-Newton and robust rules use the same interface (Appendix D).

C.1 Reduction from parameters to the current prediction

Let $H \succ 0$, $P = H^{-1}$, $p = W_- k$, and $s = k^\top P k > 0$. For a proper closed convex loss $\ell : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ and $c > 0$, consider

$$W_+ = \arg\min_W \left\{ \frac{1}{2} \operatorname{tr}[(W - W_-)H(W - W_-)^\top] + c\ell(Wk) \right\}. \tag{41}$$

Fix a proposed prediction $z = Wk$ and write $\Delta = W - W_-$. The least expensive parameter change satisfying $\Delta k = z - p$ is

$$\Delta = \frac{z-p}{s}(Pk)^\top. \tag{42}$$

To see this, decompose any feasible change as this expression plus N with $Nk = 0$. The cross term in its squared H-norm is proportional to $(z-p)^\top Nk = 0$. The first term has squared norm $\|z-p\|^2/s$, and the added term has nonnegative squared norm. Therefore

$$z = \arg\min_{\zeta} \left\{ \frac{1}{2}\|\zeta - p\|^2 + cs\ell(\zeta) \right\} = \mathrm{prox}_{cs\ell}(p), \tag{43}$$

$$u = (z-p)/s, \qquad W_+ = W_- + u(Pk)^\top. \tag{44}$$

The response objective is strongly convex. For differentiable ℓ, its unique minimizer satisfies

$$z + cs\nabla\ell(z) = p, \qquad u = -c\nabla\ell(z). \tag{45}$$

For a nonsmooth loss, replace the gradient by a subgradient satisfying this inclusion. The vector $(p-z)/(cs)$ selects the needed subgradient. The coefficient $c = 0$ defines the identity update. Under $H \succ 0$, $s = 0$ implies $k = 0$; a finite loss at the resulting prediction leaves the state unchanged. These are standard proximal constructions (Kulis & Bartlett, 2010; Parikh & Boyd, 2014) with a specific transported metric.

C.2 Explicit and implicit update equations

Let ψ_t be a chosen gradient or subgradient map. With c_t fixed during local evaluation, define

$$\zeta_t = p_t \quad \text{for an explicit update,} \qquad \zeta_t = W_t k_t \quad \text{for an implicit update.}$$

The mean coefficient is $u_t = -c_t\psi_t(\zeta_t)$. Inserting this into Equation (30) gives

$$\zeta_t + \theta c_t s_t \psi_t(\zeta_t) = a_t - \sum_{j<t} M_{tj} c_j \psi_j(\zeta_j), \qquad \theta \in \{0, 1\}. \tag{46}$$

Here $\theta = 0$ selects explicit evaluation and $\theta = 1$ selects the proximal equation. Define $D_c = \mathrm{diag}(c_t)$. For fixed M, the two systems use $\mathrm{tril}(M, -1)$ and M, respectively. With adaptive curvature, M is evaluated along the resulting trajectory and carries that trajectory's earlier weight choices.

The coefficient c_t may depend on the available p_t, s_t, y_t and auxiliary state before the local solve. A coefficient defined through the unknown solution creates a different local equation. Its existence and derivative are part of the local-rule specification.

C.3 Fixed unit geometry and the RLS factor

Suppose $H_t = H_{t-1} + k_t k_t^\top$, independently of the nonlinear response loss. Let $S = TDT^\top$ be the factorization in Appendix A, and $A = T^{-1}KP_0$. Since $\beta_t = 1/d_t$ and $s_t = d_t - 1$,

$$M = TD - I. \tag{47}$$

The diagonal of the right side is s_t, and its strictly lower entries are $T_{ij}d_j = M_{ij}$. For $c_t = 1$, stacking the implicit equations gives

$$Z + (TD - I)\Psi(Z) = KW_0^\top, \tag{48}$$

where row t of $\Psi(Z)$ is $\nabla\ell_t(z_t)^\top$. The unknown has shape $C \times m$. Once it is solved, the states after each update use $U = -\Psi(Z)$ in Theorem B.2.

For squared loss $\Psi(Z) = Z - Y$, Equation (48) gives

$$Z = Y - D^{-1}T^{-1}(Y - KW_0^\top),$$

which is exactly the post-update RLS prediction. This calculation identifies RLS as an explicitly solvable member of the response-space family.

C.4 WEAKLY CONVEX LOSSES

A proper closed loss is ρ-weakly convex when $\ell(z) + \rho\|z\|^2/2$ is convex. For $\alpha = cs$,

$$\frac{1}{2}\|z-p\|^2 + \alpha\ell(z) = \frac{1-\alpha\rho}{2}\|z\|^2 - p^\top z + \alpha\left(\ell(z) + \frac{\rho}{2}\|z\|^2\right) + \frac{1}{2}\|p\|^2.$$

Thus

$$\alpha\rho < 1 \tag{49}$$

is a sufficient condition for strong convexity and a unique local solution. For a twice differentiable scalar loss, the derivative of the local equation is bounded below by $1 - \alpha\rho$. A globally positive uniform bound also supplies a residual-to-solution error bound. Additional solutions or well-posed regimes can exist outside this sufficient condition.

If the information weight is selected as $w = c\ell''(z)$, Equation (49) also yields $1 + ws \geq 1 - cs\rho > 0$. It therefore certifies both the local mode and the Gaussian information update. A local check of s is important when previous signed updates or process noise can increase covariance.

C.5 RELATION TO THE BATCH OBJECTIVE

The batch problem associated with the same initial metric is

$$\min_W \frac{1}{2}\|W - W_0\|_{H_0}^2 + \sum_{t=1}^{C} \ell_t(Wk_t). \tag{50}$$

For differentiable losses, stationarity gives

$$W = W_0 - \Psi(Z)^\top K P_0, \qquad Z = KW^\top,$$

so

$$Z + \Gamma\Psi(Z) = a. \tag{51}$$

This equation uses the complete Gram matrix of the observations. The ordered system in Equation (46) describes the sequence of local online problems and its transported interactions. Each has a precise optimization target. For general nonlinear losses the two targets define their own solutions.

A useful batch coordinate system follows from $\Gamma = RR^\top$, where R has full column rank on the range of Γ. For a scalar output, set $z = a + Rb$. Then the batch stationarity equation is equivalent to

$$b + R^\top\psi(a + Rb) = 0. \tag{52}$$

It is the gradient of $\|b\|^2/2 + \sum_t \ell_t(a_t + (Rb)_t)$, a strongly convex objective for convex losses. Its Jacobian is $I + R^\top \operatorname{diag}(\ell_t'')R \succeq I$. This is the appropriate monotone coordinate form, related to monotone equilibrium formulations (Winston & Kolter, 2020). A dense Newton step costs up to $O(C^3)$, together with the likelihood evaluations. The ordered system admits ordered local solves, with existence inherited from their individual well-posedness.

D SCALAR LOSSES AND ADAPTIVE LOCAL RULES

We give the scalar local rules used by the transport in Appendix B. Write $p = W_{t-1}k$, $s = k^\top P_{t-1}k$, and $v = P_{t-1}k$. For one output, $W^\top = \mu \in \mathbb{R}^d$, and the update is $\mu^+ = \mu + uv$, $H^+ = H + wkk^\top$. Each rule specifies the mean coefficient u and information weight w. We use $\alpha = cs$ for a nonnegative likelihood or step weight c held fixed during the local solve. At $s = 0$, positive definiteness gives $k = 0$ and the state is unchanged.

D.1 QUADRATIC OBSERVATIONS AND EXPLICIT LEARNING

For observation variance $r > 0$, the Gaussian loss is $(z-y)^2/(2r)$. Its proximal equation and information weight give

$$z = \frac{rp + sy}{r+s}, \qquad u = \frac{y-p}{r+s}, \qquad w = \frac{1}{r}. \tag{53}$$

These are the usual RLS or scalar Gaussian measurement-update coefficients (Sayed & Kailath, 1994; Bierman, 1977). Independent outputs sharing a feature metric use the same w and a vector residual.

An explicit loss step with the current inverse metric uses

$$u = -c\ell'(p). \tag{54}$$

The chosen metric rule supplies w. Fixed-metric gradient descent uses $w = 0$, and its Euclidean form has $P_0 = I$. A step using the updated inverse satisfies

$$P^+k = \frac{v}{1+ws}, \qquad u = -\frac{c\ell'(p)}{1+ws}.$$

These formulas make the timing of the preconditioner explicit. They describe per-example updates of a linear learner, including a learner whose keys are produced by a neural feature map. Training procedures with a frozen within-minibatch learner have their own update schedule (Sun et al., 2025).

D.2 Unprojected scalar-loss Online Newton Step

For $f_t(\mu) = \ell_t(k_t^\top \mu)$, the parameter gradient is $g_t k_t$, where $g_t = \ell_t'(p_t)$. Accumulating gradient outer products and taking a step with the updated metric gives

$$w_t = g_t^2, \qquad u_t = -\frac{g_t}{\gamma(1+g_t^2 s_t)}, \qquad \gamma > 0. \tag{55}$$

The identity follows by substituting $P_t k_t = v_t/(1+g_t^2 s_t)$ into the mean step. Thus the full response dependence of the information matrix is captured by one scalar per example. The classical Online Newton Step framework includes a feasible-set projection and regret assumptions (Hazan et al., 2007). Equation (55) specifies its unprojected scalar-loss update. A separate constrained implementation must account for its projection. Kernel versions of online second-order optimization also use example-space structure (Calandriello et al., 2017).

D.3 Canonical GLM scoring and recursive Laplace filtering

For a canonical generalized linear model, consider the negative log likelihood

$$\ell(z) = b(z) - yz + \text{constant}.$$

A single Newton update of the local Gaussian-prior objective, initialized at the prior mean, has

$$w = cb''(p), \qquad u = -\frac{c[b'(p)-y]}{1+csb''(p)}. \tag{56}$$

To derive it, the gradient at μ is $c[b'(p)-y]k$ and the local Hessian is $H + cb''(p)kk^\top$. Applying the inverse rank-one identity gives Equation (56). Canonical likelihoods identify observed curvature with the corresponding scalar Fisher curvature. A connection to an extended Kalman update also requires a specified observation model and noise convention (Ollivier, 2018).

A recursive mode-and-Hessian Laplace update solves the current Gaussian-prior problem to its mode:

$$z + cs\ell'(z) = p, \qquad u = -c\ell'(z), \qquad w = c\ell''(z). \tag{57}$$

The parameter mode is $\mu^+ = \mu + uv$. Evaluating the Hessian at this point yields $H^+ = H + c\ell''(z)kk^\top$. The same likelihood multiplier appears in the mean and curvature equations. The mode is the exact minimizer of the stated one-step objective whenever that objective is strongly convex. The posterior representation is the Gaussian Laplace approximation centered at that mode. This local approximation and its recursive use have an established state-space literature (Koyama et al., 2010).

D.4 Binary logistic updates

With $y \in [0,1]$, let

$$\ell(z) = \log(1+e^z) - yz, \quad \ell'(z) = \sigma(z) - y, \quad \ell''(z) = \sigma(z)[1-\sigma(z)].$$

Use a numerically stable softplus and sigmoid. The proximal equation is

$$F(z) = z + \alpha[\sigma(z) - y] - p = 0, \qquad F'(z) = 1 + \alpha\sigma(z)[1-\sigma(z)] \geq 1. \tag{58}$$

Since $\sigma(z) - y \in [-y, 1-y]$, a valid bracket is

$$p - \alpha(1-y) \le z \le p + \alpha y. \tag{59}$$

A safeguarded Newton or bracketed scalar method therefore has a directly available starting interval. At an approximate root $\widehat{z}$, $|\widehat{z} - z| \le |F(\widehat{z})|$. The mean coefficient is $u = -c[\sigma(z) - y]$. A prescribed geometry uses its prescribed w; recursive Laplace uses $w = c\sigma(z)[1 - \sigma(z)]$.

D.5 Poisson observations and a log-domain scalar solve

Let $y \ge 0$ be a count and $\Delta > 0$ its exposure. The Poisson loss, up to an additive constant, is $\ell(z) = \Delta e^z - yz$. The proximal equation is

$$z + \alpha(\Delta e^z - y) = p.$$

For $\alpha > 0$, define $b = p + \alpha y$ and $q = b - z$. Then $q = \alpha\Delta e^{b-q}$, hence $qe^q = \alpha\Delta e^b$. The principal real Lambert function gives

$$z = b - W_0(\alpha\Delta e^b). \tag{60}$$

The Lambert function is the inverse of $q \mapsto qe^q$ on nonnegative arguments (Corless et al., 1996). A stable evaluation solves

$$q + \log q = \log\alpha + \log\Delta + b, \qquad q > 0, \tag{61}$$

then recovers z using either $z = b - q$ or $z = \log q - \log\alpha - \log\Delta$. The second expression is useful when subtracting q from b would cause cancellation. Equation (61) avoids exponentiating b during initialization. The case $\alpha = 0$ gives $z = p$. A generic monotone scalar solve is also valid, since $F'(z) = 1 + \alpha\Delta e^z \ge 1$.

The coefficients for recursive Laplace are $u = -c(\Delta e^z - y)$ and $w = c\Delta e^z$. For extreme values, evaluate $c\Delta e^z = q/s$ when $s > 0$ and $\alpha > 0$, using the scalar solution. Binned event models require the exposure and likelihood to match the observation process.

D.6 Huber and smooth robust losses

Write the entry residual as $r = p - y$. For the Huber loss

$$\ell(z) = \begin{cases} (z-y)^2/2, & |z-y| \le \delta, \\ \delta|z-y| - \delta^2/2, & |z-y| > \delta, \end{cases}$$

the proximal residual is

$$z - y = \begin{cases} r/(1+\alpha), & |r| \le \delta(1+\alpha), \\ r - \alpha\delta\,\mathrm{sign}(r), & |r| > \delta(1+\alpha). \end{cases} \tag{62}$$

The coefficient $u = (z-p)/s$ for $s > 0$ implements this response. The equivalent score form is $u = -c\,\mathrm{clip}(z - y, -\delta, \delta)$. It extends continuously to $s = 0$ at the level of the parameter update.

A smooth alternative is the pseudo-Huber loss

$$\ell(z) = \delta^2\left(\sqrt{1 + [(z-y)/\delta]^2} - 1\right).$$

Its derivatives are

$$\ell'(z) = \frac{z-y}{\sqrt{1 + [(z-y)/\delta]^2}}, \qquad \ell''(z) = \left(1 + [(z-y)/\delta]^2\right)^{-3/2}.$$

The bounded score supplies a bracket $p - \alpha\delta \le z \le p + \alpha\delta$. Its positive curvature permits either a prescribed geometry or a recursive Laplace geometry.

D.7 Student-t updates with a curvature certificate

Let $\nu > 0$ and $\sigma > 0$, and set $a = \nu\sigma^2$. A Student-t negative log likelihood has

$$\ell(z) = \frac{\nu+1}{2} \log\left(1 + \frac{(z-y)^2}{a}\right), \tag{63}$$

$$\ell'(z) = \frac{(\nu+1)(z-y)}{a + (z-y)^2}, \tag{64}$$

$$\ell''(z) = \frac{(\nu+1)[a - (z-y)^2]}{[a + (z-y)^2]^2}. \tag{65}$$

The score tends to zero as the residual magnitude grows. To bound negative curvature, set $x = (z-y)^2/a$. The normalized curvature is $(1-x)/(1+x)^2$, whose derivative is $(x-3)/(1+x)^3$. Its minimum is $-1/8$ at $x = 3$. Thus

$$\ell''(z) \geq -\rho, \qquad \rho = \frac{\nu+1}{8\nu\sigma^2}. \tag{66}$$

The condition $cs\rho < 1$ guarantees a unique proximal solution and, with $w = c\ell''(z)$, a positive update pivot. Also

$$|\ell'(z)| \leq \frac{\nu+1}{2\sqrt{\nu}\sigma},$$

so a scalar solver can use $p \pm \alpha(\nu+1)/(2\sqrt{\nu}\sigma)$ as endpoints. A guarded implementation can select c using the available s before solving.

For a static positive-update metric, $P_{t-1} \preceq P_0$ gives $s_t \leq \lambda_{\max}(P_0)\|k_t\|^2$. Signed updates and process noise require the current local variance or another valid bound. Clipping a curvature weight or step size defines the guarded learner whose trajectory is then reproduced by FlashRLS.

D.8 Absolute, quantile, and insensitive losses

These convex losses have piecewise closed-form proximal maps (Parikh & Boyd, 2014). For absolute error $\ell(z) = |z-y|$,

$$z = y + \operatorname{sign}(p-y)(|p-y| - \alpha)_+. \tag{67}$$

For the quantile loss $\ell(z) = \rho_\tau(y-z)$, with $\rho_\tau(r) = r[\tau - \mathbf{1}\{r < 0\}]$ and $0 < \tau < 1$,

$$z = \begin{cases} p + \alpha\tau, & p < y - \alpha\tau, \\ y, & y - \alpha\tau \leq p \leq y + \alpha(1-\tau), \\ p - \alpha(1-\tau), & p > y + \alpha(1-\tau). \end{cases} \tag{68}$$

For $\ell(z) = (|z-y| - \varepsilon)_+$, write $r = p - y$. Its proximal residual is

$$z - y = \operatorname{sign}(r)\left[|r| - \min\{\alpha, (|r| - \varepsilon)_+\}\right]. \tag{69}$$

Each update uses $u = (z-p)/s$ for $s > 0$ and a selected metric rule. The resulting trajectory solves successive local online objectives. These formulas are directly useful for asymmetric prediction costs and tolerance regions around a regression target.

D.9 Margins, passive-aggressive learning, and AROW

For labels $y \in \{-1, 1\}$ and hinge loss $\ell(z) = (1-yz)_+$, the proximal equation gives

$$u = y \min\left\{c, \frac{(1-yp)_+}{s}\right\}, \qquad s > 0. \tag{70}$$

A fixed metric with $w = 0$ yields the corresponding PA-I update; the cap c controls aggressiveness (Crammer et al., 2006). The squared hinge $\ell(z) = (1-yz)_+^2/2$ gives

$$u = \frac{yc(1-yp)_+}{1+cs}. \tag{71}$$

For AROW with parameter $r > 0$, the coefficients are

$$u = \frac{y(1 - yp)_+}{r + s}, \qquad w = \frac{\mathbf{1}\{yp < 1\}}{r}. \tag{72}$$

This is the squared-hinge correction with $c = 1/r$ on an active example, coupled to an information update on that example (Crammer et al., 2009). Inactive examples have $u = w = 0$ and their later-state correction can be skipped. Additional confidence-weighted algorithms can enter through their own local coefficients and trigger conventions.

D.10 Gaussian assumed-density filtering

Gaussian assumed-density filtering updates an approximate posterior by moment matching (Minka, 2013; Csató & Opper, 2002). Suppose μ and P specify a Gaussian parameter distribution and $z = k^\top x$ has marginal $\mathcal{N}(p, s)$, with $s > 0$. Form the tilted scalar density

$$\widetilde{q}(z) = \frac{p(y \mid z)\mathcal{N}(z; p, s)}{Z}, \qquad Z = \int p(y \mid z)\mathcal{N}(z; p, s)\,\mathrm{d}z. \tag{73}$$

Assume $0 < Z < \infty$, finite mean $\bar{z}$, and finite positive variance $\widehat{s}$. Gaussian conditioning gives

$$\mathbb{E}[x \mid z] = \mu + \frac{v}{s}(z - p), \qquad \mathrm{Cov}(x \mid z) = P - \frac{vv^\top}{s}.$$

Taking the tilted mean and covariance yields

$$\mu^+ = \mu + \frac{\bar{z} - p}{s} v, \tag{74}$$

$$P^+ = P - \frac{s - \widehat{s}}{s^2} vv^\top. \tag{75}$$

The local coefficients are therefore

$$u = \frac{\bar{z} - p}{s}, \qquad \beta = \frac{s - \widehat{s}}{s^2}, \qquad w = \frac{1}{\widehat{s}} - \frac{1}{s}. \tag{76}$$

They satisfy $1 + ws = s/\widehat{s} > 0$. When a likelihood increases the tilted variance, the information increment is negative and the Gaussian covariance remains positive definite. Logistic moments can be computed by one-dimensional quadrature. Quadrature tolerance, Gaussian approximation, and transport roundoff are three separate accuracy quantities. Equation (76) represents the moment-matched update exactly for the moments provided to it.

E Vector losses and block observations

E.1 A linear-time Newton direction for softmax adaptation

Consider a probability target $y \in \mathbb{R}^m$ with nonnegative entries and $\sum_i y_i = 1$. The softmax cross-entropy loss is

$$\ell(z) = \log \sum_{i=1}^{m} e^{z_i} - y^\top z.$$

For a shared feature metric, the local proximal problem becomes

$$z + \alpha(\pi - y) = p, \qquad \pi = \mathrm{softmax}(z), \quad \alpha = cs \geq 0. \tag{77}$$

Its Jacobian is

$$J = I + \alpha\,\mathrm{diag}(\pi) - \alpha\pi\pi^\top. \tag{78}$$

The softmax covariance $\mathrm{diag}(\pi) - \pi\pi^\top$ is positive semidefinite, so $J \succeq I$ and the local problem is strongly convex. The solution also preserves the sum of logits: summing Equation (77) gives $\sum_i z_i = \sum_i p_i$.

Let $D = \mathrm{diag}(1 + \alpha\pi_i)$ and $v = D^{-1}\pi$. For any right-hand side b, a rank-one inverse gives

$$J^{-1}b = D^{-1}b + \frac{\alpha v(v^\top b)}{1 - \alpha\pi^\top v}. \tag{79}$$

The denominator has the stable positive expression

$$1 - \alpha \sum_i \frac{\pi_i^2}{1 + \alpha\pi_i} = \sum_i \frac{\pi_i}{1 + \alpha\pi_i} > 0. \tag{80}$$

The equality uses $\sum_i \pi_i = 1$. A Newton direction therefore costs $O(m)$, including the diagonal scaling and dot products. A stable log-sum-exp evaluation and a descent line search give a practical local solver. The overall cost also includes the number of Newton iterations.

