Query Dual Dynamics Delta: Read-Driven Modulation of Delta-Rule Writes in Fixed-Capacity Fast-Weight Memory

Dian Yu[1*] and Haoyu Yang[1]

[1*], University of New South Wales, Sydney, NSW, 2052, Australia.

Abstract

A fixed-capacity fast-weight memory is normally updated from the stream being written, although its usefulness is determined by a possibly different stream of future reads. This paper introduces Query Dual Dynamics Delta ($\mathbf{QD^3}$), an online operator that turns past queries into causal guidance for subsequent Delta writes. One state stores key–value associations; a second, low-rank state follows the dominant directions and energy of the query stream. Their only coupling is an interpretable scalar: the estimated historical query heat of the incoming write key. The operator therefore treats memory maintenance as coordination between two interacting event streams in an adaptive intelligent system. Query heat increases the residual-correction rate, while a hard cap preserves the non-expansive single-address behavior of the normalized Delta rule. The construction adds $\boldsymbol{O(d_k r)}$ state, is constant in sequence length, and recovers ordinary Delta exactly when query modulation is disabled. We evaluate the operator without end-to-end training so that memory behavior is measured directly. Hyperparameters are selected on eight validation seeds and frozen for 16 paired test seeds. $\mathbf{QD^3}$ reduces normalized recall error by 31.0% on a concentrated spatial field, 18.7% on a source-token workload, 32.1% on a module-import graph, and 2.62% on held-out WikiText-2 token sequences. An estimator- and capacity-matched write-trace control establishes that the additional information comes from reads rather than extra state. Uniform-query, moving-hotspot, source, and rank controls further delimit the mechanism.

Keywords: fast-weight memory, Delta rule, linear attention, adaptive intelligent systems, workload adaptation, online subspace learning

1 Introduction

A compact memory is valuable precisely because it does not grow with the history it summarizes. The same constraint also creates competition: after sufficiently many key–value pairs have been written, improving one association can disturb another. Delta-style fast weights address part of this problem by writing only the residual that is still missing at the current key [10, 12]. They answer a local question—*what is wrong at this write address?*—but not the allocation question—*which addresses will matter when this memory is read?*

That distinction is immaterial when writes and reads have the same distribution. It becomes important for a long-running service, agent, map, or compressed sequence state in which the stored population is broad but the workload is concentrated. Viewed as an adaptive system, the memory is driven by two coupled but statistically different event streams: writes determine what can be stored, while reads reveal which stored content serves the active workload. A small set of entities may be queried repeatedly even though every entity is written with comparable frequency. Write counts cannot reveal this demand. Recency cannot reveal it either. The read events themselves carry the missing signal.

We therefore treat a query as more than a transient probe. In Query Dual Dynamics Delta (QD^3), reads learn a compact workload state, writes learn the key–value map, and a single non-negative heat value connects them. The core innovation is this causal cross-event allocation: accumulated read geometry controls the rate of *later* residual writes in a continuous, fixed-capacity memory. It neither performs an instantaneous query correction nor evicts discrete cached tokens. The two independently evolving states give the method its name.

The design follows four principles: only earlier reads may affect a write; cold-start behavior must remain ordinary Delta; the workload state must be much smaller than the memory it guides; and prioritization must remain inside the normalized Delta stability interval. These requirements lead directly to a causal low-rank query trace and a clipped effective write rate.

We study the operator separately from an end-to-end model so that its central hypothesis is directly testable: given identical writes, capacity, and evaluation samples, does read history improve later recall? The study uses a smooth two-dimensional field, two real-data-derived controlled workloads, and held-out WikiText-2 sequences [5]. Its decisive baseline matches the low-rank estimator, state size, update count, normalization, and tuning budget, but learns from write keys instead of query keys.

The contributions are:

- a causal read-driven Delta operator that uses accumulated query geometry to allocate corrective write rate;
- a low-rank query trace whose heat has a direct interpretation as discounted squared query–key alignment, with constant state in the stream length;
- a clipped update with exact reduction to ordinary Delta and a simple non-expansion guarantee at the written address; and

- a paired, validation-separated study that includes an estimator-matched source control, full-moment and weighted-linear references, rank and source ablations, uniform-query controls, and a changing workload.