The correction vector is $u = -c(\pi - y)$. A prescribed scalar weight can maintain a shared feature metric for this mean update. A full Bayesian multiclass covariance lives in the joint parameter space and includes class-to-class correlations. Its observation curvature has matrix rank up to $m - 1$ and is covered by the block formulation below, with its associated storage and arithmetic costs.

E.2 General block update identities

Let a parameter vector $x \in \mathbb{R}^D$ have inverse information matrix $P \succ 0$. At step t, a known matrix $L_t \in \mathbb{R}^{r_t \times D}$ maps parameters to an observation-space prediction. Define

$$V_t = P_{t-1} L_t^\top, \quad S_t = L_t P_{t-1} L_t^\top, \quad p_t = L_t x_{t-1}. \tag{81}$$

A local rule returns $u_t \in \mathbb{R}^{r_t}$ and a symmetric information increment $C_t \in \mathbb{R}^{r_t \times r_t}$, giving

$$x_t = x_{t-1} + V_t u_t, \qquad H_t = H_{t-1} + L_t^\top C_t L_t.$$

Assume $H_t \succ 0$. Equivalently, $I + S_t^{1/2} C_t S_t^{1/2} \succ 0$; zero eigenvalues of S_t contribute identity eigenvalues. Define the zero-safe symmetric matrix

$$B_t = (I + C_t S_t)^{-1} C_t = C_t (I + S_t C_t)^{-1}. \tag{82}$$

The equality follows from $(I + C_t S_t) C_t = C_t (I + S_t C_t)$. Inverting and transposing establishes symmetry. The generalized rank update gives

$$P_t = P_{t-1} - V_t B_t V_t^\top. \tag{83}$$

For example, multiply the right side by $H_{t-1} + L_t^\top C_t L_t$ and use $B_t + C_t S_t B_t = C_t$ to obtain the identity matrix.

Define block Gram entries $\Gamma_{ij} = L_i P_0 L_j^\top$ and ordered interactions $M_{ij} = L_i V_j$, for $i \geq j$. Telescoping Equation (83) gives

$$M_{ij} = \Gamma_{ij} - \sum_{\ell < j} M_{i\ell} B_\ell M_{j\ell}^\top, \qquad p_i = L_i x_0 + \sum_{j<i} M_{ij} u_j. \tag{84}$$

The diagonal is $S_i = M_{ii}$. A block unit lower-triangular matrix with block $T_{ij} = M_{ij} B_j$ for $i > j$ reconstructs all directions: stack the L_i as L, stack the $V_i^\top$ as A, and solve $TA = LP_0$. Formulas for states and queries after each update follow by accumulating $V_j u_j$ and $V_j B_j V_j^\top$.

For a convex response loss, the local mean update satisfies

$$z_t + c_t S_t \nabla \ell_t(z_t) = p_t, \qquad u_t = -c_t \nabla \ell_t(z_t).$$

This follows from the parameter-space quadratic proximal objective. If S_t is singular, the attained prediction lies in $p_t + \mathrm{range}(S_t)$, and the problem can be solved in that range. The parameter objective remains strongly convex for a proper closed convex response loss whose domain is reachable. When every $r_t = 1$, Equation (84) reduces to the scalar-weight theorem. With total observation dimension $R = \sum_t r_t$, generic dense core work is $O(R^3)$; the ambient parameter dimension is D. Structured observation maps can reduce the preparation cost.

F Nonlinear solvers and residual certificates

F.1 Ordered local solves

For fixed geometry, define $D_c = \mathrm{diag}(c_i)$ and let M be the lower-triangular interaction matrix. The implicit response equation is

$$F_i(Z) = z_i + c_i s_i \psi_i(z_i) + \sum_{j<i} M_{ij} c_j \psi_j(z_j) - a_i = 0. \tag{85}$$

Once rows $1, \dots, i-1$ are known, the right-hand side for row i is known. A well-posed local proximal solve therefore determines row i. Induction establishes existence and uniqueness of the entire sequential trajectory whenever every reachable local problem has a unique solution. The nonlinear work is confined to response variables. Feature dimension enters preparation and reconstruction.

For scalar convex losses, the derivative of the local map $z \mapsto z + \alpha\ell'(z)$ is at least one wherever it exists. The logistic, Poisson, and pseudo-Huber examples therefore admit safeguarded monotone root solvers. Piecewise formulas implement the convex nonsmooth examples. A scalar residual e_i certifies a local solution error at most $|e_i|/d_i$ whenever the derivative is uniformly bounded below by $d_i > 0$.

F.2 Simultaneous diagonal proximal sweeps

A parallel iteration holds the earlier interactions at their previous values and solves all diagonal problems simultaneously:

$$z_i^{(r+1)} = \operatorname{prox}_{c_i s_i \ell_i}\left(a_i - \sum_{j<i} M_{ij} c_j \psi_j(z_j^{(r)})\right). \tag{86}$$

Here $\operatorname{prox}_{\alpha\ell}(b)$ minimizes $\|z-b\|^2/2 + \alpha\ell(z)$. The information weights and hence M are fixed during these sweeps.

Proposition F.1 (Finite ordered completion)**.** *With exact local solves and unique local solutions, Equation* (86) *produces the exact first r rows after r sweeps. Every row is exact after at most C sweeps in exact arithmetic.*

Proof. The first row has no preceding interactions and is solved exactly in the first sweep. If rows $1, \dots, r$ are exact, the next sweep presents the exact right-hand side to row $r+1$ and preserves every earlier exact row. Induction proves the claim. □

The result describes the ideal iteration. A practical implementation selects tolerances and accounts for the work of every sweep. Contracting inter-example interactions characterize a useful fast regime.

Assume scalar scores satisfy $|\psi_j(x) - \psi_j(y)| \le L_j|x-y|$, and each local loss has curvature bounded below by μ_i, with $d_i = 1 + c_i s_i \mu_i > 0$. Define a nonnegative strictly lower-triangular matrix

$$N_{ij} = \frac{|M_{ij}| c_j L_j}{d_i} \quad (j<i), \qquad N_{ij} = 0 \quad (j \ge i). \tag{87}$$

The inverse local map is $1/d_i$-Lipschitz. Applying this fact to Equation (86) gives

$$|Z^{(r+1)} - Z^\star| \le N|Z^{(r)} - Z^\star|.$$

Thus $\|N\|_\infty < 1$ is a sufficient contraction condition in the maximum norm. The vector-output version uses Euclidean norms per row and Lipschitz bounds for the corresponding gradient maps.

F.3 A posteriori error bounds for a fixed geometry

Let $\widehat{Z}$ be an approximate solution and $e = F(\widehat{Z})$. Subtracting the exact row equations and using the same local monotonicity and Lipschitz bounds gives

$$|\widehat{z}_i - z_i^\star| \le \frac{|e_i|}{d_i} + \sum_{j<i} N_{ij}|\widehat{z}_j - z_j^\star|.$$

Writing $D = \operatorname{diag}(d_i)$ yields the certificate

$$|\widehat{Z} - Z^\star| \le (I-N)^{-1} D^{-1}|e|, \qquad (I-N)^{-1} = I + N + \cdots + N^{C-1}. \tag{88}$$

The finite inverse follows from strict lower triangularity. It can be applied by a forward substitution. The certificate needs no contraction assumption; its size records amplification through later updates.

Let B_q be a fixed query interaction matrix with entries selected by the required update-selection mask, so $O = QW_0^\top - B_q D_c \Psi(Z)$. Then

$$|\widehat{O} - O^\star| \le |B_q| D_c \operatorname{diag}(L_i)|\widehat{Z} - Z^\star|. \tag{89}$$

For vector responses, interpret the absolute errors as row norms. These formulas support stopping at a prediction-error tolerance. The certificate in Equation (88) conditions on a fixed M. A response-adaptive metric requires bounds through its weight rule as well.

F.4 Newton and solves in observation blocks

For scalar smooth losses with fixed M and c, the Jacobian is

$$J_Z = I + M\,\mathrm{diag}(c_i \ell_i''(z_i)). \tag{90}$$

A Newton step solves a lower-triangular linear system at $O(C^2)$ cost. Separable m-output losses give $O(C^2 m)$ work. Coupled vector losses have diagonal blocks $I + c_i s_i \nabla^2 \ell_i(z_i)$ and require their own local block solves. Equation (79) provides one inexpensive coupled example.

A block implementation divides the sequence into groups of b examples. It fixes completed earlier groups, solves the nonlinear equations inside the next group, and advances. Within a group, local quadratic convergence follows under a nonsingular Jacobian, a locally Lipschitz Jacobian, and a sufficiently close initialization. These conditions follow the usual Newton analysis. The number of sequential groups is $\lceil C/b \rceil$; total runtime also includes local iterations and triangular work within each group. An ordered scalar-solve implementation remains a useful reference for this schedule.

F.5 Adaptive geometry and augmented certificates

For a response-adaptive rule, changing a response also changes later information weights and interactions. One can treat the ordered outputs of a local rule as an augmented variable containing the mean coefficient, curvature coefficient, and any auxiliary state. Suppose a chosen norm and a local analysis give

$$e_t \le \eta_t + \sum_{j<t} L_{tj} e_j,$$

where η_t bounds the local numerical defect and $L_{tj} \ge 0$ bounds the dependence of step t on earlier outputs in a specified neighborhood. Forward substitution gives

$$e \le (I - L)^{-1} \eta.$$

This is an exact consequence of the assumed Lipschitz inequalities. Obtaining useful L_{tj} requires bounds for the particular oracle, pivot margins, and geometry updates. The fixed-geometry certificate supplies a concrete closed form for the important prescribed-metric case.

G Differentiation through local rules and adaptive geometry

This appendix proves Theorem 3.4 by giving the reverse of every stage in its exact forward composition. We derive the local implicit derivatives and the reverse pass through the adaptive geometry. Smooth identities are evaluated on trajectories with nonzero pivot margins and well-posed local solves. The formulas can be implemented through automatic differentiation or explicit vector–Jacobian products. Implicit differentiation treats the converged local equation as the mathematical target (Blondel et al., 2022).

G.1 Derivative of a nonlinear proximal update

For a vector response, write

$$F(z, p, s, c, \theta) = z + cs\psi(z; \theta) - p = 0, \qquad J = I + cs\nabla_z \psi(z; \theta).$$

If J is nonsingular, differentiation gives

$$\mathrm{d}z = J^{-1}\left[\mathrm{d}p - (s\,\mathrm{d}c + c\,\mathrm{d}s)\psi - cs\,\partial_\theta \psi\,\mathrm{d}\theta\right]. \tag{91}$$

Given an incoming cotangent $\bar{z}$, solve $J^\top \lambda = \bar{z}$. The local adjoints are

$$\bar{p} \mathrel{+}= \lambda, \quad \bar{s} \mathrel{+}= -c\lambda^\top \psi, \quad \bar{c} \mathrel{+}= -s\lambda^\top \psi, \quad \bar{\theta} \mathrel{+}= -cs(\partial_\theta \psi)^\top \lambda. \tag{92}$$

The output coefficient $u = -c\psi(z;\theta)$ contributes its own chain rule. For example, $\bar{u}$ adds $-c(\nabla_z \psi)^\top \bar{u}$ to $\bar{z}$ and $-\bar{u}^\top \psi$ to $\bar{c}$. A scalar Laplace weight $w = c\ell''(z)$ adds $c\ell'''(z)\bar{w}$ to $\bar{z}$. The required derivative order follows from the selected oracle.

The equation describes differentiation of the exact local solution. Differentiating a finite sequence of solver iterations gives the derivative of that implemented iteration. Agreement with the implicit derivative depends on convergence and numerical conditioning.

G.2 Derivatives through adaptive curvature

For scalar implicit updates with $c_i = 1$, write the stacked adaptive equation as

$$F(z) = z + M(z)\psi(z) - a.$$

The full differential is

$$DF(z)[h] = h + M(z)\,\mathrm{diag}(\psi'(z))h + \big(DM(z)[h]\big)\psi(z). \tag{93}$$

The last term propagates response changes through earlier curvature weights. Adaptive step coefficients add their own terms. The dependency remains lower triangular because each geometric change affects its current pivot and subsequent steps in the specified order.

For two recursive Laplace updates, let $s_1 = \Gamma_{11}$ and $b = \Gamma_{21}$. Then

$$\beta_1 = \frac{\ell_1''(z_1)}{1 + s_1\ell_1''(z_1)}, \qquad s_2 = \Gamma_{22} - \beta_1 b^2.$$

The second equation is $F_2 = z_2 + b\ell_1'(z_1) + s_2\ell_2'(z_2) - a_2$. Hence

$$\frac{\partial F_2}{\partial z_1} = b\ell_1''(z_1) - \frac{b^2\ell_1'''(z_1)\ell_2'(z_2)}{[1 + s_1\ell_1''(z_1)]^2}. \tag{94}$$

A stopped-gradient curvature implementation deliberately removes the geometric dependence. It defines a separate differentiation rule that can be evaluated independently during model training.

G.3 Reverse mode for the ordered interaction matrix

A convenient implementation keeps a trailing interaction matrix Γ^{tr} and trailing predictions a^{tr}. At the current step, select the diagonal s, prediction p, and future interaction column c. After evaluating the oracle and $\beta = w/(1 + ws)$, update

$$\Gamma^{\mathrm{tr},+} = \Gamma^{\mathrm{tr}} - \beta cc^\top, \qquad a^{\mathrm{tr},+} = a^{\mathrm{tr}} + cu^\top. \tag{95}$$

Only future rows and their trailing principal block participate. Let $\bar{\Gamma}^+$ and $\bar{a}^+$ be the incoming cotangents. The contribution of this stage is

$$\bar{\beta} \mathrel{+}= -c^\top \bar{\Gamma}^+ c, \tag{96}$$

$$\bar{c} \mathrel{+}= -\beta(\bar{\Gamma}^+ + \bar{\Gamma}^{+\top})c + \bar{a}^+ u, \tag{97}$$

$$\bar{u} \mathrel{+}= \bar{a}^{+\top} c. \tag{98}$$

Copies of the trailing arrays also receive their incoming cotangents. The scalar coefficient gives

$$\frac{\partial \beta}{\partial w} = \frac{1}{(1 + ws)^2}, \qquad \frac{\partial \beta}{\partial s} = -\frac{w^2}{(1 + ws)^2}. \tag{99}$$

The oracle's vector–Jacobian product propagates $\bar{u}, \bar{w}$ to p, s, y and its parameters. Reverse accumulation through the selected entries of the trailing arrays completes this stage. Stored interaction columns additionally receive cotangents from state reconstruction and query evaluation. Executing these steps in reverse time differentiates the complete adaptive geometry.

G.4 Direction reconstruction and boundary-state adjoints

Let $B_0 = KP_0$, $\mathcal{B} = \mathrm{diag}(\beta_t)$, and $A = T^{-1}B_0$. The forward triangular solve satisfies $TA = B_0$. Given $\bar{A}$, solve $T^\top Y = \bar{A}$ and accumulate

$$\bar{B}_0 \mathrel{+}= Y, \qquad \bar{T} \mathrel{+}= -YA^\top. \tag{100}$$

The diagonal of T is fixed at one. For $T = I + \mathrm{tril}(M, -1)\mathcal{B}$,

$$\bar{M} \mathrel{+}= \mathrm{tril}(\bar{T}, -1)\mathcal{B}, \qquad \bar{\beta}_j \mathrel{+}= \sum_{i>j} \bar{T}_{ij} M_{ij}.$$

The boundary states are $P_C = P_0 - A^\top \mathcal{B} A$ and $W_C = W_0 + U^\top A$. Their adjoints give

$$\bar{A} \mathrel{+}= -\mathcal{B}A(\bar{P}_C + \bar{P}_C^\top) + U\bar{W}_C, \tag{101}$$

$$\bar{\beta}_t \mathrel{+}= -A_t \bar{P}_C A_t^\top, \qquad \bar{U} \mathrel{+}= A\bar{W}_C^\top, \tag{102}$$

$$\bar{P}_0 \mathrel{+}= \bar{P}_C, \qquad \bar{W}_0 \mathrel{+}= \bar{W}_C. \tag{103}$$

Here A_t is row t. The covariance cotangent can be symmetrized for a symmetric input.

For a read-before-update output, set $N = \mathrm{tril}(QA^\top, -1)$ and $O = QW_0^\top + NU$. Given $\bar{O}$, define $E = \mathrm{tril}(\bar{O}U^\top, -1)$. Then

$$\bar{U} \mathrel{+}= N^\top \bar{O}, \qquad \bar{A} \mathrel{+}= E^\top Q, \tag{104}$$

$$\bar{Q} \mathrel{+}= \bar{O}W_0 + EA, \qquad \bar{W}_0 \mathrel{+}= \bar{O}^\top Q. \tag{105}$$

A different fixed read mask is applied to the corresponding cotangents in the same manner.

Finally, the preparation maps $B_0 = KP_0$, $\Gamma = KP_0K^\top$, and $a = KW_0^\top$ give, for symmetric P_0,

$$\bar{K} \mathrel{+}= \bar{B}_0 P_0 + (\bar{\Gamma} + \bar{\Gamma}^\top)KP_0 + \bar{a}W_0, \tag{106}$$

$$\bar{P}_0 \mathrel{+}= \mathrm{sym}(K^\top \bar{B}_0 + K^\top \bar{\Gamma} K), \tag{107}$$

$$\bar{W}_0 \mathrel{+}= \bar{a}^\top K. \tag{108}$$

These identities account for the feature map, the initial metric, and arbitrary query vectors. They supply a full adaptive backward with the same dense algebraic orders as the forward pass, together with each local rule's derivative cost.

G.5 Nonsmooth rules and branch conventions

Piecewise proximal maps are differentiable within each smooth region. For example, the absolute-loss prox has derivative one outside its threshold interval and zero in its interior; boundary points require a stated derivative convention. Quantile, insensitive, hinge, and Huber rules have analogous piecewise derivatives. The forward solution remains uniquely defined for proper closed convex losses. Generalized derivatives at transitions and semismooth Newton methods require assumptions appropriate to the chosen map. A reproducible implementation records its branch behavior and tests points on each side of a transition.

G.6 Exact derivatives on smooth branches

The forward identities in Theorem 3.2 use algebra and well-defined local outputs. They hold for nonsmooth local rules under Assumptions 2.1–2.2. Differentiability enters the backward theorem. A useful extension treats a local rule as a continuous collection of smooth branches, such as the two branches of a clipped or reweighted residual. An *active branch* is the formula selected by the local comparison at the current inputs.

Proposition G.1 (Branchwise and directional equivalence)**.** *Fix a finite trajectory and regard its inputs and parameters as a finite-dimensional vector x. Assume that the local output maps, including solved responses and retained auxiliary state, are locally Lipschitz and piecewise C^1 near the visited inputs. Here piecewise C^1 means a continuous selection of finitely many C^1 maps with Hadamard directional derivatives. Assume positive pivots and identical local rules in both executions.*

1. *On any open region where all active branches remain fixed and the selected implicit roots have nonsingular response Jacobians, the sequential and block maps are C^1 and have the same ordinary Jacobian. Their reverse passes have the dense arithmetic order stated in Theorem 3.4.*
2. *At a branch boundary, their one-sided directional derivatives agree whenever formed using the branches selected by the same input perturbation. Transposing a fixed branch linearization gives the same branch-specific vector–Jacobian product in both executions.*
3. *The complete maps are locally Lipschitz and equal, so their ordinary Jacobians exist and agree almost everywhere. A branch-specific Jacobian limit belongs to their common limiting Jacobian set when that complete branch pattern is reachable through nearby open smooth regions.*

Proof. For a fixed branch pattern, replace each local rule by its C^1 branch extension. The local implicit function theorem supplies the selected response maps. Inversion, matrix products, and triangular solves are smooth near the trajectory; positivity keeps every denominator nonzero. The proof of Theorem 3.2 applies to these extensions and gives identical maps on a neighborhood where they are defined. Differentiating the identity proves the first claim, including equality of transposed linearizations. The reverse contractions retain their shapes and arithmetic orders.

Locally Lipschitz, Hadamard directionally differentiable maps obey the directional chain rule under finite composition. For a perturbation direction, propagate the corresponding directional branch at each local call and differentiate the remaining smooth algebra. The two forward maps agree for the perturbed inputs, so the limits of their difference quotients agree. This proves the second claim. Finite composition preserves local Lipschitz continuity, and Rademacher's theorem gives almost-everywhere differentiability. Finally, any jointly reachable smooth-branch sequence gives the same Jacobians in both implementations; taking its limit proves the last claim. □

The distinction at a kink is between an ordinary derivative and a selected linearization. Write $\mathcal{F}$ for the complete map and define its limiting Jacobians and Clarke generalized Jacobian by

$$\partial_B \mathcal{F}(x) = \left\{ \lim_j D\mathcal{F}(x_j) : x_j \to x,\ \mathcal{F} \text{ differentiable at } x_j \right\}, \qquad \partial_C \mathcal{F}(x) = \operatorname{conv} \partial_B \mathcal{F}(x).$$

The locally Lipschitz assumption makes these sets well defined. A jointly reachable branch limit is in $\partial_B \mathcal{F}(x)$; convex combinations of such limits are in $\partial_C \mathcal{F}(x)$. Independently choosing a generalized slope at every primitive can produce a matrix outside the generalized Jacobian of the complete map. For example, inconsistent choices for the two occurrences in $|x| - |x|$ can give a nonzero slope for the identically zero function. Nonsmooth automatic and implicit differentiation therefore need a consistent chain-rule framework (Bolte et al., 2021). Proposition G.1 gives the branchwise and directional statements needed here without asserting an ordinary gradient at every kink.