2 Related work

2.1 Fast weights and Delta-based linear attention

Fast weights are parameters that change on the time scale of a sequence and store temporary associations [1, 8]. Kernelized linear attention can be written as a recurrent matrix update with state independent of sequence length [3]. The fast-weight-programming view makes the connection explicit: keys and values form programming instructions for a finite associative matrix [10]. Purely additive writing accumulates interference, whereas the Delta rule first subtracts the value already predicted at the current key and writes only the residual. Parallel DeltaNet later supplied a sequence-parallel implementation and demonstrated the rule at language-model scale [14].

Recent work has improved how the current token edits this matrix. Gated DeltaNet combines targeted Delta correction with a forgetting gate [13]; the 2026 preprint Gated DeltaNet-2 (arXiv:2605.22791) separates channel-wise erase and write gates; and DeltaProduct applies several Delta steps per token, increasing the rank and expressivity of the transition [9]. These mechanisms decide how to edit from the current input and learned gates. QD^3 addresses a different variable: the empirical distribution of earlier reads.

The closest query-aware operators also use the current query more directly. Q-Delta mixes current key- and query-based prediction errors in one state transition [7]. The 2026 preprint Query-derived Erase Direction (arXiv:2608.13668) introduces a second, current-query erase direction to remove interference that a key-directed edit cannot reach. Both are instantaneous query–write couplings at a sequence position. QD^3 instead accumulates a query second-moment surrogate over many read events and uses it to price future writes. The distinction is operational: a current query answers *what should be corrected now?*; a query history estimates *where future correction effort is likely to be useful?*

2.2 Access-aware retention and online subspaces

Read frequency has also been exploited in explicit key–value caches. H_2O retains attention heavy hitters together with recent tokens [15], while SnapKV estimates important prompt positions from an observation window and compresses the discrete cache [4]. Those methods choose which token records remain available. A fast-weight matrix has no token list to evict after associations have been superposed, so access information must act through the continuous write rule instead. This is the gap addressed here.

The low-rank query state is updated by an Oja-type subspace rule [6]. Oja's algorithm is a standard streaming route to principal directions, with established multi-component guarantees [2]. OjaKV recently used online Oja adaptation for low-rank compression of an explicit KV cache [16]. In QD^3, no cached value is projected or

Table 1 Position of QD^3 among memory-management mechanisms. "Current" denotes information at the present sequence position; "history" denotes a separately accumulated statistic. Citations and fuller distinctions are given in Sect. 2.

Method family	Guidance	What changes	Difference from QD^3
DeltaNet	current key and residual	one corrective write	no read-history model
Gated DeltaNet	current-token gates	forgetting and/or writing	learned current-token control
DeltaProduct	current-token residuals	multiple Delta steps	increases transition rank
Q-Delta / QED	current query	correction or erase direction	instantaneous query-aware edit
H_2O / SnapKV	attention observations	stored token positions	evicts from a discrete KV cache
QD^3 (ours)	query history	future residual-write rate	cumulative continuous-state allocation

discarded by this subspace. It approximates only the query workload and produces a scalar importance signal for the separate value memory.

At a statistical level, the motivation is related to covariate-shift correction: when training and test inputs differ, importance weighting aligns the fitted objective with the test distribution [11]. Exact importance weighting requires a density ratio. QD^3 neither estimates nor claims to recover that ratio. It uses a cheaper geometric proxy—discounted query energy near a key—that can be maintained online when only read addresses, not read-time target values, are available.

3 Problem formulation

Consider two interleaved event streams. A write event provides a key–value pair $(\boldsymbol{k}_t, \boldsymbol{v}_t)$ with $\boldsymbol{k}_t \in \mathbb{R}^{d_k}$ and $\boldsymbol{v}_t \in \mathbb{R}^{d_v}$. A query event provides an address $\boldsymbol{q}_j \in \mathbb{R}^{d_k}$. The indices t and j count writes and queries separately; $j(t)$ denotes the number of queries observed before write t. All keys and queries are normalized to unit Euclidean norm in our experiments.