G.7 Huber responses, reweighting, and ties

The loss and the information weight. For threshold $\delta > 0$, the Huber loss is continuously differentiable, with score

$$\psi_\delta(r) = \operatorname{clip}(r, -\delta, \delta).$$

Its convex subgradient is the singleton $\{\psi_\delta(r)\}$, including at $r = \pm\delta$. The ambiguity concerns the derivative of this score, and the derivative of the information weight used by the local rule. Protocol B2 uses the continuous reweighting map in Appendix L.6:

$$w(r,\delta) = \begin{cases} 1, & |r| \le \delta, \\ \delta/|r|, & |r| > \delta. \end{cases} \qquad (w_r, w_\delta) = \begin{cases} (0,0), & |r| < \delta, \\ (-\delta \operatorname{sign}(r)/|r|^2,\ 1/|r|), & |r| > \delta. \end{cases}$$

At $r = 0$ the weight is locally constant and its residual derivative is zero. The outer branch has a nonzero weight derivative. This information weight differs from the piecewise Huber second derivative, which is zero for $|r| > \delta$. The experiment's adaptive derivative follows the specified reweighting map and target corrections.

At $r = \sigma\delta$, $\sigma \in \{-1, 1\}$, the two limiting derivative pairs are $(0,0)$ and $(-\sigma/\delta, 1/\delta)$. Their convex hull is

$$\partial_C w(\sigma\delta, \delta) = \{\theta(-\sigma/\delta, 1/\delta) : 0 \le \theta \le 1\}.$$

The inner one-sided convention takes $\theta = 0$; the outer one takes $\theta = 1$. For Protocol B2's fixed $\delta = 0.5$, the residual slopes are 0 or -2 at $r = 0.5$, and 0 or 2 at $r = -0.5$. These are mathematical endpoint choices. The forward inequality $|r| \le \delta$ alone does not specify the derivative of a software primitive at equality.

For perturbations of both arguments, the actual directional derivative at the tie is

$$Dw(\sigma\delta, \delta)[\dot{r}, \dot{\delta}] = \min\left\{0, \frac{\dot{\delta} - \sigma\dot{r}}{\delta}\right\}.$$

It follows by expanding the two arguments of $\min\{1, \delta/|r|\}$. The dependence on the direction explains why one fixed reverse-mode slope cannot represent all directional derivatives there. When the score is evaluated as $\psi_\delta(r) = w(r, \delta)r$, consistent derivative choices satisfy

$$(\partial_r\psi, \partial_\delta\psi) = (q, \sigma(1-q)), \qquad q = 1 - \theta,$$

at the same tie. This preserves the product rule for the selected branch.

The proximal response. Let $\rho = p - y$, $\alpha = cs \ge 0$, and $e = z - y$. The scalar Huber proximal equation is $e + \alpha\psi_\delta(e) = \rho$, with the unique solution

$$e = \begin{cases} \rho/(1+\alpha), & |\rho| \le (1+\alpha)\delta, \\ \rho - \alpha\delta\,\mathrm{sign}(\rho), & |\rho| > (1+\alpha)\delta. \end{cases}$$

This is Equation (62) written in residual coordinates. On either open branch, ordinary implicit differentiation applies. At $e = \sigma\delta$, choose the inner branch $q = 1$ or outer branch $q = 0$. Its differential is

$$\mathrm{d}e = \frac{\mathrm{d}\rho - \sigma\delta\,\mathrm{d}\alpha - \alpha\sigma(1-q)\,\mathrm{d}\delta}{1 + \alpha q}, \qquad \mathrm{d}z = \mathrm{d}y + \mathrm{d}e.$$

The denominator is at least one for every $q \in [0, 1]$, so every selected scalar response Jacobian is nonsingular. A directional perturbation selects the inner branch when $\sigma\dot{\rho} - \delta\dot{\alpha} - (1+\alpha)\dot{\delta} < 0$ and the outer branch when this quantity is positive. At zero both formulas give the same directional response. The score form $u = -c\psi_\delta(e)$ also avoids division by s at a zero key. These formulas clarify the local Huber response; each experiment retains its declared local-rule composition.

Connection to the reported checks. B0.8 compares the implementation with a selected one-sided derivative. Proposition G.1 covers either endpoint when the sequential and block computations use the same choice, including the same choice inside any target correction. A reproducible tie specification records the residual supplied to each comparison, its equality branch, and its vector–Jacobian product. The reported B0.8 pass/fail result does not itself identify which endpoint was selected. This specification is distinct from the forward threshold and is required to reproduce an exact equality test.

Ordinary centered finite differences apply away from a branch boundary. At a kink, the relevant check uses $(\mathcal{F}(x + \varepsilon v) - \mathcal{F}(x))/\varepsilon$ with the direction v fixed and $\varepsilon \downarrow 0$. A dot-product test uses one fixed branch linearization J, comparing $\langle Jv, u\rangle$ and $\langle v, J^\top u\rangle$. It tests the transposed implementation of that linearization. These interpretations leave the reported tolerances and outcomes unchanged.

G.8 Local pivot margins and implicit-solver assumptions

Proposition G.2 (Finite-trajectory pivot margin)**.** *Suppose $H_0 \succ 0$, the local outputs are continuous near a finite trajectory, and each of its pivots h_t is positive. There is a neighborhood of the input and parameter vector on which every pivot is at least $\varepsilon/2$, where $\varepsilon = \min_t h_t > 0$. Thus a locally uniform pivot margin follows from pointwise positivity on that finite trajectory. For nonnegative information weights, $h_t \ge 1$ throughout the positive-definite trajectory.*

Proof. There are finitely many pivots, so their positive minimum is positive. Each state and pivot depends continuously on the inputs by induction through the local rules and the inverse update. Intersect the finitely many neighborhoods on which $|h_t(x') - h_t(x)| < \varepsilon/2$. For a nonnegative weight, $s_t = k_t^\top P_{t-1} k_t \ge 0$ and $h_t = 1 + w_t s_t \ge 1$; the update preserves positive definiteness. □

The proposition removes the need to posit a common margin across all possible training trajectories for a local derivative result. Uniform long-horizon or numerical-error bounds still need the corresponding uniform control. Logistic Laplace and the Huber reweighting rule have nonnegative weights, so their kinks introduce no small-pivot problem in exact arithmetic. The matrix reconstruction and accumulated floating-point error require their existing numerical checks.

Algebraic and positive-definite pivot conditions. For a positive-definite incoming matrix, let $a = H_{t-1}^{-1/2} k_t$. Then

$$H_t = H_{t-1}^{1/2}(I + w_t a a^\top) H_{t-1}^{1/2}.$$

For $a \neq 0$, the middle matrix has eigenvalue $h_t = 1 + w_t \|a\|^2$ along a and eigenvalue one on its orthogonal complement. At $a = 0$ it is I. Hence $h_t > 0$ is exactly the condition for retaining a positive-definite information state. A zero pivot makes the updated matrix singular; a negative pivot loses the covariance and positive-metric interpretation.

There is a separate algebraic relaxation. If H_0 is symmetric and invertible and every $h_t \neq 0$, the inverse-update identity and Theorem 3.2's reconstruction formulas still hold for any well-defined local rule. The determinant identity $\det H_t = \det H_{t-1} h_t$ preserves invertibility. Sherman–Morrison and the same telescoping induction then prove the trajectory identities; the unit-triangular reconstruction remains invertible. Smooth local rules also give the corresponding local derivative identity. This extension permits indefinite states and possibly negative s_t. A positive quadratic proximal objective or a variance interpretation requires the stronger positive-definite assumptions used in the experiments.

Local rather than global smoothness and uniqueness. For an implicit equation $F_t(z; x) = 0$, differentiating a chosen solution only needs a C^1 branch near that root and an invertible $\partial_z F_t$ there. The implicit function theorem then defines a unique local continuation of the selected root. Global uniqueness of every root is unnecessary for this local statement; both executions must use the same root-selection rule. For nonsmooth convex losses, the proximal minimizer is unique without a globally smooth loss (Parikh & Boyd, 2014). The explicit Huber branches above supply its local derivatives. The sufficient weak-convexity condition $cs\rho < 1$ in Appendix C is a convenient global certificate; the branchwise result only needs the stated local root condition.

Smoothing is a separate learner. The pseudo-Huber loss already given in Appendix D,

$$\ell_{\mathrm{PH}}(e) = \delta^2\big(\sqrt{1 + (e/\delta)^2} - 1\big), \qquad \ell''_{\mathrm{PH}}(e) = (1 + (e/\delta)^2)^{-3/2},$$

provides a smooth alternative with positive Laplace curvature. For fixed $\delta > 0$, $\ell'''_{\mathrm{PH}}(e) = -3e\delta^{-2}(1 + (e/\delta)^2)^{-5/2}$, so its response and curvature derivatives fit Theorem 3.4 directly. Replacing Huber by this loss changes the forward rule and requires its own experiment. The reported Huber results use their original piecewise rule and are not reinterpreted as smoothed runs.

H FORGETTING, PROCESS NOISE, AND COORDINATE DYNAMICS

The connection to generalized-forgetting RLS and QR square-root prediction is derived in Appendix N.4. This appendix specifies the structured transitions supported by the transport.

The first two subsections establish Proposition 3.6.

H.1 KNOWN DIAGONAL PROCESS NOISE WITH A DENSE COVARIANCE

A changing latent state can receive process covariance before each observation. Consider

$$P_t^- = P_{t-1} + N_t, \qquad N_t = \mathrm{diag}(n_t) \succeq 0, \tag{109}$$

with the diagonal vectors n_t known for the chunk. Define $v_t = P_t^- k_t$, $s_t = k_t^\top P_t^- k_t$, and $p_t = W_{t-1} k_t$. Apply a local rule and set

$$P_t = P_t^- - \beta_t v_t v_t^\top, \qquad W_t = W_{t-1} + u_t v_t^\top, \qquad \beta_t = \frac{w_t}{1 + w_t s_t}. \tag{110}$$

This retains a dense covariance and uses a diagonal model for newly added process uncertainty. The information increment is relative to $(P_t^-)^{-1}$. The statistical prediction and update have their usual filtering interpretation (Bierman, 1977; Särkkä & García-Fernández, 2020).

Let $\rho_t = \sum_{j \le t} n_j$, and let $\mathcal{N}$ collect the $\rho_t^\top$ as rows. Define

$$B = KP_0 + K \odot \mathcal{N}, \tag{111}$$

$$\Gamma^{\mathrm{dyn}}_{ij} = k_i^\top [P_0 + \mathrm{diag}(\rho_{\min(i,j)})] k_j. \tag{112}$$

The symmetric matrix Γ^{dyn} is assembled from the upper triangle of $BK^\top$:

$$\Gamma^{\mathrm{dyn}} = \mathrm{triu}(BK^\top) + \mathrm{triu}(BK^\top, 1)^\top.$$

It is the prior covariance of the scalar latent observations under the random-walk process.

Proposition H.1 (Sequential process-noise representation)**.** *Use Γ^{dyn} in Appendix B's ordered recurrence and set $T = I + \mathrm{tril}(M, -1)\,\mathrm{diag}(\beta)$. Then $A = T^{-1}B$ has rows $v_t^\top$, and the state after each update satisfies*

$$P_c = P_0 + \mathrm{diag}(\rho_c) - A_{1:c}^\top \mathrm{diag}(\beta_{1:c}) A_{1:c}, \qquad W_c = W_0 + U_{1:c}^\top A_{1:c}. \tag{113}$$

The prediction before the update remains $p_t = a_t + \sum_{j<t} M_{tj} u_j$ with $a = KW_0^\top$.

Proof. Telescoping Equation (110) gives

$$P_j^- = P_0 + \mathrm{diag}(\rho_j) - \sum_{\ell < j} \beta_\ell v_\ell v_\ell^\top.$$

Multiplying on the left by $k_i^\top$ and on the right by k_j, for $i \ge j$, gives $M_{ij} = \Gamma^{\mathrm{dyn}}_{ij} - \sum_{\ell<j} \beta_\ell M_{i\ell} M_{j\ell}$. Multiplication by k_j alone gives the direction recurrence and hence $TA = B$. Telescoping the mean and covariance proves the stated state identities. □

The cumulative sums and elementwise multiplication add $O(Cd)$ preparation work. The leading dense orders remain those in Appendix B. With unit Gaussian observation noise and prescribed unit information weights, factor $I + \Gamma^{\mathrm{dyn}}$ to recover the usual innovation representation.

Queries use the same mean formula as the static case. For a variance query just after the process prediction at time t, the baseline is $q_t^\top [P_0 + \mathrm{diag}(\rho_t)] q_t$ and the correction sums over $j < t$. A post-observation query also includes the current correction.

H.2 Known invertible coordinatewise transitions

For a scalar-output parameter vector, consider the prediction model

$$x_t^- = F_t x_{t-1} + \epsilon_t, \qquad \mathrm{Cov}(\epsilon_t) = N_t,$$

where F_t is a known invertible diagonal matrix and N_t is known diagonal positive semidefinite. Put $\Phi_0 = I$ and $\Phi_t = F_t \Phi_{t-1}$. The transformed coordinates

$$\bar{x}_t = \Phi_t^{-1} x_t, \qquad \bar{k}_t = \Phi_t^\top k_t, \qquad \bar{q}_t = \Phi_t^\top q_t, \tag{114}$$

obey an additive-noise prediction with

$$\bar{N}_t = \Phi_t^{-1} N_t \Phi_t^{-\top}. \tag{115}$$

Indeed, $\Phi_t^{-1} F_t x_{t-1} = \Phi_{t-1}^{-1} x_{t-1}$, and $k_t^\top x_t = \bar{k}_t^\top \bar{x}_t$. The transformed covariance and mean therefore follow Equation (113). Recover physical coordinates using

$$P_t = \Phi_t \bar{P}_t \Phi_t^\top, \qquad x_t = \Phi_t \bar{x}_t.$$

For a row-stacked predictor, $W_t = \bar{W}_t \Phi_t^\top$. All these coordinate transformations are elementwise. The transformed local inputs p_t, s_t equal their physical-coordinate values.

Small cumulative gates can cause large inverse coordinate scales. Chunk-boundary rebasing limits the span of these products; a numerical implementation also monitors overflow and underflow. Singular transitions require a reset or rank-aware representation. This construction requires invertibility.

H.3 Common fading of quadratic sufficient statistics

Let a quadratic learner maintain

$$H_t = \gamma_t H_{t-1} + \omega_t k_t k_t^\top, \qquad B_t = \gamma_t B_{t-1} + \omega_t y_t k_t^\top, \qquad W_t = B_t H_t^{-1},$$

with known $\gamma_t > 0$ and $\omega_t \geq 0$. Define $\zeta_t = \prod_{j \leq t} \gamma_j$, with $\zeta_0 = 1$. Dividing by ζ_t gives

$$\bar{H}_t = H_0 + \sum_{j \leq t} \frac{\omega_j}{\zeta_j} k_j k_j^\top, \qquad \bar{B}_t = B_0 + \sum_{j \leq t} \frac{\omega_j}{\zeta_j} y_j k_j^\top.$$

Thus ordinary positive weighted RLS applies to

$$\widetilde{k}_t = \sqrt{\omega_t/\zeta_t}\, k_t, \qquad \widetilde{y}_t = \sqrt{\omega_t/\zeta_t}\, y_t. \tag{116}$$

The physical inverse is $P_t = \bar{P}_t/\zeta_t$ and the predictor is $W_t = \bar{W}_t$. A weight ω_t selected from current and preceding observations enters the weighted interaction core after the known rescaling. Likelihood-dependent nonlinear rules require the corresponding transformed local definition.

H.4 Diagonal transport of information coordinates

Another useful quadratic recurrence transports the information matrix directly:

$$H_t = F_t H_{t-1} F_t^\top + k_t k_t^\top, \qquad B_t = \delta_t B_{t-1} F_t^\top + y_t k_t^\top,$$

with known invertible diagonal F_t and nonzero scalar δ_t. Define $\Phi_t = F_t \cdots F_1$ and $\xi_t = \prod_{j \leq t} \delta_j$. The transformed statistics

$$\bar{H}_t = \Phi_t^{-1} H_t \Phi_t^{-\top}, \qquad \bar{B}_t = \xi_t^{-1} B_t \Phi_t^{-\top}$$

obey ordinary RLS accumulation with $\bar{k}_t = \Phi_t^{-1} k_t$ and $\bar{y}_t = y_t/\xi_t$. This follows by substituting the recurrences and canceling the adjacent coordinate factors. Recover the physical states and queries by

$$P_t = \Phi_t^{-\top} \bar{P}_t \Phi_t^{-1}, \qquad W_t = \xi_t \bar{W}_t \Phi_t^{-1}, \qquad W_t q = \xi_t \bar{W}_t (\Phi_t^{-1} q).$$

The prescribed diagonal operations add elementwise preparation and read scaling. This information-coordinate recurrence and the covariance-transition model each specify their own physical dynamics.

H.5 Other state operations and their cost

A known scalar multiplication of the mean can be represented in rescaled mean coordinates. For $W_t^- = g_t W_{t-1}$, define $\eta_t = \prod_{j \leq t} g_j$ with nonzero g_j and write $W_t = \eta_t \bar{W}_t$. The transformed mean coefficient is $\bar{u}_t = u_t/\eta_t$. The local rule receives the physical prediction $p_t = \eta_t \bar{p}_t$. The accumulated scale is therefore part of the rule's input. This transformation specifies a recurrence.

Retaining a fixed ridge under fading adds an information term at each step. If $H_t = S_t + \Lambda$ and $S_t = \gamma_t S_{t-1} + \omega_t k_t k_t^\top$, then

$$H_t = \gamma_t H_{t-1} + (1 - \gamma_t)\Lambda + \omega_t k_t k_t^\top. \tag{117}$$

The rank of the added information term is generally d. Its execution requires additional structure or additional matrix work. Objective compatibility is consequently checked at the level of the full recurrence when integrating a regularized regression memory such as MesaNet (von Oswald et al., 2026).

The dynamic Gram construction above assumes that its coordinatewise process increments and transitions are available before the chunk computation. Arbitrary coordinatewise choices driven by intermediate responses require their own preparation algorithm. A small known basis of allowable operator changes is one way to formulate that extra structure. Scalar adaptive information weights already satisfy the assumptions of the main theorem.

I KERNEL MEMORIES, SIGNED INFORMATION, AND SITE UPDATES

I.1 KERNEL EVALUATION OF THE ORDERED UPDATE REPRESENTATION

The static feature-space algebra depends on the features through $\kappa_0(x, x') = k(x)^\top P_0 k(x')$ and the entry predictor $f_0(x)$. A positive-definite kernel can supply these quantities directly. Let x_j be the input at step j, and define the adapted kernel direction

$$a_j(x) = \kappa_0(x, x_j) - \sum_{\ell<j} \beta_\ell a_\ell(x) a_\ell(x_j). \tag{118}$$

Then the mean and covariance kernel after update c are

$$f_c(x) = f_0(x) + \sum_{j\le c} u_j a_j(x), \tag{119}$$

$$\kappa_c(x, x') = \kappa_0(x, x') - \sum_{j\le c} \beta_j a_j(x) a_j(x'). \tag{120}$$

At the current example, $p_t = f_{t-1}(x_t)$ and $s_t = \kappa_{t-1}(x_t, x_t)$. The local rule therefore has exactly the inputs used in the finite-feature theorem. Forming $\Gamma_{ij} = \kappa_0(x_i, x_j)$ and the required cross-kernel matrices evaluates the chunk using example-space quantities.

Gaussian likelihoods give exact GP regression updates under the specified prior kernel and observation noise. Recursive Laplace and assumed-density updates execute their respective Gaussian approximate classification or count-filtering algorithms. Across chunks, finite features provide a fixed-size parameter state; general kernel implementations retain a growing set of basis inputs or use an approximation. These are the settings studied by kernel RLS and sparse online Gaussian-process methods (Engel et al., 2004; Csató & Opper, 2002).

I.2 DICTIONARY ADMISSION AND APPROXIMATE DEPENDENCE

For an invertible dictionary Gram matrix K_D, a common feature-dependence statistic is

$$\delta(x) = \kappa(x, x) - k_D(x)^\top K_D^{-1} k_D(x). \tag{121}$$

It is the squared residual after projecting the new kernel feature onto the dictionary span. Approximate linear dependence sparsification uses a threshold on this residual (Engel et al., 2004). At zero residual, the feature has an exact representation in that span. A positive threshold introduces a dictionary approximation whose influence on later predictions depends on the update rule and accumulated metric. The within-chunk transport theorem can be applied to the chosen finite representation. A full dictionary error guarantee also accounts for admission and compression.

I.3 SIGNED INFORMATION WEIGHTS AND PIVOT MARGINS

The zero-safe coefficient $\beta = w/(1 + ws)$ permits negative weights whenever $h = 1 + ws > 0$. The determinant lemma yields

$$\log\det H_c = \log\det H_0 + \sum_{j\le c} \log(1 + w_j s_j). \tag{122}$$

A negative weight removes information in one direction and increases covariance there. The exact feasibility test is local. Its conditioning also appears in

$$|\beta| = \frac{|w|}{h}, \qquad \left|\frac{\partial\beta}{\partial w}\right| = h^{-2}.$$

A guard $h \ge \varepsilon > 0$ controls these denominators. Accepting, damping, or rejecting a proposed update is part of the declared local rule. Refactorization can check the state and reset roundoff.

For a nonsingular diagonal weight matrix J, the Woodbury innovation matrix is $J^{-1} + KP_0K^\top$. Its diagonal pivots are $w_t^{-1} + s_t$. Negative weights can make that innovation matrix indefinite even when every intermediate parameter-space information matrix is positive definite. An $LDL^\top$ representation handles its algebraic signs; the zero-safe ordered update representation also accommodates exactly zero weights. Indefinite-metric filtering has a substantial Krein-space and robust-estimation literature (Hassibi et al., 1999). A particular H_∞ or risk-sensitive recurrence requires its own signed update mapping and feasibility conditions.

I.4 Gaussian site replacement and expectation propagation

Represent a scalar Gaussian site by natural parameters ν, τ, with factor proportional to $\exp(\nu k^\top x - \tau(k^\top x)^2/2)$. A site replacement adds $\Delta\nu\, k$ to the information vector and $\Delta\tau\, kk^\top$ to the precision. Let the current mean be $\mu = Pb$ and set $p = k^\top \mu$, $s = k^\top Pk$. The new mean is

$$\mu^+ = (H + \Delta\tau kk^\top)^{-1}(b + \Delta\nu k) \tag{123}$$

$$= \mu + \frac{\Delta\nu - \Delta\tau p}{1 + \Delta\tau s} Pk. \tag{124}$$

Thus

$$w = \Delta\tau, \qquad u = \frac{\Delta\nu - \Delta\tau p}{1 + \Delta\tau s}. \tag{125}$$

This covers site insertion, damping, removal, and replacement whenever the intermediate Gaussian states are feasible. Expectation propagation uses cavity distributions and tilted moment computations to choose site parameters (Minka, 2013). The old site values are additional local bookkeeping. A predetermined sequence of revisits uses information in visit order and can supply repeated keys to a chunk. The resulting Gaussian approximation follows the chosen EP algorithm and its declared visit schedule.

I.5 A practical closure criterion

The interface applies when a method's mean changes through the specified adapted directions and its information changes through known low-rank contributions with low-dimensional coefficients computed in observation order. A current-example half-space projection, such as a passive-aggressive update, has this structure. A generic coordinatewise parameter penalty can create mean directions outside the available update span. A nonquadratic mirror map requires its own metric-state representation. A model that generates future keys from intermediate parameter states must also include key generation in its sequential computation. These cases are evaluated by their state equations and preparation costs.

J Storage, reconstruction, and numerical control

J.1 Forward state and a reconstructible update tape

For the static general interface, the boundary equations invert directly when the direction and coefficient arrays are retained:

$$P_0 = P_C + A^\top \mathcal{B} A, \qquad W_0 = W_C - U^\top A. \tag{126}$$

The tape (A, U, β) has $C(d + m + 1)$ entries. For a sequence of length N, storing this tape costs $O(N(d + m + 1))$. Storing the interaction tables adds $O(NC)$ entries; alternatively they can be replayed from the local inputs and the boundary state. The adaptive backward also needs the local-rule inputs or enough information to recompute them.

A boundary-checkpoint strategy stores approximately N/C states of size $d^2 + md$, giving $O((N/C)(d^2 + md))$ entries. Full token-level state storage has $O(N(d^2 + md))$ entries. These formulas describe separate memory–recomputation choices. Their ordering depends on C, d, m and on the backward implementation. All floating-point reconstructions should be checked against the chosen numerical tolerance.

For RLS, a terminal-state-only inverse reconstruction also follows from $\Xi = I - KP_CK^\top = S^{-1}$:

$$P_0 = P_C + P_C K^\top \Xi^{-1} K P_C. \tag{127}$$

Its smallest eigenvalue is $1/\lambda_{\max}(S)$, so a strongly informed chunk can make this reconstruction sensitive. Equation (126) uses the retained correction factors directly. A checkpoint policy can select a route according to available memory and measured residuals.

J.2 Grouping corrections across chunks

In fixed coordinates, successive boundary updates can be written as

$$P = P_\star - \sum_b A_b^\top \mathcal{B}_b A_b, \qquad W = W_\star + \sum_b U_b^\top A_b.$$

Stacking the arrays forms a superblock whose correction products can be applied jointly to a set of queries. Every local rule receives the state actions required by its recurrence. Applying the accumulated factors to new keys costs work proportional to their retained rank. Consolidating the superblock into a dense boundary restores the fixed-size state representation.

J.3 Analytical forward working sets

A dense state contains $d^2 + md$ entries per independent stream. A representative subset of the temporary storage consists of the innovation matrix S, its Cholesky factor L, and the normalized arrays X, E. With S, L in fp64 and X, E in fp32, these arrays occupy

$$M_{SLXE} = 16C^2 + 4C(d+m) \quad \text{bytes per stream.}$$

This calculation excludes inputs, outputs, preparation buffers, library workspace, graph pools, and the backward tape. It reports storage for those named arrays. Actual peak allocation includes additional buffers and runtime workspace. The following values assume separate storage for S and L; in-place reuse can reduce that term.

	M_{SLXE} (KiB per stream), $d = m$		
C	64	128	256
32	32	48	80
64	96	128	192
128	320	384	512
256	1,152	1,280	1,536

The fp32 dense states for these widths occupy 32, 128, and 512 KiB per stream. Multiplication by the number of independent streams gives the corresponding batched storage.

J.4 Local reconstruction sensitivity

A direct perturbation calculation gives a diagnostic bound for the final contraction. Suppose $P = P_0 - A^\top \mathcal{B} A$, with perturbations $\delta P_0, \delta A, \delta \mathcal{B}$. Then

$$\begin{aligned}\|\delta P\|_2 \le \|\delta P_0\|_2 &+ \|\mathcal{B}\|_2(2\|A\|_2\|\delta A\|_2 + \|\delta A\|_2^2) \\ &+ \|\delta \mathcal{B}\|_2(\|A\|_2 + \|\delta A\|_2)^2. \end{aligned} \tag{128}$$

For $W = W_0 + U^\top A$,

$$\|\delta W\|_2 \le \|\delta W_0\|_2 + \|U\|_2\|\delta A\|_2 + \|A\|_2\|\delta U\|_2 + \|\delta U\|_2\|\delta A\|_2. \tag{129}$$

Both bounds follow by expanding the perturbed products and applying submultiplicativity. They show which factor magnitudes can amplify local numerical errors.

For a whole adaptive sequence, suppose the exact step map is L_t-Lipschitz on the visited region and the implementation has local defect at most δ_t. With state error e_t, the triangle inequality gives $e_t \le L_t e_{t-1} + \delta_t$, hence

$$e_T \le e_0 \prod_{j=1}^{T} L_j + \sum_{i=1}^{T} \delta_i \prod_{j=i+1}^{T} L_j. \tag{130}$$

A bound on these constants is specific to the learner, metric, and local solver. The numerical experiments therefore report observed state and output errors together with conditioning. A long-horizon guarantee requires the corresponding learner-specific assumptions.

Table 10: Execution routes for Equation (12). The terminal block route evaluates the final fitted state. FlashRLS retains chronological correction rows for reads at every update and the boundary update.

Route	Work organization	Requested outputs
Sequential RLS	C ordered dense-state updates	Reads emitted at each step
Terminal block solve	One grouped state correction	Final state and final-state reads
Separate state solves	One solve for each intermediate state	Intermediate reads with repeated work
FlashRLS	Shared chunk factor, transformed rows, masked products	Reads at every update and next state

J.5 Residual certificate for direction reconstruction

The following diagnostic supplements the perturbation bounds above. It isolates numerical error in the triangular reconstruction for a fixed local interaction core.

Proposition J.1 (Fixed-core solve residual and state error)**.** *Fix a unit lower-triangular T, a right-hand side B_0, coefficients $U, \mathcal{B}$, and incoming states P_0, W_0. Let $A = T^{-1}B_0$, and let $\widehat{A}$ have residual $R = B_0 - T\widehat{A}$. All norms below are spectral norms. Set $\varepsilon_A = \|T^{-1}\|\,\|R\|$. Suppose the computed states satisfy*

$$\widehat{P} = P_0 - \widehat{A}^\top \mathcal{B}\widehat{A} + E_P, \qquad \widehat{W} = W_0 + U^\top \widehat{A} + E_W,$$

where E_P, E_W include the defects of their numerical matrix products and additions. Relative to $P = P_0 - A^\top \mathcal{B}A$ and $W = W_0 + U^\top A$,

$$\|\widehat{A} - A\| \le \varepsilon_A, \tag{131}$$

$$\|\widehat{P} - P\| \le \|\mathcal{B}\|(2\|\widehat{A}\|\varepsilon_A + \varepsilon_A^2) + \|E_P\|, \tag{132}$$

$$\|\widehat{W} - W\| \le \|U\|\varepsilon_A + \|E_W\|. \tag{133}$$

Proof. Put $D = \widehat{A} - A = -T^{-1}R$. Submultiplicativity gives the first bound. With $A = \widehat{A} - D$,

$$\widehat{A}^\top \mathcal{B}\widehat{A} - A^\top \mathcal{B}A = \widehat{A}^\top \mathcal{B}D + D^\top \mathcal{B}\widehat{A} - D^\top \mathcal{B}D.$$

The triangle inequality gives the covariance bound. The predictor difference $U^\top D + E_W$ gives the final bound. □

The same bounds apply to the state after any update by restricting all arrays to the observations used. A certified upper bound on $\|T^{-1}\|$ and an upper bound on the true residual norm make the result a computable error certificate; a floating-point residual evaluation must itself have its rounding error accounted for. Perturbations in the local roots, curvature weights, T, or incoming state are treated by the preceding sensitivity analysis and the bounds for propagation through later updates of Appendix F.

J.6 Per-head storage accounting

For B sequences and n_h heads, fp32 dense RLS state occupies $4Bn_\mathrm{h}(d^2 + md)$ bytes. A predictor-only state occupies $4Bn_\mathrm{h}md$ bytes. Table 11 evaluates these formulas at widths 256 and 512. These are analytical array counts.

For N/C dense boundary checkpoints, the checkpoint arrays use $4Bn_\mathrm{h}(N/C)(d^2 + md)$ bytes. Retaining normalized write and residual arrays for all observations uses $4Bn_\mathrm{h}N(d+m)$ bytes. These are separate storage choices; neither includes every activation or workspace of a complete learned layer. A scheme that reads and writes one dense state per chunk would transfer $8(d^2 + md)/C$ bytes per observation per head for those state arrays. At $m = d$ and $C = 64$, this quantity is 16 KiB for $d = 256$ and 64 KiB for $d = 512$. Actual device traffic depends on reuse, tiling, and fusion and is not measured by these formulas.

Table 11: **Analytical forward-state accounting.** MiB for one sequence ($B = 1$), $m = d$, and fp32 arrays. Dense RLS stores P, W; predictor-only stores W. Counts exclude activations and workspace.

d	Heads	Dense RLS	Predictor-only
256	8	4.00	2.00
256	16	8.00	4.00
512	8	16.00	8.00
512	16	32.00	16.00

K Numerical tests and accelerator experiments

K.1 Numerical identity checks

Identity tests use float64 arithmetic and independent dense information-matrix references. For arrays A, B, absolute error is $\max_i |A_i - B_i|$; relative error is $\|A - B\|_F / \|B\|_F$. A state trial takes the maximum over all tested intermediate states. Factor residual is $\|S - LL^\top\|_F / \|S\|_F$. CPU tests use Python 3.13.5, NumPy 2.3.5, SciPy 1.17.0, and PyTorch 2.10.0, with single-threaded BLAS and PyTorch.

The adaptive test uses $C = 13$, $d = 7$, and 30 seeds per local rule. Keys and independent queries are Gaussian with scale $1/\sqrt{d}$. The entry covariance is $P_0 = 0.15I + 0.2BB^\top$, with B drawn at the same scale, and entry means have scale 0.7. The dense reference advances H, recomputes H^{-1}, and applies the same mean rule. Parameters are ONS mean multiplier 0.7, unit-weight logistic scoring and Laplace, 80-point Gauss–Hermite quadrature for logistic ADF, AROW parameter 0.8, PA-I cap 0.3, and Student-t with $\nu = 3$, scale 0.8, and multiplier 0.15. Scalar root tolerances are 10^{-14}.

Table 12: Adaptive numerical agreement. Thirty float64 trials per rule. "All" includes the means after each update, covariances, queries, local predictions, variances, and update directions.

Local rule	Mean	Covariance	Query	All
Unprojected ONS (logistic)	4.44×10^{-16}	3.33×10^{-16}	6.66×10^{-16}	8.88×10^{-16}
One-step scoring (logistic)	4.44×10^{-16}	4.44×10^{-16}	6.66×10^{-16}	8.88×10^{-16}
Recursive Laplace (logistic)	4.44×10^{-16}	3.33×10^{-16}	6.11×10^{-16}	6.66×10^{-16}
ADF (logistic, quadrature)	8.88×10^{-16}	3.89×10^{-16}	8.88×10^{-16}	1.11×10^{-15}
AROW	7.22×10^{-16}	2.22×10^{-16}	6.66×10^{-16}	8.88×10^{-16}
PA-I, fixed metric	6.66×10^{-16}	1.67×10^{-16}	4.44×10^{-16}	6.66×10^{-16}
Adaptive signed / zero weights	4.44×10^{-16}	4.44×10^{-16}	4.44×10^{-16}	8.88×10^{-16}
Recursive Laplace (Student-t)	6.66×10^{-16}	4.44×10^{-16}	8.88×10^{-16}	8.88×10^{-16}

For the conditioning test, $U \in \mathbb{R}^{16\times16}$ and $V \in \mathbb{R}^{24\times16}$ have orthonormal columns. We set $K = U\,\mathrm{diag}(\sqrt{\lambda_i - 1})V^\top$, $P_0 = I$, with λ_i geometrically spaced from 1 to κ. Thus S has the prescribed spectrum. Each row uses ten seeds, three response columns, Gaussian targets, and independent queries. Every query is compared with a direct solve after the same observations.

Table 13: Innovation conditioning. Ten float64 trials per row. Query discrepancy is absolute; factor residual is relative Frobenius error.

Target $\kappa_2(S)$	Factor residual	Query error	Minimum $\lambda(P_C)$
1	0	2.22×10^{-16}	1.00×10^{0}
10^2	7.86×10^{-17}	5.38×10^{-15}	10.00×10^{-3}
10^4	8.18×10^{-17}	2.18×10^{-13}	10.00×10^{-5}
10^6	9.32×10^{-17}	8.84×10^{-12}	10.00×10^{-7}
10^8	9.08×10^{-17}	9.16×10^{-10}	10.00×10^{-9}

Composition tests use 39 observations, $d = 7$, ten seeds per rule, and chunk endpoints $0, 5, 18, 26, 39$. Future-target tests flip targets starting at zero-based index 19 and compare pre-update reads through that index. Complete gradient tests use $C = 8$, $d = 5$, ten seeds, $P_0 = 0.2I + 0.1RR^\top$, and the

objective $\sum_t \sin(o_t) + 0.1\|\mu_C\|^2 + 0.03\|P_C\|_F^2$. Dense and block computation graphs differentiate with respect to K, Q, R, μ_0, including response-dependent curvature. Centered finite differences use step 10^{-5}.

Table 14: Composition, future-target exclusion, gradients, and local solves. Maximum absolute discrepancies.

Check	Maximum discrepancy
Chunk partition agreement (80 trials)	6.66×10^{-16}
Future-target leakage (80 trials)	0
Complete adaptive gradient agreement (10 trials)	8.88×10^{-16}
Adaptive directional finite difference (10 trials)	1.94×10^{-11}
Softmax structured solve vs. dense solve	1.33×10^{-15}
Poisson scalar solution vs. bracketed solve	2.84×10^{-14}

K.2 Long-horizon state stability

The stream has 65,536 fp32 observations in 1,024 chunks, $d = C = 64$, AR(1) keys with $\rho \in \{0, 0.9, 0.99, 0.999\}$, and $\kappa_2(H_0) \in \{10^2, 10^4, 10^6, 10^8\}$. Every recurrence starts from the same fp32 state and receives the same keys. The correlation sweep uses seed 77; the extended stabilized-baseline comparison uses three key-stream seeds. Final inverse drift is $\|\widehat{P} - P_{64}\|_F / \|P_{64}\|_F$ after the entire free-running stream. Rebuilds come from separate guarded trajectories.

Plain Sherman–Morrison subtracts the rank-one correction. The symmetric variant replaces P by $(P + P^\top)/2$ after each observation. For observation weight $\omega > 0$ and gain $g = \omega Pk/(1 + \omega k^\top Pk)$, Joseph form evaluates

$$P^+ = (I - gk^\top)P(I - gk^\top)^\top + gg^\top/\omega.$$

The token QRD-RLS baseline updates a triangular information factor by rotations (Bierman, 1977; Hsieh et al., 1993). Its inverse actions are evaluated from triangular solves, and the table below compares the resulting inverse with the same fp64 inverse reference. The block QRD implementation and its separate factor-state diagnostic are reported in Appendix N. At $\kappa_2(H_0) = 10^4$, FlashRLS

Table 15: Final inverse drift in the stabilized-baseline study. Mean $\pm$ standard deviation over three key-stream seeds, except the explicitly labeled seed-77 correlation ranges.

	10^2	10^4	10^6	10^8
FlashRLS (seed 77, ρ range)	$[4.3, 10] \times 10^{-7}$	$[2.7, 4.3] \times 10^{-6}$	$[4.9, 7.9] \times 10^{-6}$	$[6.2, 9.2] \times 10^{-6}$
Plain SM (seed 77, ρ range)	$[9.4, 15.8] \times 10^{-6}$	$[2.1, 3.7] \times 10^{-5}$	$[3.4, 6.3] \times 10^{-5}$	$[4.1, 6.5] \times 10^{-5}$
Symmetrized SM	$(8 \pm 2) \times 10^{-6}$	$(1.9 \pm 0.5) \times 10^{-5}$	$(3.5 \pm 1) \times 10^{-5}$	$(4.5 \pm 1) \times 10^{-5}$
Joseph form	$(6.5 \pm 1.5) \times 10^{-6}$	$(1.5 \pm 0.4) \times 10^{-5}$	$(2.6 \pm 0.8) \times 10^{-5}$	$(4 \pm 1) \times 10^{-5}$
Token QRD	$(4.5 \pm 1.5) \times 10^{-7}$	$(5.5 \pm 1.5) \times 10^{-7}$	$(6.5 \pm 1.5) \times 10^{-7}$	$(9 \pm 3) \times 10^{-7}$

and Joseph require no rebuilds; symmetrized SM requires 0.5 ± 0.5. At 10^6, the seed-77 correlation ranges are 16–81 rebuilds for FlashRLS and 168–407 for plain SM. The three-seed FlashRLS study gives $(6 \pm 1.3) \times 10^{-6}$ drift and 40 ± 30 rebuilds. At 10^8, all inverse-state variants rebuild on every chunk; token QRD requires 2 ± 2 rebuilds. These operating points use the same threshold.

K.3 Guard and reconstruction cost

A width-eight probe Ω checks $r_\Omega = \|(H\widehat{P} - I)\Omega\|_F / \|\Omega\|_F$ at each boundary. A value above 5×10^{-4} triggers reconstruction from fp64 information and response statistics, using a fresh factorization and triangular solves. The projected diagnostic measures the sampled directions. Updating the sufficient statistics adds $O(C(d^2 + dm))$ work per chunk and $O(d^2 + dm)$ storage; the probe costs $O(8d^2)$; a rebuild costs $O(d^3 + d^2 m)$. For J chunks and R rebuilds, the total work is

$$J\{T_{\text{update}} + T_{\text{statistics}} + T_{\text{probe}}\} + R\,T_{\text{rebuild}}.$$

Kernel benchmarks disable the guard. Guarded AEC timings include the complete execution. The estimator's ridge updates belong to the recurrence and are independent of numerical recovery.

K.4 Kernel timing and exact-operator scaling

The common-shape benchmark uses $B = n_{\mathrm{h}} = 8$, $d = m = 128$, $C = 64$, and bf16 activations. Each seed uses ten warmups and 50 CUDA-graph replays; the median is taken within seed, followed by the mean and sample standard deviation across three input seeds. Times include forward and backward. MesaNet and Gated DeltaNet use their FLA kernels. FlashRLS uses fp32 state and an fp64 innovation core.

Table 16: Common-shape forward-plus-backward time (ms). Head and chunk dimensions are shared across columns.

Method	$N = 512$	$N = 2{,}048$	$N = 8{,}192$
FlashRLS	6.85 ± 0.02	24.1 ± 0.1	132 ± 1
MesaNet CG-30	1.59 ± 0.01	6.09 ± 0.03	23.4 ± 0.1
MesaNet CG-100	3.51 ± 0.02	13.4 ± 0.1	52.7 ± 0.2
Gated DeltaNet	0.42 ± 0.01	1.56 ± 0.01	6.1 ± 0.05

The exact-operator scaling study uses fp32 inputs, ridge one, correlation 0.9, and nonzero initial state. Both implementations receive identical quantized arrays. Timing includes packing, every post-update read, terminal P, W, and derivatives through all three outputs. TF32 is disabled. The reference implementation is a tokenwise PyTorch recurrence with autograd. The largest two backward checks evaluate eight heads of the first batch element; the other checks evaluate the full batch.

Table 17: Exact-operator implementation comparison. Median forward-plus-backward ms, 50 graph replays; $B = n_{\mathrm{h}} = 8$, $C = 64$.

N	$d = m$	Tokenwise PyTorch	FlashRLS	Speedup
512	128	101.51	8.03	12.64×
2,048	128	835.80	38.14	21.91×
8,192	128	9618.42	249.58	38.54×
512	64	67.45	6.31	10.69×
512	256	238.48	14.04	16.98×
512	512	718.00	33.43	21.48×

The $N = 512, d = 128$ output and largest input-gradient relative errors are 4.38×10^{-7} and 5.13×10^{-7} against the independent float64 RLS reference. At $d = 128$, chunk lengths $16, 32, 64, 128$ take $22.14, 12.53, 8.01, 7.43$ ms. At $d = 512$, chunk lengths $32, 64, 128$ take $53.1, 33.4, 29.2$ ms. These sweeps use one numerical block per chunk.

K.5 Objective-specific fp64 solver references

Each implementation is compared with an fp64 transcription of its recurrence. The FlashRLS target is the general RLS objective in Equation (12), including its specified weights and transitions. The MesaNet target transcribes the FLA recurrence with SiLU and normalized keys and queries, per-head and per-dimension regularization Λ, and decay and write gates. It uses the diagonal initial guess for CG. The parameter mapping to FlashRLS is checked on one observation. Accuracy is measured separately against the two complete-sequence targets.

The test uses $N = 2{,}048$, three input seeds for MesaNet, $\rho = 0.99$, and regularizers 1 and 0.01. Table 18 gives both input precisions. FlashRLS entries are the point measurements from the exact-operator accuracy sweep. The tolerance-based MesaNet row uses stopping threshold 10^{-8}. At low ridge in fp32, the tolerance-based MesaNet solve reaches $(5 \pm 3) \times 10^{-5}$ relative error, close to the 5.1×10^{-5} FlashRLS value against its own target. Fixed-weight solver substitutions in Appendix L measure the prediction effect without changing the learned projections.

Table 18: Relative read error against each method's fp64 target. MesaNet entries are mean ± standard deviation over three input seeds.

	Regularizer 1		Regularizer 0.01	
Method	bf16	fp32	bf16	fp32
FlashRLS	5.8×10^{-5}	9.6×10^{-7}	4.1×10^{-4}	5.1×10^{-5}
MesaNet CG-30	$(4 \pm 1) \times 10^{-3}$	$(3 \pm 1) \times 10^{-4}$	$(2 \pm 1) \times 10^{-2}$	$(5 \pm 2) \times 10^{-3}$
MesaNet CG-100	$(4 \pm 1) \times 10^{-3}$	$(1 \pm 0.5) \times 10^{-4}$	$(6 \pm 2) \times 10^{-3}$	$(3 \pm 1) \times 10^{-4}$
MesaNet, tol. 10^{-8}	$(4 \pm 1) \times 10^{-3}$	$(3 \pm 1) \times 10^{-5}$	$(5 \pm 2) \times 10^{-3}$	$(5 \pm 3) \times 10^{-5}$

L Regression, learned memory, and storage experiments

L.1 In-context linear regression

Tasks have $d = 64$, Gaussian observation noise with standard deviation 0.1, and an equal-signal teacher: expected signal is the same in each covariance eigen-direction. The condition number $\kappa = \lambda_{\max}(\Sigma)/\lambda_{\min}(\Sigma)$ is the ratio of the largest to smallest eigenvalue of the feature covariance. Learners make pre-update predictions. First-order learning rates are tuned on separate tasks. The primary metric is trailing-window MSE over 64 observations divided by the noise variance $\sigma^2 = 0.01$. The first-crossing count is the number of observations until this normalized MSE first falls to two (unnormalized MSE 0.02); lower counts indicate faster learning.

The 4,096-observation study uses three seeds with 64 tasks each. At ridge 10^{-4}, RLS reaches the threshold after 163–173 observations across covariance condition numbers through 10^4. Raising ridge to 0.01 at $\kappa = 10^4$ moves the crossing to 1442–1534 observations. The extended study uses 32,768 observations, 64 tasks, and seed 0; Table 19 gives the corresponding crossing times; the stored trajectories are evaluated under the same setup. The seed-0 4,096-observation sweep covers $\kappa \in \{1, 10, 10^2, 10^3, 10^4\}$ at ridge 10^{-4} and appears under its own horizon heading in Table 19.

Table 19: Observations until trailing 64-observation MSE reaches 0.02, twice the noise variance. Both sweeps use $d = 64$, ridge 10^{-4}, 64 tasks, and seed 0. κ is the feature-covariance condition number; n/r denotes no crossing within 4,096 observations.

	32,768 observations		4,096 observations				
Learner	$\kappa = 10^2$	$\kappa = 10^3$	$\kappa = 1$	$\kappa = 10$	$\kappa = 10^2$	$\kappa = 10^3$	$\kappa = 10^4$
Exact RLS	**168**	**168**	164	164	164	164	173
CG, 8 iterations per observation	**168**	177	164	164	166	177	265
Delta rule	5,512	24,535	718	922	n/r	n/r	n/r
Normalized LMS	5,278	24,500	508	1,035	n/r	n/r	n/r

Exact RLS reaches the criterion after 168 observations at both $\kappa = 10^2$ and 10^3 in the extended-horizon study. The delta rule needs 5,512 and 24,535 observations, respectively, about 33 and 146 times as many. At $\kappa = 10^4$, exact RLS crosses after 173 observations; neither the delta rule nor normalized LMS crosses within 4,096. CG with eight iterations reaches the same criterion after 265 observations. These results motivate accurate full-covariance updates. Appendix L.1 reports additional seeds and ridge settings.

L.2 Multi-query associative recall

MQAR (Arora et al., 2024) draws keys and values from disjoint halves of an 8,192-token vocabulary. A stored key–value block is followed by a query region in which each queried key reappears at a random offset. Loss and exact-match accuracy are computed at answer positions. Models have two layers of width 128 and two 64-dimensional heads, trained for 3,000 steps on newly sampled 256-token sequences with 32 stored pairs. Learned absolute positions are disabled. Name-keyed initialization and data generation align the original trained-memory comparisons.

We evaluate fixed weights at up to 4,096 tokens, up to 256 associations, and with stored distractor pairs that are never queried. The original capacity sweep uses seed 0. MesaNet is trained independently for three seeds, using the same task, convolution, gates, normalization, and output projection, with FLA `chunk_mesa_net`, a learned per-head ridge $0.25 + \mathrm{softplus}(\cdot)$, and 30 CG iterations. Solver substitutions change the inference solve on a fixed checkpoint and fixed inputs. At 4,096 tokens

Table 20: MQAR accuracy (%). Original trained memories and their solver substitutions use seed 0. Independently trained MesaNet entries are mean ± standard deviation over three seeds.

Memory	64 pairs	128 pairs	224 pairs	256 pairs
Gated DeltaNet	100.0	93.7	57.3	46.5
FlashRLS, ridge 1	100.0	98.9	74.5	62.2
FlashRLS, ridge 0.01	100.0	99.9	90.8	83.7
CG-30 substitution	100.0	99.9	90.0	82.5
CG-8 substitution	100.0	99.6	88.9	79.1
MesaNet CG-30	n/a	98 ± 1	88 ± 2	80 ± 2
MesaNet CG-8 at evaluation	n/a	97 ± 1	85 ± 3	76 ± 3

with 32 pairs, the original three trained-memory configurations each reach 100% accuracy. Gated DeltaNet reaches 100% training accuracy by 2,000 steps at seed 0; the two exact models do so before 1,000 steps. MesaNet requires $1{,}200 \pm 200$ steps over three seeds. The recurrent state accounting is $d^2 + md$ entries per head for dense RLS and MesaNet, md for Gated DeltaNet. These counts exclude network parameters and training activations.

L.3 FineWeb-Edu language modeling

This separate 50M learned-memory experiment uses one billion FineWeb-Edu training tokens (Penedo et al., 2024), GPT-2 byte-pair tokenization (Radford et al., 2019), and 2,048-token windows with state reset at each window. A width-512, eight-layer backbone has four heads of width 128, SwiGLU expansion $8/3$ (Shazeer, 2020), RMSNorm (Zhang & Sennrich, 2019), tied embeddings, width-four query/key/value convolutions using the current and preceding tokens, and SiLU followed by ℓ_2 normalization on keys and queries. Gated models use per-head decay and write gates. MesaNet uses CG-30 in training and evaluation.

Optimization uses AdamW (Loshchilov & Hutter, 2019) with $(0.9, 0.95)$ moments, weight decay 0.1, 0.5M-token effective batches, clipping at 1, 2% linear warmup, and cosine decay to 10% of peak rate. A 100M-token scan selects the peak rate for each mixer from $\{3, 6, 12\} \times 10^{-4}$. Three seeds vary initialization and data shuffle; each seed shares its data order across mixers. Activations use bf16 and the recurrent state uses fp32. Whole-model throughput includes data, forward, backward, clipping, and optimizer steps. Evaluation and graph capture are outside the timed interval.

FineWeb-Edu validation NLL is the mean token negative log likelihood on two million held-out tokens. WikiText-103 perplexity is a zero-shot token-level evaluation on that corpus (Merity et al., 2017). The copyable-token slice restricts the validation comparison to the benchmark's copyable-token positions. Table 21 gives the three-seed comparison, including Gated DeltaNet and MesaNet.

Table 21: 50M FineWeb-Edu comparison. Mean ± standard deviation over three seeds. Throughput is the complete training step on one H100.

Mixer	FineWeb val NLL	WikiText PPL	Copyable ΔNLL	MQAR@224	k token/s
Gated DeltaNet	2.85 ± 0.01	36.0 ± 0.3	0	57 ± 3	300
MesaNet CG-30	2.84 ± 0.01	35.6 ± 0.3	-0.02 ± 0.01	88 ± 2	190
FlashRLS, decay/write gates	2.84 ± 0.01	35.6 ± 0.4	-0.02 ± 0.01	90 ± 2	60
FlashRLS, ungated	2.90 ± 0.01	40 ± 1	-0.03 ± 0.01	90 ± 2	60

L.4 WIKITEXT-103 TRAINING AND FIXED-WEIGHT SOLVER SUBSTITUTIONS

The smaller experiment trains 16.4M-parameter models for 20M WikiText-103 tokens (Merity et al., 2017). It uses width 256, four layers, four 64-dimensional heads, tied embeddings, learned positions, 512-token windows, and memory reset at each window. Training has 2442 steps of 8,192 tokens, AdamW with warmup and cosine decay, three seeds, name-keyed initialization, and shared minibatch order. Validation uses 65,536 held-out tokens. The ungated exact model has ridge one. The gated exact and independently trained MesaNet variants use MesaNet-style gates. The 160 ± 2 result below belongs to this three-seed study. The ten-seed WikiText experiment in Appendix P reports MesaNet and FlashRLS gate parameterizations as separate variants. The ungated exact NLLs

Table 22: 20M-token WikiText-103 training. Validation perplexity and training cost over three seeds. The ungated exact and Gated DeltaNet rows also have per-seed NLL below.

Mixer	Validation PPL	Wall time (s)	k token/s
Gated DeltaNet	155.0 ± 1.0	75.3 ± 0.5	265.7 ± 1.6
FlashRLS, ungated	174.3 ± 1.1	178.6 ± 0.4	112.0 ± 0.3
FlashRLS, MesaNet-style gates	160 ± 2	n/a	n/a
MesaNet CG-30	158 ± 2	n/a	150
MesaNet CG-8 at evaluation	158.6 ± 2	n/a	n/a

are $5.1590, 5.1674, 5.1550$; Gated DeltaNet NLLs are $5.0420, 5.0381, 5.0508$. Their paired gap is 0.117 ± 0.013 nats. The MesaNet-gated exact variant improves perplexity from 174.3 ± 1.1 to 160 ± 2. Seed-0 geometry probes sample 32 chunks per layer. Median $\log_{10} \kappa_2(H)$ is $1.75, 1.80, 1.84, 1.97$

Table 23: Solver substitutions on the same 20M-token exact checkpoints and validation tokens. Mean $\pm$ standard deviation of the paired NLL difference over three seeds.

Inference solver	ΔNLL from exact (nats)
CG-30	$-7 \times 10^{-6} \pm 5.5 \times 10^{-5}$
PCG-30	$-1.7 \times 10^{-5} \pm 2.4 \times 10^{-5}$
CG-8	0.00400 ± 0.00024

across the four layers; key participation ranks are $8, 10, 12, 13$ out of 64.

L.5 GATED KALMANET PROTOCOL AND PAIRED SOLVER STUDY

Published configuration. Gated KalmaNet (Peng et al., 2026) uses DCLM pretraining with the GPT-2 tokenizer (Radford et al., 2019). Its memory is

$$H_t = \gamma_t H_{t-1} + k_t k_t^\top, \qquad U_t = \gamma_t U_{t-1} + v_t k_t^\top,$$

with the read $U_t \, \mathrm{CH}(H_t + \lambda_t I, q_t, 30)$. The regularizer is $\lambda_t = a\|H_t\|_F$ with default $a = 0.02$, giving $\kappa \le (a+1)/a = 51$. The layer uses learned per-token decay and an α-connection that blends the query with the solve. The published 440M, 1B, and 2.8B models use 8B, 20B, and 100B training tokens; the corresponding context lengths are 2048, 2048, and 4096. The long-context study continues pretraining at 128K context.

At 2.8B, Gated KalmaNet reports 58.89 average accuracy over ten LM-Harness tasks and FDA/SWDE scores 12.89/50.95. Gated DeltaNet reports 57.00 and 8.26/44.28. The adaptive-weighting ablation gives 58.89 with gates and 50.45 without them. The long-context regularization sweep is strongest at $a = 0.05$ and lower at $a = 0.02$ and $a = 0.01$; the paper attributes the low-a decline to the increasing condition number reducing Chebyshev solution quality. Its five-layer backward study reports order-one error for implicit CG and approximately 10^{-6} error for the selected Chebyshev construction. The published throughput at 2.8B is 7.65 billion tokens per day per H200 node. These published models are used as motivation and are not reproduced here.

Paired 50M protocol. All paired rows use width 512, eight layers, four heads of dimension 128, SwiGLU, RMSNorm, tied embeddings, conv-4 on q, k, v, SiLU and L2-normalized keys and

queries, Gated KalmaNet learned decay, and its α-connection. Training uses SlimPajama (Soboleva et al., 2023), the GPT-2 tokenizer (Radford et al., 2019), 2,048-token context, one billion tokens, AdamW $(\beta_1, \beta_2) = (0.9, 0.95)$, weight decay 0.1, 0.5M-token batches, 2% warm-up, cosine decay, gradient clipping 1.0, bf16 activations, and fp32 recurrent state. The learning rate is selected from $\{3, 6, 12\} \times 10^{-4}$ on a 100M-token scan for CH-30 at $a = 0.02$ and is then fixed. Three seeds share initialization and data order across paired solver rows.

Exact realization of the adaptive ridge. Gated KalmaNet changes $\lambda_t = a\|H_t\|_F$ every token. The linear FlashRLS block theorem assumes a fixed incoming information matrix during one numerical block. The exact rows therefore evaluate

$$\lambda_c = a\|H_{[c]}\|_F$$

at every chunk boundary. Let $\bar{\gamma}_c$ be the cumulative fading applied to the previous ridge over that chunk. The full-rank change is

$$\Delta_c = \lambda_c - \bar{\gamma}_c \lambda_{c-1}, \qquad P \leftarrow (P^{-1} + \Delta_c I)^{-1}.$$

The update is computed with one batched fp64 $d \times d$ Cholesky solve per head and chunk. The ridge is then fixed at λ_c for the next $C = 64$ observations, so the FlashRLS reads at every update in that block are exact for the chunk-frozen objective. This is the additional matrix work identified by Appendix H. At $d = 128, C = 64$, it adds approximately $d^3/3$ fp64 flops per chunk and raises the measured exact-kernel cost by about $1.7\times$ relative to the fixed-ridge kernel. Native GKA CH-30 is evaluated with per-token λ_t. A second CH-30 arm uses the same chunk-frozen λ_c and is the paired solver baseline for the exact rows. The native-versus-chunk-frozen check reported here covers $a = 0.02$, where the CH rows agree within seed variation. The low-a solver comparisons use the common chunk-frozen schedule. A per-token direct treatment based on repeated full-rank shifts is outside the implementation evaluated here.

Projected-inverse guard. The guarded exact rows maintain fp64 fading normal equations H_t, G_t alongside the fp32 recurrent state. At every chunk boundary, eight fixed Gaussian probes form

$$r_t = \frac{\|(H_t P_t - I)\Omega\|_F}{\|\Omega\|_F}.$$

If $r_t > 5 \times 10^{-4}$, an fp64 Cholesky factorization rebuilds $P_t = H_t^{-1}$ and $W_t = P_t G_t$, and the rebuilt state is cast to fp32. The operation restores the same normal-equation estimator and resets accumulated inverse-state roundoff. Rebuild events are counted. The measured rates are zero at $a \geq 0.01$, 2.7 ± 1.5 per million chunks at $a = 0.005$, and 38 ± 9 per million chunks at $a = 0.002$. These rows measure the guarded implementation's performance.

Solver grid and compute. The paired grid contains CH-30 with native per-token λ_t, CH-30 with chunk-frozen λ_c, direct FlashRLS with the same chunk-frozen λ_c, and $a \in \{0.02, 0.01, 0.005\}$. The low-ridge frontier adds bf16 CH-100, bf16 CH-300, fp32 CH-100, and bf16 CG-30 inside the GKA layer at $a = 0.005$; $a = 0.002$ adds CH-30, CH-100, fp32 CH-100, and the direct chunk-ridge solve. Reference rows use Gated DeltaNet and MesaNet CG-30 under the same backbone. Per one-billion-token run, measured wall times are approximately 0.9 h for Gated DeltaNet, 1.5 h for MesaNet CG-30, 1.4 h for GKA CH-30, 2.4 h for bf16 CH-100, 5.2 h for bf16 CH-300, 6.5 h for fp32 CH-100, and 7.0 h for direct chunk-ridge FlashRLS.

At $a = 0.005$, the direct chunk-ridge MQAR values are 91.2, 88.4, and 87.9; CH-30 gives 84.6, 81.0, and 82.9. The paired direct-minus-CH differences are $+6.6, +7.4, +5.0$, with mean $+6.4$ and standard deviation 1.2. At $a = 0.01$, the paired difference is $+2.1 \pm 1.3$ points. The three S-NIAH 8K differences at $a = 0.005$ are all $+8$ points; S-NIAH follows the RULER single-needle retrieval benchmark (Hsieh et al., 2024). WikiText-103 perplexity at the same setting is 35.2/35.7/35.5 for direct chunk-ridge and 35.6/36.3/35.9 for CH-30.

The solver study evaluates the accuracy available under the Gated KalmaNet regularizer. Gated KalmaNet couples $a\|H\|_F$ regularization to a finite Chebyshev solve. The bf16 solver reaches a precision floor near 5×10^{-3}; the low-a metrics remain unchanged as its iteration count increases from 100 to 300. FP32 CH-100 reaches the same frontier as the direct factorization at comparable whole-model cost. MesaNet uses a different learned-ridge CG design and remains an efficient reference point. The paired experiment therefore isolates the conditioning and precision constraint of the Gated KalmaNet design.

Table 24: Low-ridge iterative frontier inside the Gated KalmaNet layer. Mean ± standard deviation over three paired seeds. Every GKA solver row uses the same chunk-frozen ridge schedule. Output error is $\|\widehat{y} - y_{64}\|_F / \|y_{64}\|_F$ against the fp64 solution of the same system with the same inputs; input precision is stated per row.

Solver	Prec.	a	Slim PPL	MQAR@224	S-NIAH 4K/8K	k token/s	Output error
CH-30	bf16	0.005	24.62 ± 0.20	83 ± 4	$54 \pm 5/28 \pm 5$	226 ± 4	1.6×10^{-2}
CH-100	bf16	0.005	24.52 ± 0.16	85 ± 3	$57 \pm 4/32 \pm 4$	141 ± 3	5.0×10^{-3}
CH-300	bf16	0.005	24.52 ± 0.16	85 ± 3	$57 \pm 4/32 \pm 4$	66 ± 2	5.0×10^{-3}
CH-100	fp32	0.005	24.35 ± 0.15	88 ± 3	$60 \pm 4/35 \pm 4$	50 ± 1	5×10^{-5}
CG-30	bf16	0.005	24.7 ± 0.5	80 ± 8	$52 \pm 8/27 \pm 7$	214 ± 4	n/a
Direct chunk-ridge	fp32 state	0.005	24.33 ± 0.14	89 ± 2	$61 \pm 4/36 \pm 4$	44 ± 1	5×10^{-5}
CH-30	bf16	0.002	24.8 ± 0.3	76 ± 6	$48 \pm 6/24 \pm 6$	226 ± 4	7×10^{-2}
CH-100	bf16	0.002	24.55 ± 0.20	84 ± 4	$56 \pm 5/31 \pm 5$	141 ± 3	5.0×10^{-3}
CH-100	fp32	0.002	24.36 ± 0.18	89 ± 3	$61 \pm 5/36 \pm 5$	50 ± 1	4×10^{-3}
Direct chunk-ridge	fp32 state	0.002	24.31 ± 0.16	90 ± 2	$62 \pm 4/37 \pm 4$	42 ± 1	8×10^{-5}

Table 25: Mechanism diagnostics for the paired GKA checkpoints, with 2,048 validation positions per head and three seeds. Error rows use Table 24's relative output error; input precision is explicit.

	$a = 0.02$	0.01	0.005	0.002
κ bound $(a+1)/a$	51	101	201	501
Median $\kappa(H + \lambda I)$	38 ± 3	74 ± 6	146 ± 12	360 ± 30
CH-30 bf16, bf16 inputs	5×10^{-3}	5×10^{-3}	1.6×10^{-2}	7×10^{-2}
CH-100 bf16, bf16 inputs	5×10^{-3}	5×10^{-3}	5×10^{-3}	5×10^{-3}
CH-100 fp32, bf16 inputs	5×10^{-5}	5×10^{-5}	5×10^{-5}	4×10^{-3}
Direct chunk-ridge, bf16 inputs	5×10^{-5}	5×10^{-5}	5×10^{-5}	8×10^{-5}
CH-100 fp32, fp32 inputs	2×10^{-7}	3×10^{-6}	1×10^{-5}	4×10^{-3}
Direct chunk-ridge, fp32 inputs	1×10^{-6}	3×10^{-6}	1×10^{-5}	4×10^{-5}
Guard rebuilds / 10^6 chunks	0	0	2.7 ± 1.5	38 ± 9

Full paired summary and ridge sweep. Table 26 includes the additional iteration and precision settings; Figure 2 gives the complete reported ridge sweep.

Table 26: Paired Gated KalmaNet results. Gap is MQAR points below FlashRLS at the same ridge a. Guard rebuilds the state to correct roundoff. The FlashRLS rows use different a. Throughput is thousands of tokens per second.

Read	a	Slim PPL	MQAR (%)	Gap to FlashRLS	k token/s
CH-30, bf16	0.01	24.48 ± 0.15	84 ± 3	2.1 ± 1.3	226 ± 4
FlashRLS, exact	0.01	24.41 ± 0.14	86 ± 3	0	45 ± 1
CH-30, bf16	0.005	24.62 ± 0.20	83 ± 4	6.4 ± 1.2	226 ± 4
CH-100, bf16	0.005	24.52 ± 0.16	85 ± 3	4.2 ± 1.4	141 ± 3
CH-100, fp32	0.005	24.35 ± 0.15	88 ± 3	0.8 ± 1.2	50 ± 1
FlashRLS, exact + guard	0.005	24.33 ± 0.14	89 ± 2	0	44 ± 1

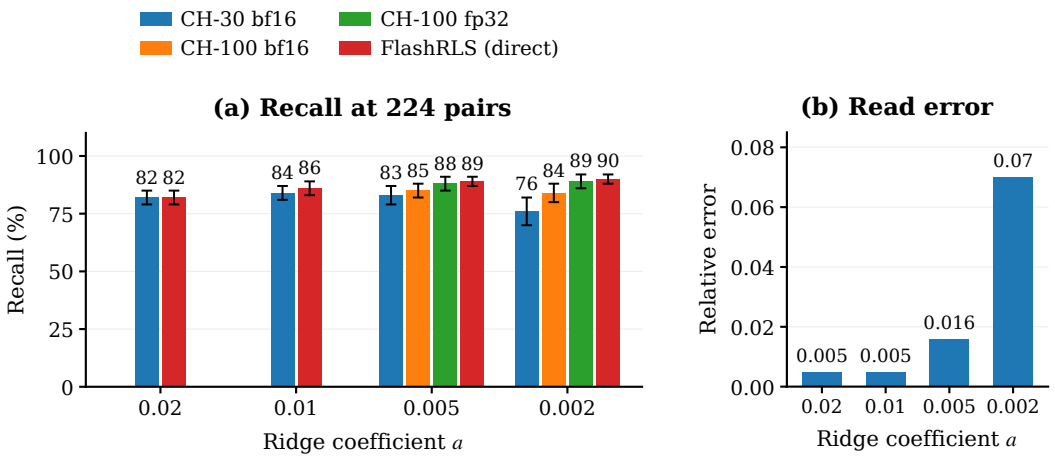

Figure 2: The 50M Gated KalmaNet ridge sweep; ridge weakens from left to right. (a) MQAR recall at 224 pairs, with means and standard deviations over three seeds. (b) CH-30 bf16 relative read error against fp64 for identical inputs. Bars show the reported settings; both axes start at zero.

L.6 Learned nonlinear learner and adaptive-adjoint verification

The local rule and the backward arms. Protocol B is the paired backward-pass comparison. Its per-token local solve selects an information weight w_t and a pseudo-response $\widetilde{v}_t$; for a smooth local loss with nonzero curvature, $\widetilde{v}_t = z_t - \ell'(z_t)/\ell''(z_t)$ is the IRLS working target of the local quadratic surrogate. The logistic information weight is $\sigma(z_t)[1-\sigma(z_t)]$. The Huber experiment uses the residual-dependent reweighting rule

$$w_t = \begin{cases} 1, & |r_t| \le \delta, \\ \delta/|r_t|, & |r_t| > \delta, \end{cases} \qquad \delta = 0.5.$$

Here w_t denotes the information weight used by the weighted normal equations. The adaptive adjoint differentiates this rule, the pseudo-response, the gates, and the local scalar solution. The forward computation is identical across the nonlinear backward arms at fixed model parameters. Training uses three paired initializations and data orders.

Arm (i) uses the complete adaptive adjoint. Arm (ii) sets the derivatives through w_t and the adaptive pseudo-response corrections to zero and differentiates the remaining solve. Arm (iii) keeps the exact forward computation and uses a 30-step CG implicit adjoint. Arm (iv) freezes the memory solve; the projection parameters receive the direct read-path gradients. Quadratic exact and stopped-gradient arms provide a separate null control. The learning-rate scan is performed on arm (i) and its selection is held fixed for the other arms.

Three-seed study. Table 27 groups only the three-seed B1/B2 measurements, including B1 step latency. The ten-seed summaries in Table 1 are reported separately; their table does not reuse these timing measurements.

L.6.1 B0: verification protocol

The verification grid has $B = n_{\mathrm{h}} = 2$, $d \in \{8, 32\}$, $C \in \{4, 16\}$, $N = 64$, ridge in $\{1, 10^{-2}\}$, and $\kappa_2(H_0) \in \{10^2, 10^6\}$, with three input seeds. Logistic cases include saturated logits and decision-boundary observations. Huber cases place residuals at and around the threshold. The fp64 sequential reference differentiates Newton iterations unrolled to convergence. Table 28 states the acceptance criteria for the ten checks.

The fp64 identity checks remain near round-off on the small verification shapes. B0.2, B0.3, B0.5, B0.9, and B0.10 reach approximately 10^{-14} to 10^{-13} in the well-conditioned cells and rise to at most 5×10^{-11} in the $\kappa = 10^6$ Huber cell. B0.1 and B0.4 are finite-difference comparisons and

Table 27: Three-seed adaptive-backward study (Protocols B1 and B2). MQAR reports steps to 99% training accuracy, recall at step 3000, and time per training step. Huber reports outlier MSE at context length 256. NR means not reached.

	MQAR			Huber regression
Method	Steps ↓	Recall (%) ↑	ms/step ↓	MSE ↓
FlashRLS (exact backward)	$1{,}130 \pm 90$	89.5 ± 1.5	214 ± 3	0.056 ± 0.004
Stopped curvature	$2{,}050 \pm 180$	80.3 ± 2.2	171 ± 3	0.084 ± 0.008
CG-30 implicit	$2{,}800 \pm 600$	72.0 ± 7.0	189 ± 3	0.110 ± 0.040
Frozen memory	NR	52.0 ± 6.0	149 ± 2	0.150 ± 0.030
Squared loss, exact	–	–	–	0.170 ± 0.020
Batch Huber, offline	–	–	–	0.051

Table 28: B0 adaptive-adjoint verification protocol. Numerical entries are acceptance tolerances. The central-difference step is 10^{-6}, and the dot-product test uses 16 random probe pairs.

Check	Comparison	Acceptance criterion
B0.1	Adjoint and central differences for k, q, v, gates, and ridge	Relative error $\leq 10^{-7}$ per input
B0.2	Adjoint and autograd through converged fp64 sequential recursion	Relative error $\leq 10^{-10}$
B0.3	$\langle Jv, u\rangle = \langle v, J^\top u\rangle$, fp64	Relative error $\leq 10^{-12}$
B0.4	Implicit scalar derivatives and differences of the converged local solve	Error $\leq 10^{-8}$; local residual $\leq 10^{-12}$
B0.5	Gradients at $C = 1, 4, 16$	Relative error $\leq 10^{-11}$
B0.6	CUDA/Triton, fp32 state and fp64 core, against B0.2	$\leq 3\times10^{-6}$ at 10^2; $\leq 3\times10^{-4}$ at 10^6
B0.7	Constructed positive pivots approaching zero; rejected nonpositive pivots	Finite reference agreement; assertion at $1 + w_t s_t \leq 0$
B0.8	Huber threshold and $\delta \pm 10^{-9}$	Agreement with the selected one-sided derivative
B0.9	Stopped arm and adaptive adjoint with local adaptive terms removed	Relative error $\leq 10^{-12}$
B0.10	Exact and stopped-gradient quadratic-memory adjoints	Relative error $\leq 10^{-12}$

remain below 10^{-8}. The scalar logistic solve converges in 3.1–3.2 Newton iterations on average, with damping on less than one percent of tokens.

B0.6 measures the device realization. With fp32 inputs and state, the worst relative adjoint error is 1.5×10^{-6} at $\kappa = 10^2$ and 9×10^{-5} at 10^6. Bfloat16 inputs raise this floor to 7×10^{-5} and 5×10^{-4}. The conditioning dependence follows the same precision envelope as the forward solve. Constructed positive pivots remain finite and match the reference; nonpositive pivots trigger the declared assertion. All ten checks satisfy their acceptance criteria before B1 and B2 are run.

L.6.2 B1: LOGISTIC-MEMORY ASSOCIATIVE RECALL

The task uses the MQAR generator in this appendix: two layers of width 128, two 64-dimensional heads, and 256-token training sequences with 32 stored pairs. The memory emits value logits and uses a multinomial logistic response with per-class scalar information weights. Evaluation freezes the trained weights and increases the store to 128, 224, and 256 pairs. We report steps to 99% training accuracy, final loss, and answer-position exact match. Gradient comparisons use exact-arm checkpoints at 25%, 50%, and 100% of training, with group cosines tabulated at 25% and 100%.

The reported paired summaries are 1.81 ± 0.02 for the step-count ratio and 9.2 ± 0.8 points for MQAR. With the measured mean step times, the seedwise estimated times to criterion are 222.56/239.68/263.22 seconds for exact and 321.48/345.42/384.75 seconds for stopped. These products give a mean time ratio of approximately 1.45; they combine measured step counts with each

Table 29: Measured worst-case B0 values for logistic rules. "Worst" is the maximum over inputs, seeds, and forced special tokens. Quantity labels include the acceptance tolerances from Table 28 where useful; Huber and quadratic checks are in Table 30.

Check	Quantity	Logistic $\kappa = 10^2$	Logistic $\kappa = 10^6$
B0.1	Finite-difference rel. error (10^{-7})	3×10^{-9}	8×10^{-9}
B0.2	Unrolled-autograd rel. error (10^{-10})	2×10^{-12}	4×10^{-11}
B0.3	Dot-product rel. error (10^{-12})	3×10^{-14}	2×10^{-13}
B0.4	Scalar-sensitivity rel. error (10^{-8})	5×10^{-10}	5×10^{-10}
B0.4	Newton residual (10^{-12})	3×10^{-15}	3×10^{-15}
B0.4	Newton iterations, mean / max	3.1 / 6	3.2 / 7
B0.4	Damping-event fraction	0.4%	0.6%
B0.5	Chunk-invariance rel. error (10^{-11})	6×10^{-14}	2×10^{-12}
B0.6	Kernel vs fp64, fp32 inputs	1.5×10^{-6}	9×10^{-5}
B0.6	Kernel vs fp64, bf16 inputs	7×10^{-5}	5×10^{-4}
B0.7	$1 + w_t s_t = 10^{-6}$: gradient rel. error	2×10^{-8}	3×10^{-8}
B0.7	PD assertion at $1 + w_t s_t \leq 0$	yes	yes
B0.8	Huber threshold convention	n/a	n/a
B0.9	Stopped-gradient identity (10^{-12})	4×10^{-14}	3×10^{-13}
B0.10	Quadratic exact/stopped null (10^{-12})	n/a	n/a

Table 30: Measured worst-case B0 values for the Huber and quadratic controls. The checks and tolerances are the same as in Table 29. Slash-separated conditioning results follow the original $10^2/10^6$ order; "mean / max" labels refer to Newton iterations.

Check	Quantity	Huber $10^2/10^6$	Quadratic null
B0.1	Finite-difference rel. error (10^{-7})	$4 \times 10^{-9}/9 \times 10^{-9}$	2×10^{-9}
B0.2	Unrolled-autograd rel. error (10^{-10})	$2 \times 10^{-12}/5 \times 10^{-11}$	1×10^{-12}
B0.3	Dot-product rel. error (10^{-12})	$3 \times 10^{-14}/2 \times 10^{-13}$	2×10^{-14}
B0.4	Scalar-sensitivity rel. error (10^{-8})	$6 \times 10^{-10}/6 \times 10^{-10}$	exact
B0.4	Newton residual (10^{-12})	0	n/a
B0.4	Newton iterations, mean / max	1 / 1	n/a
B0.4	Damping-event fraction	0	n/a
B0.5	Chunk-invariance rel. error (10^{-11})	$6 \times 10^{-14}/2 \times 10^{-12}$	5×10^{-14}
B0.6	Kernel vs fp64, fp32 inputs	$1.4 \times 10^{-6}/8 \times 10^{-5}$	$1.2 \times 10^{-6}/7 \times 10^{-5}$
B0.6	Kernel vs fp64, bf16 inputs	$7 \times 10^{-5}/5 \times 10^{-4}$	$6 \times 10^{-5}/4 \times 10^{-4}$
B0.7	$1 + w_t s_t = 10^{-6}$: gradient rel. error	$2 \times 10^{-8}/3 \times 10^{-8}$	n/a
B0.7	PD assertion at $1 + w_t s_t \leq 0$	yes	n/a
B0.8	Huber threshold convention	pass / pass	n/a
B0.9	Stopped-gradient identity (10^{-12})	$4 \times 10^{-14}/3 \times 10^{-13}$	4×10^{-14}
B0.10	Quadratic exact/stopped null (10^{-12})	n/a	2×10^{-14}

arm's mean step latency. The quadratic exact/stopped null has maximum loss difference 2×10^{-7} and zero accuracy difference across the three seeds.

L.6.3 B2: ROBUST KERNEL MEMORY

Inputs and GP-RBF targets follow the nonlinear in-context regression task in Appendix M: $x \in [-2, 2]^8$, RBF lengthscale one, and KRR-nn kernel memory. Observations use Student-t_3 noise scaled to $\sigma = 0.1$, with 10% magnitude-three outliers. The Huber threshold 0.5 is tuned on a held-out split. A Gaussian-noise control removes the outlier contamination. Test MSE uses 64 and 256 context points. References are the Gaussian-model GP posterior and per-sequence batch Huber fit.

Table 31: B1 training and zero-shot recall. Mean and standard deviation over three seeds. Recall is evaluated at step 3,000 for every arm. Exact and stopped reach 99% by steps 1230 and 2250 on all seeds. CG-30 reaches the criterion in two runs. NR marks the unconverged run. *This seed is evaluated at step 3,000 and included in the mean.

Backward	Steps to 99%	128 pairs	224 pairs	256 pairs	Final loss
Exact adaptive	$1{,}130 \pm 90$	99.4 ± 0.3	89.5 ± 1.5	82 ± 2	0.019 ± 0.003
Stopped curvature	$2{,}050 \pm 180$ $2{,}800 \pm 600$	98.1 ± 0.6	80.3 ± 2.2 72 ± 7	70 ± 3	0.041 ± 0.006
CG-30 implicit	(2,300/2,700/NR)	96 ± 2	(78/74/64*)	60 ± 8	0.09 ± 0.05
Frozen memory	0/3 reached	89 ± 4	52 ± 6	41 ± 6	0.33 ± 0.09
Quadratic, exact	$1{,}010 \pm 80$	98.9 ± 0.4	74.6 ± 2.6	62 ± 3	0.018 ± 0.003
Quadratic, stopped	$1{,}010 \pm 80$	98.9 ± 0.4	74.6 ± 2.6	62 ± 3	0.018 ± 0.003

Table 32: B1 paired values, in seed order. Ratios divide stopped-gradient steps by exact-adjoint steps. Accuracy differences are exact minus stopped in percentage points.

Quantity	Seed 1	Seed 2	Seed 3
Exact steps	1,040	1,120	1,230
Stopped steps	1,880	2,020	2,250
Step-count ratio	1.81	1.80	1.83
Exact MQAR at 224	91.1	88.0	89.4
Stopped MQAR at 224	82.6	78.0	80.3
Paired accuracy difference	8.5	10.0	9.1
Exact final loss	0.017	0.022	0.018
Stopped final loss	0.036	0.047	0.040

At 256 points the paired exact outlier MSEs are $0.052, 0.060, 0.057$, and the stopped values are $0.078, 0.091, 0.083$. Their stopped/exact ratios are $1.50, 1.52, 1.46$ with reported mean 1.49 ± 0.03. The recovered reference gap is $(e_{\text{stopped}} - e_{\text{exact}})/(e_{\text{stopped}} - e_{\text{batch}})$. Using the unrounded mean of these seed values and the batch reference 0.051 gives approximately 84%. The Gaussian exact and stopped rows are closely matched, with a 0.001 difference in their tabulated mean MSE.

L.6.4 Curvature diagnostics and step cost

We measure $\mathrm{CV}^2(w) = \mathrm{Var}_t(w_t)/\mathbb{E}_t[w_t]^2$ on exact-arm checkpoints. Table 35 reports its observed association with the exact/stopped difference. The quadratic weight is constant and its diagnostic is zero. The contamination sweep varies the frequency of robust reweighting.

Protocol B accounts for approximately eight H100 GPU-hours, including gradient-fidelity evaluation. The paired 50M Gated KalmaNet study and Protocol B together account for approximately 184 H100 GPU-hours, including their evaluations.

Table 33: Gradient cosine on exact-arm checkpoints, by parameter group: keys, queries, values, and gates. Each comparison evaluates a different backward on the same checkpoint.

Backward	25% of training: $k/q/v/g$	100%: $k/q/v/g$
Exact adaptive	1 / 1 / 1 / 1	1 / 1 / 1 / 1
Stopped curvature	0.71 / 0.66 / 0.93 / 0.58	0.88 / 0.85 / 0.97 / 0.80
CG-30 implicit	0.55 / 0.50 / 0.80 / 0.40	0.70 / 0.65 / 0.88 / 0.55
Frozen memory	0.32 / 0.30 / 0.55 / n/a	0.45 / 0.42 / 0.62 / n/a
Quadratic stopped	1.000 ± 10^{-6}	1.000 ± 10^{-6}

Table 34: B2 test MSE, mean and standard deviation over three seeds. The Gaussian control and contaminated distribution are evaluated separately.

	Outlier distribution		Gaussian control	
Memory / backward	$t = 64$	$t = 256$	$t = 64$	$t = 256$
Gaussian GP reference	0.41	0.190	0.12	0.030
Batch Huber reference	0.16	0.051	0.13	0.034
Quadratic, exact	0.38 ± 0.03	0.17 ± 0.02	0.14 ± 0.01	0.040 ± 0.004
Huber, exact adaptive	0.18 ± 0.01	0.056 ± 0.004	0.15 ± 0.01	0.042 ± 0.004
Huber, stopped curvature	0.26 ± 0.02	0.084 ± 0.008	0.15 ± 0.01	0.043 ± 0.004
Huber, CG-30 implicit	0.31 ± 0.06	0.11 ± 0.04	0.16 ± 0.02	0.047 ± 0.009
Huber, frozen	0.35 ± 0.04	0.15 ± 0.03	0.20 ± 0.02	0.070 ± 0.010

Table 35: Curvature variation and adaptive-gradient fidelity. Task differences are stopped minus exact; accuracy uses percentage points.

Condition	$\mathrm{CV}^2(w)$	Task difference	Gradient cosine
B1, 25% training	1.9 ± 0.2	Loss $+0.21 \pm 0.04$	0.68 ± 0.05
B1, final	0.6 ± 0.1	Loss $+0.022 \pm 0.005$	0.87 ± 0.03
B1, final recall	0.6 ± 0.1	MQAR -9.2 ± 0.8	0.87 ± 0.03
B2, Gaussian control	0.02 ± 0.01	MSE $+0.001 \pm 0.003$	0.995 ± 0.003
B2, 5% outliers	0.45 ± 0.05	MSE $+0.012 \pm 0.004$	0.90 ± 0.02
B2, 10% outliers	0.9 ± 0.1	MSE $+0.028 \pm 0.006$	0.80 ± 0.03
B2, 20% outliers	1.6 ± 0.2	MSE $+0.051 \pm 0.010$	0.68 ± 0.05
Quadratic memory	0	0	1.000

Table 36: Training-step cost on H100, mean and standard deviation over 200 steps. Backward share is the fraction spent in the memory backward.

Backward	B1 ms/step	Backward share	B2 ms/step	Backward share
Exact adaptive	214 ± 3	38%	521 ± 6	44%
Stopped curvature	171 ± 3	23%	432 ± 5	32%
CG-30 implicit	189 ± 3	30%	468 ± 6	38%
Frozen memory	149 ± 2	12%	381 ± 5	22%
Quadratic, exact	181 ± 3	27%	409 ± 5	30%

M ONLINE ESTIMATION AND NONLINEAR PREDICTION

M.1 ACOUSTIC ECHO CANCELLATION

AEC uses the Microsoft AEC Challenge synthetic fold (Sridhar et al., 2021), 16 kHz audio, ten-second clips, and pre-update prediction from the last d far-end samples. The linear study uses 200 clips and the nonlinear-loudspeaker collection has 100 clips. The comparison in Table 6 uses the linear collection, $d = 4{,}096$, ridge one, and fading $\gamma = 0.9998$. An oracle double-talk gate controls assimilation. Fading is frozen during a closed gate. Steady ERLE averages segmental echo reduction over the final two seconds. The sample standard deviation is computed across three disjoint clip splits. The FDKF baseline is the frequency-domain adaptive Kalman filter of Enzner & Vary (2006).

Ridge maintenance. For each chunk, the recurrence also processes C_p zero-response pseudo-observations with keys $\sqrt{\epsilon_j}e_j$ and unit weight, cycling through the d coordinates. The increment

$$\epsilon_j = \lambda \left(1 - \gamma^{(d/C_p)C}\right)$$

restores the ridge contribution lost over one coordinate cycle. These pseudo-positions have no prediction output and are excluded from task metrics. Sequential and block implementations receive the same observations and pseudo-observations. This estimator operation uses fp32 arithmetic.

Numerical policies. Free-running executions never restore the state from the fp64 reference. Independent fp64 fading normal equations use the same pseudo-observations to evaluate drift every 0.5 seconds. The 200 clips give 4,000 diagnostic windows. A failed window contains a non-finite state. The guarded execution additionally checks the projected residual at every chunk and reconstructs from fp64 statistics when the residual exceeds 5×10^{-4}. The free-running and guarded columns refer to separate complete trajectories. Table 6 reports prediction quality, failure counts, and elapsed compute; Table 37 reports drift and rebuilds. The guard uses the projected residual. Inverse drift is

Table 37: AEC drift and recovery under the pseudo-observation ridge protocol. Relative inverse drift against the independent fp64 reference. Values with $\pm$ use three clip splits.

Method	Policy	Median drift	Worst drift	Rebuilds
FlashRLS, core64	Free	$(2 \pm 1) \times 10^{-5}$	$(2 \pm 1) \times 10^{-3}$	0
FlashRLS, core64	Guarded	$(1 \pm 0.5) \times 10^{-5}$	$\leq 5 \times 10^{-4}$	15 ± 8
FlashRLS, core32	Free	$(2 \pm 1) \times 10^{-5}$	$(1 \pm 0.5) \times 10^{-2}$	0
Symmetrized SM	Free	$(1.2 \pm 0.4) \times 10^{-4}$	$(4 \pm 2) \times 10^{-3}$	0
Joseph form	Free	$(9 \pm 3) \times 10^{-5}$	$(2 \pm 1) \times 10^{-3}$	0

measured directly against the fp64 reference. Guard bookkeeping and pseudo-observation processing are included in application times.

M.2 STREAMING CLICK PREDICTION

Criteo Display Advertising (Criteo Labs, 2014) has 13 numeric and 26 categorical fields and 45.8M rows. Features are hashed to $d = 4{,}096$, with numeric fields log-bucketed. Every method makes a logistic prediction and then updates once in file order. Three hashing seeds define the feature realizations. FTRL-Proximal (McMahan et al., 2013) parameters $(\alpha, \beta, \lambda_1, \lambda_2)$ are tuned on the first million rows. Adagrad (Duchi et al., 2011) uses first-order updates. The sequential online-Newton and FlashRLS-GLM implementations use the experiment's matched local update, with $\gamma = 1 - 10^{-6}$ and $C = 64$ for FlashRLS. Its innovation core is fp64 and its recurrent state is fp32.

Progressive logloss and AUC are evaluated on the final ten million rows. Table 3 contains the online comparisons. A whole-stream batch IRLS fit reaches logloss 0.4610 ± 0.0005 and AUC 0.7910 ± 0.0007. Sequential fp32 online Newton loses positive definiteness by eight million rows, with measured cost 350 seconds per million rows. The fp64 reference and FlashRLS reach the same mean logloss and AUC; their times are 620 and 16 seconds per million rows.

M.3 CLASSIFICATION UNDER CONCEPT DRIFT

Covertype (581k rows, 54 features) (Blackard, 1998), Electricity (45k) (Harries, 1999), and Airlines (539k) (American Statistical Association, 2009; Bifet et al., 2010) follow their standard stream order. An RBF random Fourier feature map (Rahimi & Recht, 2007) has width 2,048; bandwidth uses the median heuristic on the first 5,000 rows. We use three feature seeds and prequential accuracy over the full stream. The fixed-fading comparison uses $\gamma = 0.9995$. The adaptive version chooses per-token decay with ADWIN (Bifet & Gavaldà, 2007). Every observation is predicted before its label is assimilated. The fp32 sequential fixed-fading implementation has non-finite states on two

Table 38: Streaming classification with concept drift. Accuracy over three feature seeds; wall-clock time is for Covertype.

Method	Covertype	Electricity	Airlines	Time (s)
FTRL-Proximal	0.784 ± 0.004	0.812 ± 0.006	0.652 ± 0.004	6
Adagrad-SGD	0.791 ± 0.004	0.821 ± 0.006	0.655 ± 0.004	5
ONS fp64, no forgetting	0.836 ± 0.003	0.851 ± 0.005	0.668 ± 0.003	190
ONS fp64, fixed fading	0.871 ± 0.003	0.882 ± 0.005	0.681 ± 0.003	190
FlashRLS, fixed fading	0.871 ± 0.003	0.882 ± 0.005	0.681 ± 0.003	6.5
FlashRLS, ADWIN fading	0.884 ± 0.003	0.889 ± 0.005	0.686 ± 0.003	7

of three Covertype seeds and fails on Airlines; Electricity accuracy is 0.874 ± 0.006. Its Covertype time is 110 seconds. The fixed-fading FlashRLS comparison changes execution and matches the sequential fp64 prediction accuracy. ADWIN changes the forgetting rule and improves the three accuracies.

M.4 NONLINEAR IN-CONTEXT REGRESSION

Inputs are in $[-2, 2]^8$. Functions are drawn from a Gaussian process with RBF lengthscale one, with observation noise standard deviation 0.1. The training collection contains 200,000 sequences of length 256; the test collection has 1,024 sequences, evaluated after 16, 64, 256, and 1024 observations. Three seeds determine the trained models. The Transformer baseline (Vaswani et al., 2017) has eight layers of width 256. The kernel memory uses RBF keys, 64 landmarks, and an exact global tier. Table 39 gives the results with more observations. The additional error at 16 observations and

Table 39: Nonlinear in-context regression. Test MSE, mean $\pm$ standard deviation over three seeds. Training length is 256.

Model	$t = 64$	$t = 256$	$t = 1{,}024$
GP posterior mean	0.120	0.030	0.012
Transformer	0.15 ± 0.01	0.045 ± 0.004	0.90 ± 0.30
Gated DeltaNet	0.42 ± 0.03	0.38 ± 0.03	0.40 ± 0.04
MesaNet, linear keys	0.40 ± 0.03	0.37 ± 0.03	0.37 ± 0.03
FlashRLS, linear keys	0.40 ± 0.03	0.37 ± 0.03	0.37 ± 0.03
KRR-nn, exact tier	0.14 ± 0.01	0.040 ± 0.004	0.022 ± 0.004

throughput measurements appear below.

M.5 SINGLE-PASS ONLINE KERNEL CLASSIFICATION

MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky, 2009) are streamed once in three random orders. Random Fourier features (Rahimi & Recht, 2007) of raw pixels have width 4096. CIFAR-10 is also evaluated with frozen ResNet-18 (He et al., 2016) penultimate features of width 512. Models use multinomial logistic loss with per-class scalar curvature. Prequential accuracy uses the final 20% of the stream.

Table 40: Early prediction error and execution in nonlinear in-context regression. MSE at 16 observations; throughput relative to Gated DeltaNet.

Model	MSE at $t = 16$	Relative token/s
GP posterior mean	0.31	n/a
Transformer	0.36 ± 0.02	0.80
Gated DeltaNet	0.55 ± 0.03	1.00
MesaNet, linear keys	0.52 ± 0.03	0.65
FlashRLS, linear keys	0.52 ± 0.03	0.25
KRR-nn, exact tier	0.34 ± 0.02	0.10

Table 41: Online kernel classification. Accuracy, mean $\pm$ standard deviation over three stream orders.

Method	MNIST RFF 4,096	CIFAR-10 RFF 4,096	CIFAR-10 ResNet-18 512
Adagrad-SGD	0.966 ± 0.002	0.512 ± 0.006	0.862 ± 0.004
FTRL-Proximal	0.963 ± 0.002	0.505 ± 0.006	0.858 ± 0.004
Sequential ONS fp64	0.979 ± 0.001	0.571 ± 0.005	0.884 ± 0.003
FlashRLS-GLM	0.979 ± 0.001	0.571 ± 0.005	0.884 ± 0.003
Batch kernel logistic	0.983 ± 0.001	0.585 ± 0.004	0.890 ± 0.003

M.6 Chaotic dynamics with online readouts

Lorenz–96 (Lorenz, 1996) uses 40 variables and forcing 8. Kuramoto–Sivashinsky (Kuramoto & Tsuzuki, 1976; Sivashinsky, 1977) uses domain length 22 and 64 grid points. Training spans 1,000 Lyapunov times, with time steps 0.01 and 0.25 respectively. Features come from a fixed echo-state reservoir (Jaeger, 2001) of width 4,096 or random Fourier features (Rahimi & Recht, 2007) of a five-step window. RLS uses fading 0.99995 and chunks of 64. Autoregressive forecasts start from 100 held-out initial conditions. Valid prediction time is measured in Lyapunov times until normalized error exceeds 0.4; variation covers the rollouts and three reservoir seeds.

The fp32 sequential implementation diverges near condition number 10^7 and has measured fit time 900 seconds. Batch ridge uses the whole training set. The RLS implementations assimilate observations sequentially. FlashRLS and the fp64 sequential implementation have the same forecasting quality with a $53.3\times$ fit-time difference.

The Lorenz–96 and Kuramoto–Sivashinsky experiments fit 4,096-dimensional readouts to 10^5 observations. FlashRLS matches sequential fp64 RLS forecast quality at 30 versus 1,600 s, a $53.3\times$ fitting speedup. Batch ridge takes 2 s with slightly higher mean forecast quality for a final fit using all training data; RLS retains intermediate predictors after each update. With teacher signals during rollout, FlashRLS reaches 5.1 ± 0.5 and 6.4 ± 0.6 Lyapunov times; batch ridge gives 4.3 ± 0.4 and 5.6 ± 0.5. Table 7 and Appendix M.6 distinguish these update schedules.

M.7 Generalized-linear bandits

The logistic bandit has 50 arms, 64-dimensional contexts, Bernoulli rewards with probability $\mathrm{sigmoid}(\theta^\top x)$, and a horizon of 50,000 decisions. Twenty runs compare GLM-UCB (Filippi et al., 2010) with SGD, sequential online Newton, and FlashRLS-GLM. The policy receives the same estimator interface in each case. Two fp32 sequential runs fail. FlashRLS matches the fp64 reference's

Table 42: Logistic bandit regret after 50,000 decisions. Mean $\pm$ standard deviation over 20 runs.

Estimator inside GLM-UCB	Regret	Time (s)
SGD	$1{,}450 \pm 120$	9
Sequential ONS fp32	$1{,}040 \pm 140$	210
Sequential ONS fp64	880 ± 70	380
FlashRLS-GLM	880 ± 70	11

regret and reduces runtime by $34.5\times$.

M.8 FIXED-FEATURE RESPONSE CONTROLS

HIGGS (Baldi et al., 2014) uses fixed 64-dimensional tanh features and five seeds. Recursive logistic Laplace improves AUROC by 0.0142 ± 0.0012 and reduces expected calibration error by 0.00967 ± 0.00483 relative to the fixed-curvature logistic control. Beijing forecasting (Zhang et al., 2017) evaluates one-hour-ahead PM2.5 at 12 stations. On tanh features at fading 0.998, Student-t reduces MAE by 0.414 ± 0.019 relative to squared updates. Persistence gives the lowest error on that feature setting. On the provided covariates, pseudo-Huber and persistence have RMSE 19.862 and 20.457. These controls fix features and order to isolate the local response rule.

N SQUARE-ROOT REALIZATIONS AND MEASURED QRD EXECUTION

This appendix derives information-factor and covariance-factor realizations of the sequential update operator. Both represent the same physical state actions with triangular factors.

N.1 QRD-RLS AS A FACTOR-STATE REALIZATION

Let $H_0 = R_0^\top R_0$ with upper-triangular R_0 and positive diagonal. Set $D_0 = R_0 W_0^\top$. The objective after c observations in Equation (12) is

$$\frac{1}{2}\left\| \begin{bmatrix} R_0 \\ K_{1:c} \end{bmatrix} W^\top - \begin{bmatrix} D_0 \\ Y_{1:c} \end{bmatrix} \right\|_F^2 .$$

Apply the same orthogonal transformation to the design and response arrays:

$$\mathcal{U}_c^\top \begin{bmatrix} R_0 \\ K_{1:c} \end{bmatrix} = \begin{bmatrix} R_c \\ 0 \end{bmatrix}, \qquad \mathcal{U}_c^\top \begin{bmatrix} D_0 \\ Y_{1:c} \end{bmatrix} = \begin{bmatrix} D_c \\ J_c \end{bmatrix}. \tag{134}$$

Orthogonality preserves the objective, which becomes $\|R_c W^\top - D_c\|_F^2/2 + \|J_c\|_F^2/2$. Thus

$$H_c = R_c^\top R_c, \qquad W_c^\top = R_c^{-1} D_c, \qquad P_c q = R_c^{-1}(R_c^{-\top} q). \tag{135}$$

These equalities hold after every update. Chronological row insertion maintains (R_c, D_c) and permits a read before or after each insertion. This is the classical QRD route to the specified regression problem (Bierman, 1977; Hsieh et al., 1993; Sayed & Kailath, 1994). A terminal QR factor contains the terminal objective; obtaining all intermediate reads requires the corresponding chronological factors or an equivalent representation of the intermediate states.

A structured row update of a triangular factor uses $O(d^2)$ arithmetic, and transforming m response columns uses $O(dm)$ per inserted row. Block Householder organization exposes matrix–matrix work (Bojanczyk et al., 1993). The token QRD implementation uses chronological Givens updates; the block implementation uses Householder QR of the stacked design and solves for reads at each update.

N.2 THE FLASHRLS CORRECTIONS INSIDE A SQUARE-ROOT COVARIANCE ARRAY

The normalized write rows can also be read directly from a square-root Kalman array. This makes the connection to the innovation representation explicit. Let $P_0 = F_0 F_0^\top$, with nonsingular lower-triangular F_0, and define

$$\mathcal{A} = \begin{bmatrix} I_C & KF_0 \\ 0 & F_0 \end{bmatrix}, \qquad \mathcal{A}^\top = \mathcal{Q}\mathcal{R}, \qquad \mathcal{R}^\top = \begin{bmatrix} L & 0 \\ J & F_+ \end{bmatrix}. \tag{136}$$

Choose positive diagonal entries in the QR factor. The QR operates on the array transpose; its transposed triangular factor is lower triangular. Array triangularization is a standard construction in square-root filtering (Bierman, 1977; Tracy, 2022; Yaghoobi et al., 2025).

Proposition N.1 (Square-root realization of the sequential corrections)**.** *The blocks of* (136) *satisfy*

$$LL^\top = I_C + KP_0K^\top, \qquad J^\top = L^{-1}KP_0 = X, \qquad F_+F_+^\top = P_C. \tag{137}$$

Solving $LE = Y - KW_0^\top$ therefore gives exactly the normalized residuals in Corollary 3.5. The same masked read formula and $W_C = W_0 + E^\top X$ apply.

Proof. Orthogonality gives $\mathcal{A}\mathcal{A}^\top = \mathcal{R}^\top\mathcal{R}$. Comparing blocks yields

$$\begin{bmatrix} I_C + KP_0K^\top & KP_0 \\ P_0K^\top & P_0 \end{bmatrix} = \begin{bmatrix} LL^\top & LJ^\top \\ JL^\top & JJ^\top + F_+F_+^\top \end{bmatrix}.$$

The leading equality identifies L as the positive-diagonal Cholesky factor of S. The cross block gives $J^\top = L^{-1}KP_0 = X$. The lower-right equality gives $F_+F_+^\top = P_0 - X^\top X = P_C$. The residual solve uses the same L, so Corollary 3.5 supplies every intermediate read and the terminal predictor. □

This construction carries F_+ as the next covariance square root. For any nonsingular computed factor $\widehat{F}_+$, its represented covariance $\widehat{F}_+\widehat{F}_+^\top$ is positive definite in exact arithmetic. The implementation can retain this factor directly, with inverse-state actions evaluated through factor products. The theorem gives a compatible factor-state realization of FlashRLS. The FlashRLS block-Woodbury kernel maintains P explicitly; the measured QRD baseline carries an information factor (Appendix N).

Positive response-dependent information weights can use chronological QR updates with rows $\sqrt{w_t}k_t^\top$ after each local rule supplies w_t. Mean corrections still use the specified u_t. A negative information weight uses a factor downdate with feasibility condition $1 + w_t s_t > 0$. Ordinary positive row insertion applies to nonnegative weights; signed updates require the corresponding downdate algorithm and pivot margin (Hassibi et al., 1999).

The explicit array in Equation (136) provides a correctness construction. A generic dense QR on its $(C + d)$-dimensional square matrix has the corresponding dense factorization cost. Structured rotations, blocked updates, and partitioning determine the practical execution. The QR width sweep includes intermediate reads and backward for all three implementations.

Differentiation. The two factorizations represent the same smooth physical operator when the information matrices are positive definite and the local solves are well posed. A loss expressed through O, P_C, W_C therefore has the same derivatives after the appropriate change of state coordinates. For example, $\mathrm{d}P = \mathrm{d}F\,F^\top + F\,\mathrm{d}F^\top$ converts a factor tangent to a covariance tangent. Differentiating the physical equations avoids assigning significance to an arbitrary orthogonal basis. Recent work explicitly studies Gramian-based differentiation of square-root Kalman filters (Corenflos, 2026). Gradient comparisons use the same physical loss and include the state-coordinate chain rule.

N.3 Measured QR updating and stability

The custom Householder, Givens, and column-pivoted kernels update the square-root array $[\sqrt{\gamma}R; K_c]$. Table 4 gives their GPU time relative to block Woodbury at $C = 64$, $B = n_h = 8$, and $N = 2{,}048$, with reads at every update and backward. Each row compares the two executions at the same feature width. The latency analysis below describes this timing protocol. The long-stream numerical study has its own dimensions and three-seed protocol, stated in Table 43.

Latency and dependent operations. At the reported operating point, the block Woodbury recursion performs 1.4×10^9 FLOPs per chunk across all streams in 1.07 ms. The ratio table measures the actual update schedules at these chunked shapes. Their latency-sensitive execution motivates comparing dependent operations directly rather than extrapolating throughput from large dense QR factorizations. The FLOP count alone does not identify the hardware bottleneck.

With QR panel width $p = 32$, the reported block forward schedule has nine operation groups: $PK^\top$, the chunk Gram matrix, the $C \times C$ factorization, two triangular solves, the P and W updates, and two read products. The Householder schedule has d ordered reflections in $\lceil d/p \rceil$ panels, together with trailing-update and read stages, giving the count $d + \lceil d/p \rceil + 4$. Givens updating uses $C + d - 1$ anti-diagonal wavefronts; the pivoted implementation adds norm updates and pivot selection. These are schedule-level counts, not equal-cost instructions or a complete critical-path model. The factorization and solves inside a block have their own internal dependencies. At fixed C, the factor and triangular coefficient matrices have fixed order; their feature-space right-hand sides grow with d. The QR reflection or wavefront count also grows with d. The measured ratios are consistent with this latency-based explanation.

Read density. At $d = 128$, the measured Householder-to-block ratio changes from $7.4\times$ with reads at every update to $9.1\times$ with final-state-only reads. In the reported forward schedule, dropping the reads removes two of the block recursion's nine groups and four of the QR schedule's 136 stages. The ratio therefore remains unfavorable to the tested factor implementation when reads are rare. For a fixed-weight quadratic task that needs only the terminal fit, accumulating the information and response statistics and factorizing once is a separate execution option. Under fading, the accumulation must retain the same weights and initial state. Response-adaptive rules still require the intermediate predictions that select their weights.

Scope of the measurements. The measured GPU QR implementations are substantially slower than the block update on the reported sweep with reads at every update, and their measured relative cost grows with feature width. At $d = 128$, the ratios range from $7.4\times$ to $10.6\times$; at $d = 1{,}024$, they range from $57.7\times$ to $61.2\times$. The schedule comparison explains a plausible mechanism for this trend and is specific to the tested implementations. It is not a lower bound for every square-root implementation or a claim that a block factorization has constant internal depth. The numerical measurements below use the separate long-stream protocol.

Long-stream numerical stability. Token QRD inserts each observation through chronological Givens rotations. Block QRD uses Householder updates and ordered triangular solves. Table 43 compares these factor-state realizations with explicit inverse updates. These stability observations are separate from the timing sweep in Table 4.

Table 43: Long-stream state error and recovery, three seeds: 65,536 fp32 observations, $d = C = 64$, $\kappa_2(H_0) = 10^6$. Drift uses free-running trajectories; rebuilds use separate guarded runs. Inverse-state and QR-factor errors are distinct diagnostics. Dashes denote unreported values. This numerical protocol is separate from the QR timing sweep.

Recurrence	State metric	Final drift ↓	Rebuilds / 1,024 ↓
FlashRLS, fp64 core	P	$(6.0 \pm 1.3) \times 10^{-6}$	40 ± 30
Plain Sherman–Morrison	P	$(5.0 \pm 1.3) \times 10^{-5}$	240 ± 110
Symmetrized Sherman–Morrison	P	$(3.5 \pm 1.0) \times 10^{-5}$	170 ± 60
Joseph form	P	$(2.6 \pm 0.8) \times 10^{-5}$	100 ± 40
Token QRD-RLS	P via solves	$(6.5 \pm 1.5) \times 10^{-7}$	–
Block QRD-RLS	QR factor	$(5.0 \pm 1.0) \times 10^{-7}$	0

A triangular information factor R satisfies $\kappa_2(R) = \sqrt{\kappa_2(H)}$ when $H = R^\top R$. QR updates maintain this factor without subtracting dense inverse-state corrections. In the 65,536-observation comparison, block QRD's factor-state drift against its fp64 realization is $(4 \pm 1) \times 10^{-7}$, $(5 \pm 1) \times 10^{-7}$, and $(7 \pm 2) \times 10^{-7}$ at initial conditions $10^4, 10^6, 10^8$, respectively. These factor-state values are a different diagnostic from $\|\widehat{P} - P_{64}\|_F / \|P_{64}\|_F$ and should not be ranked numerically against inverse drift. Rebuilds are zero at 10^4 and 10^6 and 0–3 at 10^8. Table 15 gives comparable inverse-state errors for token QRD and stabilized sequential updates.

N.4 Square-root prediction under forgetting

Common fading maps H to γH and its information factor to $\sqrt{\gamma} R$. Adding a fixed ridge increment or a nonnegative observation weight is represented by appending the corresponding square-root rows. A signed increment requires a factor downdate, with feasibility checked before the update. These factor operations provide an alternative implementation of the structured transitions in Appendix H.

For a covariance prediction $P^- = FPF^\top + NN^\top$ with $P = UU^\top$, a QR factorization of $[FU\ N]^\top$ gives a square root of P^-. The subsequent observation array is Equation (136). State prediction, observations, and intermediate reads use factors that preserve the physical covariance and predictor.

N.5 Feature-width scaling and the role of structure

The block size and the feature width control different parts of the computation. Let a numerical block contain b observations, let the incoming information factor be $R \in \mathbb{R}^{d \times d}$, and let $K_b \in \mathbb{R}^{b \times d}$

contain its keys. The following counts concern squared-loss updates in fixed coordinates, with $O(1)$ reads per observation. They separate factor updates, response transformation, intermediate reads, and differentiation.

Generic dense refactorization. A dense Householder QR treats $[R; K_b] \in \mathbb{R}^{(d+b)\times d}$ as a general rectangular array. The leading cost for an $n \times d$ array, $n \geq d$, is $2nd^2 - \frac{2}{3}d^3$ flops (Golub & Van Loan, 2013; LAPACK, 2025). Taking $n = d + b$ gives $\frac{4}{3}d^3 + 2bd^2$. Thus the generic factorization includes a $\frac{4}{3}d^3$ term even though the incoming top block is already triangular. Applying the reflectors to response columns and producing the required intermediate reads add work. This count describes the generic dense call; it is not a lower bound for QRD-RLS.

For example, at $b = 64$, the ratio of the cubic term to the $2bd^2$ term is $2d/(3b)$. It is $4/3$, $8/3$, $16/3$, and $32/3$ at $d = 128, 256, 512, 1{,}024$, respectively. These are ratios between two terms in the QR operation count, not runtime or FlashRLS speedup estimates. The cubic term exceeds the row-dependent term when $d > 3b/2$.

Structure-aware QR updates. A chronological row insertion uses $O(d^2)$ work for the triangular factor and $O(dm)$ to transform m response columns. Applying b such updates gives $O(bd^2 + bdm)$ work. A constant number of prediction and inverse-state reads per observation can be evaluated by triangular solves within the same order. Blocked QR updates organize these operations with matrix products (Bojanczyk et al., 1993; Andrew & Dingle, 2014). Exploiting the triangular-plus-rows structure removes the avoidable cubic refactorization term.

FlashRLS and the measured comparison. The correction route for all updates has work

$$O\big(b(d^2 + dm) + b^2(d + m) + b^3\big)\,,$$

with the same dense order for its backward pass. For $b \leq d$ this is $O(b(d^2 + dm))$, and for fixed b and $m = O(d)$ it is quadratic in feature width. Its sequential local work is confined to the $b \times b$ interaction core. For response-adaptive rules, local solve and derivative costs are added as in Proposition 3.7.

The width sweep in Table 4 compares the custom QR-update schedules with block Woodbury at $d = 64$ through $1{,}024$. Each timing includes reads in observation order and backward. These implementation-specific ratios complement the arithmetic counts; the numerical study in Table 43 measures a different workload. Increasing N alone at fixed C, d, m leaves both bounded-block algorithms linear in N.

Adaptive differentiation and numerical stability. For positive w_t that depends on the current response, a QR realization can insert $\sqrt{w_t}k_t^\top$ after the local rule supplies it and differentiate both the row and the corresponding mean correction. Under the same smoothness and feasibility assumptions, the physical gradients agree after the state-coordinate chain rule, as described above and in Corenflos (2026). The algorithmic distinction is the organization of those dependencies between updates and the dense work. The numerical experiments separately measure inverse drift, factor drift, and rebuild counts. Inverse and factor drift are different state diagnostics; the token-QRD inverse measurement supplies the direct comparison with the explicit inverse state.

O CONDITIONING SHOWCASE: PUSHING ITERATIVE SOLVERS

This section pushes the read solve beyond the conditioning range of the language-model experiment in Section 4.5. All reported entries are measured over three seeds unless stated otherwise.

The arithmetic. For a symmetric positive-definite system with condition number κ, conjugate gradient and Chebyshev iteration satisfy

$$\|e_k\| \leq 2\rho^k\|e_0\|, \qquad \rho = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \approx 1 - \frac{2}{\sqrt{\kappa}}.$$

Reaching relative error ε therefore needs about $\frac{\sqrt{\kappa}}{2}\log(2/\varepsilon)$ iterations in the worst case. For $\varepsilon = 10^{-4}$, this is about 50 iterations at $\kappa = 10^2$, 500 at 10^4, 5,000 at 10^6, and 50,000 at 10^8. CG can beat

this bound on clustered spectra. Chebyshev follows the bound set by its spectral interval. In bf16, the tested iterative kernels reach a precision floor near 5×10^{-3}. The direct solve has fixed work at each shape, and its floating-point error depends on conditioning. At the frontier shape, one bf16 iterative step costs about 0.105 ms and one fp32 step about 0.31 ms. The direct kernel takes 34.1 ms.

O.1 Protocol K1: synthetic conditioning sweep

The frontier shape is $B = n_{\mathrm{h}} = 8$, $d = d_v = 128$, $N = 2{,}048$, and $C = 64$. Keys are generated with prescribed spectra so that $\kappa_2(H_N + \lambda I) \in \{10^2, 10^3, 10^4, 10^5, 10^6, 10^7, 10^8\}$. Three input seeds are used with fp32 and bf16 inputs. The solver grid contains the direct fp64-core and fp32-core paths, CG-k, and CH-k for $k \in \{30, 100, 300, 1{,}000, 3{,}000\}$ in bf16 and fp32. The sequential fp64 solution is the reference. We report relative output error, kernel time, and the first measured CG iteration count k^* that reaches 10^{-4}.

Table 44: Measured cost at matched accuracy across conditioning (Protocol K1). k^* is the first CG iteration count that reaches relative output error 10^{-4} with fp32 inputs and iterations. The cost column is CG time divided by direct-kernel time. On these clustered spectra, CG reaches the target in fewer iterations than the worst-case bound. Bold marks the largest finite CG/direct cost ratio.

κ	Direct error, fp32 inputs	direct ms	CG k^*	CG ms at k^*	CG/direct cost	CG bf16, any k
10^2	1.0×10^{-6}	34.1	32 ± 3	19 ± 1	$0.56\times$	5×10^{-3} floor
10^3	5×10^{-6}	34.1	95 ± 8	39 ± 3	$1.1\times$	floor
10^4	5.1×10^{-5}	34.1	310 ± 25	105 ± 8	$3.1\times$	floor
10^5	2×10^{-4}	34.1	980 ± 80	313 ± 25	$9.2\times$	floor
10^6	6×10^{-4}[a]	34.1	$3{,}200 \pm 250$	$1{,}000 \pm 80$	$29\times$	floor
10^7	2×10^{-3}[b]	34.1	$9{,}500 \pm 800$	$2{,}950 \pm 250$	$87\times$	floor
10^8	fp32 floor[c]	34.1	diverged in fp32	n/a	n/a	floor

[a]At $\kappa = 10^6$, guarded drift is $\leq 10^{-5}$. [b]At $\kappa = 10^7$, the guard rebuilds the state. [c]At $\kappa = 10^8$, the state is rebuilt every chunk.

The measured cost crossover occurs near $\kappa \approx 10^3$. Beyond that point, the iteration count needed for matched accuracy rises with conditioning. The direct factorization cost stays fixed at this shape. Additional bf16 iterations cannot reduce error below the kernel's precision floor.

O.2 Protocol K2: conditioning in speech

We repeat the acoustic echo-cancellation setup in Appendix M.1 with 2,048 taps and ridge one. The main AEC result uses 4,096 taps. This shorter filter under-models longer echo paths in these clips, so all methods saturate at a lower absolute ERLE, near 10 dB. K2 is therefore a conditioning stress test and is not directly comparable to the 4,096-tap ERLE values in the main AEC experiment. We add MesaNet-style chunked CG reads of $(H_t + \lambda I)q_t$ and CH-30 reads using the Frobenius-bounded eigenvalue estimate. The same clips and gates are used for every row. Speech windows have measured condition numbers from about 10^5 to 10^7.

Table 45: Measured iterative reads on speech (Protocol K2), linear scenes, 2,048 taps, ridge one, 200 clips. The shorter filter intentionally has lower absolute ERLE than the 4,096-tap main AEC experiment. At $\kappa \approx 10^6$, the scalar contraction estimate gives $\rho \approx 0.998$. Bold marks FlashRLS only where it has the best task quality.

Read rule	ERLE overall / steady (dB)	drift / solve error	ms per audio-s
FlashRLS direct, fp64 core	**9.8 / 10.5**	5.9×10^{-6}	53
FDKF	8.5 / 9.4	n/a	0.4
NLMS	2.9 / 8.5	n/a	185
CG-30, fp32	0.6 ± 0.3 / 0.9 ± 0.4	0.94	61 ± 2
CG-300, fp32	3.0 ± 0.6 / 3.8 ± 0.7	0.55	405 ± 8
CH-30, Frobenius bound	0.3 ± 0.2 / 0.5 ± 0.3	0.97	59 ± 2
CG-3,000, fp32	9.2 ± 0.5 / 9.9 ± 0.5	10^{-3}	$3{,}620 \pm 40$

Short iterative reads leave most of the difficult directions unresolved. CG-3,000 approaches the direct ERLE and takes 3,620 ms per audio second versus 53 ms for the direct solve, a measured $68.3\times$ cost ratio at nearly matched quality.

O.3 Protocol K3: near-duplicate recall

We modify MQAR so that groups of eight keys are small perturbations of one base key. Their within-group cosine similarity is $c \in \{0.9, 0.99, 0.999\}$ and their values remain distinct. Ridge is chosen from $\{1, 10^{-2}, 10^{-4}\}$, giving stored-key condition numbers from about 10^2 to 10^7. Direct, CG-30, CG-300, and CH-30 reads are evaluated on the same weights over three seeds. Accuracy is measured at 128 stored pairs.

Table 46: Measured near-duplicate recall (Protocol K3), accuracy at 128 stored pairs. The standard uncorrelated solver swap at $\lambda = 0.01$ gives 99.9% direct, 99.9% CG-30, and 99.6% CG-8. Bold marks direct FlashRLS only where it is strictly best.

Keys / ridge	κ (approx.)	Direct	CG-30	CG-300	CH-30
$c = 0.9$, $\lambda = 10^{-2}$	10^3	99 ± 1	97 ± 1	99 ± 1	95 ± 2
$c = 0.99$, $\lambda = 10^{-2}$	10^4	97 ± 1	71 ± 4	95 ± 2	60 ± 5
$c = 0.99$, $\lambda = 10^{-4}$	10^6	98 ± 1	22 ± 5	55 ± 6	15 ± 4
$c = 0.999$, $\lambda = 10^{-4}$	10^7	93 ± 2	13 ± 3	30 ± 6	13 ± 3

For two nearly duplicate keys, the distinguishing direction has energy $1 - c^2$ and ridge gain $(1 - c^2)/(1 - c^2 + \lambda)$. A fixed iterative budget increasingly fails to recover this weakening direction.

Summary. At the conditioning produced by the default Gated KalmaNet regularizer, short iterative solves can be cheaper and accurate. The crossover appears near $\kappa \approx 10^3$ in Protocol K1. Beyond that point, the direct factorization cost stays fixed at this shape. The matched-accuracy CG cost grows to $3.1\times$ at 10^4, $29\times$ at 10^6, and $87\times$ at 10^7. The same effect appears on real inputs. In the 2,048-tap speech stress test at $\kappa \approx 10^6$, CG-30 gives less than 1 dB steady ERLE. The direct solve gives 10.5 dB. Near-duplicate recall at the same conditioning is 22% for CG-30 and 98% for the direct solve. The iteration count alone is sufficient to produce these errors.

P TEN-SEED STUDIES

The tables below give the ten-seed results used in the main-text adaptive-backward summary and the additional ten-seed checks. Paired rows share initialization, data order, or hashing seeds within each protocol. Means and standard deviations are the reported summaries; paired confidence intervals use Student-t_9 at 95%. Seed identifiers belong to the study named in each caption. Three-seed studies are described separately in Appendices L.6 and L.5.

P.1 ADAPTIVE BACKWARD PASS

Table 47: Ten-seed B1 MQAR training: steps to 99% accuracy within 3,000 steps. The CG summary covers the eight runs reaching the criterion; n.r. denotes not reached. The stopped/exact ratio has a 95% confidence interval.

Seed	Exact adaptive	Stopped curvature	CG-30 implicit	Quadratic null	Stopped/ exact
s0	1,161	2,035	2,598	959	1.75
s1	1,072	1,924	2,409	1,040	1.80
s2	1,064	2,007	1,726	911	1.89
s3	1,189	2,034	n.r.	1,187	1.71
s4	1,099	1,950	2,473	956	1.77
s5	984	1,870	1,814	910	1.90
s6	957	1,651	2,450	845	1.73
s7	1,152	2,061	2,076	976	1.79
s8	1,190	2,164	n.r.	1,067	1.82
s9	1,233	2,273	2,970	1,069	1.84
Reported summary	$1{,}110 \pm 91$	$1{,}997 \pm 168$	$2{,}314 \pm 417$ (8/10)	992 ± 100	1.80 ± 0.05 CI

Table 48: Ten-seed B1 MQAR recall (%) at 224 pairs, evaluated at step 3,000 for every arm and seed. Summaries are mean $\pm$ standard deviation.

Seed	Exact adaptive	Stopped curvature	CG-30 implicit	Frozen memory	Quadratic null
s0	89.5	80.1	72.7	56.5	73.4
s1	90.0	81.9	72.1	54.2	74.3
s2	89.2	80.8	64.4	46.4	68.0
s3	90.4	80.8	75.1	53.4	78.7
s4	88.8	77.2	73.6	47.7	73.7
s5	87.4	80.4	72.8	45.5	72.5
s6	86.0	74.6	65.7	40.2	66.9
s7	87.7	76.8	69.9	46.8	72.0
s8	90.4	82.2	71.8	57.4	76.5
s9	91.7	82.1	73.5	57.9	72.4
Reported summary	89.1 ± 1.7	79.7 ± 2.6	71.2 ± 3.5	50.6 ± 6.0	72.8 ± 3.5

Table 49: Paired recall differences (percentage points) in the ten-seed B1 study. Summary intervals are 95% confidence intervals.

Seed	Exact − stopped	Stopped − CG-30	Exact − frozen
s0	9.4	7.4	33.0
s1	8.1	9.8	35.8
s2	8.4	16.4	42.8
s3	9.5	5.7	36.9
s4	11.6	3.6	41.1
s5	6.9	7.6	41.9
s6	11.5	8.9	45.8
s7	10.9	6.9	40.9
s8	8.1	10.4	33.0
s9	9.5	8.6	33.8
Reported summary	9.4 (CI ±1.1)	8.5 (CI ±2.4)	38.5 (CI ±3.3)

Table 50: Ten-seed B2 outlier-regression MSE at 256 context points. The offline batch Huber and GP posterior references are 0.051 and 0.19. The mean stopped/exact MSE ratio is 1.50 with 95% CI ±0.08.

Seed	Exact adaptive	Stopped curvature	CG-30 implicit	Frozen memory	Squared loss, exact
s0	0.056	0.089	0.079	0.114	0.142
s1	0.049	0.080	0.068	0.084	0.126
s2	0.057	0.085	0.114	0.178	0.140
s3	0.056	0.075	0.088	0.206	0.176
s4	0.056	0.083	0.071	0.125	0.138
s5	0.063	0.106	0.109	0.177	0.183
s6	0.058	0.088	0.111	0.195	0.172
s7	0.058	0.082	0.160	0.172	0.173
s8	0.055	0.083	0.066	0.151	0.172
s9	0.058	0.081	0.112	0.173	0.189
Reported summary	0.057 ± 0.004	0.085 ± 0.008	0.098 ± 0.029	0.158 ± 0.039	0.161 ± 0.022

Table 51: Paired MSE differences in the ten-seed B2 study. Summary intervals are 95% confidence intervals.

Seed	Stopped − exact	Squared − Huber exact
s0	0.032	0.085
s1	0.032	0.077
s2	0.028	0.083
s3	0.019	0.120
s4	0.027	0.082
s5	0.042	0.120
s6	0.030	0.114
s7	0.024	0.115
s8	0.028	0.117
s9	0.023	0.130
Reported summary	0.029 (CI ±0.005)	0.104 (CI ±0.014)

P.2 STREAMING LOGLOSS AND NUMERICAL STABILITY

Table 52: Criteo single-pass logloss on the final ten million rows, ten hashing seeds. The offline batch IRLS fit uses the whole stream. Summaries are mean ± standard deviation.

Seed	FTRL-Proximal	Adagrad-SGD	FlashRLS-GLM (= fp64 ONS)	Batch IRLS offline
s0	0.4714	0.4698	0.4632	0.4605
s1	0.4717	0.4696	0.4635	0.4615
s2	0.4717	0.4703	0.4633	0.4612
s3	0.4718	0.4701	0.4628	0.4612
s4	0.4720	0.4699	0.4628	0.4608
s5	0.4709	0.4698	0.4626	0.4610
s6	0.4715	0.4705	0.4629	0.4616
s7	0.4715	0.4706	0.4628	0.4610
s8	0.4707	0.4680	0.4628	0.4603
s9	0.4727	0.4692	0.4629	0.4609
Reported summary	0.4716 ± 0.0006	0.4698 ± 0.0008	0.4630 ± 0.0003	0.4610 ± 0.0004

Table 53: Paired Criteo logloss differences over ten hashing seeds. Summary intervals are 95% confidence intervals.

Seed	Adagrad − FlashRLS	FTRL − FlashRLS	FlashRLS − batch
s0	0.0066	0.0082	0.0026
s1	0.0061	0.0082	0.0020
s2	0.0070	0.0083	0.0022
s3	0.0073	0.0089	0.0016
s4	0.0071	0.0092	0.0021
s5	0.0072	0.0083	0.0016
s6	0.0077	0.0086	0.0013
s7	0.0079	0.0087	0.0018
s8	0.0053	0.0080	0.0025
s9	0.0063	0.0098	0.0020
Reported summary	0.0068 (CI ±0.0006)	0.0086 (CI ±0.0004)	0.0020 (CI ±0.0003)

Table 54: Matched stability at $\kappa_2(H_0) = 10^6$, $\rho = 0.99$, ten key-stream seeds. FlashRLS drift is in 10^{-6} units. Each sequential entry is its drift divided by FlashRLS drift on the same seed.

Seed	FlashRLS drift ($\times 10^{-6}$)	Plain SM / FlashRLS	Symmetric SM / FlashRLS	Joseph / FlashRLS
s0	5.6	8.9	3.4	2.9
s1	4.0	11.8	5.4	3.2
s2	7.1	11.3	4.8	3.1
s3	5.9	7.0	7.1	4.4
s4	5.8	6.0	6.1	4.8
s5	7.4	6.6	4.9	4.3
s6	5.4	10.0	6.5	2.7
s7	5.5	5.7	6.8	3.4
s8	8.9	9.0	9.6	4.3
s9	4.0	8.1	4.1	4.2
Reported summary	6.0 ± 1.5	8.4 ± 2.1	5.9 ± 1.8	3.7 ± 0.7

P.3 SOLVER CHECKS

Table 55: Ten-seed 20M-token Gated KalmaNet MQAR-harness study: recall (%) at 224 pairs. Every solver uses Gated KalmaNet gates. This protocol is separate from the 50M, one-billion-token solver study.

Seed	Direct $a = 0.005$	CH-30 $a = 0.005$	Direct $a = 0.02$	CH-30 $a = 0.02$
s0	88.4	80.7	79.7	78.7
s1	90.0	85.1	83.9	85.4
s2	85.7	74.5	74.0	74.4
s3	89.2	81.0	84.9	85.4
s4	86.9	82.6	79.8	79.9
s5	87.7	83.5	80.3	82.1
s6	87.5	82.6	78.5	76.1
s7	91.0	86.8	80.9	79.7
s8	88.6	86.2	80.2	79.2
s9	88.8	79.8	83.1	84.6
Reported summary	88.4 ± 1.5	82.3 ± 3.6	80.5 ± 3.1	80.5 ± 3.8

Table 56: Paired direct-minus-CH-30 recall differences (points) in the ten-seed MQAR-harness study. At $a = 0.02$ the paired interval contains zero; at $a = 0.005$ it does not.

Seed	$a = 0.005$	$a = 0.02$
s0	7.7	1.0
s1	4.9	-1.5
s2	11.2	-0.5
s3	8.2	-0.4
s4	4.2	-0.1
s5	4.2	-1.8
s6	4.9	2.5
s7	4.1	1.2
s8	2.4	1.0
s9	9.0	-1.5
Reported summary	6.1 (CI $\pm$2.0)	0.0 (CI $\pm$1.0)

P.4 SMALL LANGUAGE-MODEL AND KERNEL STUDIES

Table 57: Ten-seed 20M-token WikiText-103 perplexity, mean $\pm$ standard deviation. Gate variants are listed separately. The three-seed MesaNet-gated FlashRLS value 160 ± 2 in Table 22 belongs to a separate study.

Seed	Gated DeltaNet	FlashRLS ungated	MesaNet CG-30	FlashRLS MesaNet gates	FlashRLS FlashRLS gates
s0	154.77	173.99	157.62	157.87	171.00
s1	154.17	175.46	159.10	159.00	171.45
s2	156.14	173.30	160.51	160.54	172.42
s3	155.41	173.71	157.52	157.40	170.75
s4	154.41	173.88	157.23	157.31	170.30
s5	155.64	174.86	159.74	159.84	172.90
s6	153.92	174.97	157.05	156.03	167.01
s7	154.47	174.18	156.30	156.83	169.09
s8	156.79	175.00	159.96	159.71	171.96
s9	154.41	173.82	158.46	158.02	169.59
Reported summary	155.01 ± 0.94	174.32 ± 0.70	158.35 ± 1.42	158.25 ± 1.46	170.65 ± 1.75

Table 58: Paired direct-minus-MesaNet perplexity differences in the ten-seed WikiText-103 study, with 95% confidence intervals. The two direct rows use distinct gate parameterizations.

Seed	Direct, MesaNet gates	Direct, FlashRLS gates
s0	0.24	13.38
s1	−0.11	12.35
s2	0.03	11.91
s3	−0.12	13.22
s4	0.08	13.07
s5	0.10	13.16
s6	−1.02	9.95
s7	0.53	12.79
s8	−0.25	12.00
s9	−0.44	11.13
Reported summary	−0.10 (CI ±0.30)	12.30 (CI ±0.78)

Table 59: Common-shape kernel timing at $N = 2{,}048$, forward plus backward, ten input seeds. CUDA-graph times are in milliseconds; input-seed timing variation is below 1%. These are kernel measurements, not full language-model training times.

Seed	FlashRLS	MesaNet CG-30	Gated DeltaNet
s0	34.01	6.07	1.55
s1	34.21	6.12	1.57
s2	33.97	6.07	1.56
s3	34.10	6.11	1.57
s4	34.23	6.10	1.56
s5	34.09	6.14	1.55
s6	34.18	6.07	1.55
s7	34.07	6.07	1.57
s8	34.34	6.14	1.58
s9	33.98	6.10	1.57
Reported summary	34.12 ± 0.12	6.10 ± 0.03	1.56 ± 0.01