The fixed-capacity value state is $\boldsymbol{M}_t \in \mathbb{R}^{d_v \times d_k}$. A query is answered by

$$\widehat{\boldsymbol{v}}(\boldsymbol{q}_j) = \boldsymbol{M}_t \boldsymbol{q}_j. \tag{1}$$

For a write, ordinary Delta first computes the current prediction and residual,

$$\widehat{\boldsymbol{v}}_t = \boldsymbol{M}_{t-1}\boldsymbol{k}_t, \qquad \boldsymbol{e}_t = \boldsymbol{v}_t - \widehat{\boldsymbol{v}}_t, \tag{2}$$

then performs the rank-one correction

$$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + \eta \boldsymbol{e}_t \boldsymbol{k}_t^{\mathsf{T}}, \tag{3}$$

where $\eta \geq 0$ is the write rate. Because $\|\boldsymbol{k}_t\| = 1$, Eq. (3) is a stochastic-gradient step on $\frac{1}{2}\|\boldsymbol{v}_t - \boldsymbol{M}\boldsymbol{k}_t\|^2$.

Let p_w denote the distribution of associations presented for writing and p_q the workload under which recall is evaluated. The desired linear state minimizes

$$\mathcal{L}_q(\boldsymbol{M}) = \frac{1}{2}\,\mathbb{E}_{(\boldsymbol{k},\boldsymbol{v})\sim p_q}\left[\|\boldsymbol{v} - \boldsymbol{M}\boldsymbol{k}\|^2\right]. \tag{4}$$

Plain Delta samples gradients under p_w. If $p_q \neq p_w$, even a well-tuned write process spends its updates according to the population presented, not the population later requested. An exact correction would multiply each gradient by $p_q(\boldsymbol{k})/p_w(\boldsymbol{k})$, but estimating a high-dimensional density ratio online is a substantially different problem. We seek a local signal that is available from queries alone and cheap enough to attach to every Delta memory.

4 Query Dual Dynamics Delta

4.1 The full query heat

Suppose first that storing a full $d_k \times d_k$ query statistic is affordable. After query j, define the exponentially weighted, uncentered second moment

$$\boldsymbol{Q}_j = \lambda \boldsymbol{Q}_{j-1} + \boldsymbol{q}_j \boldsymbol{q}_j^\top, \qquad 0 < \lambda \leq 1. \tag{5}$$

For any candidate write key $\boldsymbol{k}$, its unnormalized query heat is

$$\bar{h}_j(\boldsymbol{k}) = \boldsymbol{k}^\top \boldsymbol{Q}_j \boldsymbol{k} = \sum_{\ell=1}^{j} \lambda^{j-\ell} (\boldsymbol{q}_\ell^\top \boldsymbol{k})^2. \tag{6}$$

Equation (6) gives the central quantity a direct reading. Every earlier query votes for a new write in proportion to squared alignment in key space; the square makes the vote independent of sign, and λ determines how quickly old demand is forgotten. A key receives high heat when it lies in directions that have repeatedly been queried. No queried target value is required.

The full trace is useful as a reference, but its d_k^2 state can rival or exceed the memory being guided. QD^3 therefore maintains a low-rank surrogate.

4.2 Low-rank query dynamics

Let $\boldsymbol{U}_j = [\boldsymbol{u}_{j,1}, \ldots, \boldsymbol{u}_{j,r}] \in \mathbb{R}^{d_k \times r}$ contain $r \ll d_k$ adaptive directions and let $\boldsymbol{a}_j \in \mathbb{R}^r_+$ store their discounted energies. When query $\boldsymbol{q}_j$ arrives, we first project it into the current subspace,

$$\boldsymbol{z}_j = \boldsymbol{U}_{j-1}^\top \boldsymbol{q}_j. \tag{7}$$

The residual $\boldsymbol{q}_j - \boldsymbol{U}_{j-1}\boldsymbol{z}_j$ is the part not reconstructed by that subspace. An Oja-type step rotates the directions toward this missing component:

$$\boldsymbol{U}_j = \boldsymbol{U}_{j-1} + \eta_u \big(\boldsymbol{q}_j - \boldsymbol{U}_{j-1}\boldsymbol{z}_j\big)\boldsymbol{z}_j^\top, \tag{8}$$

where η_u is the subspace rate. At the same event, projected energy is accumulated as

$$\boldsymbol{a}_j = \lambda \boldsymbol{a}_{j-1} + \boldsymbol{z}_j \odot \boldsymbol{z}_j. \tag{9}$$

Here $\odot$ denotes elementwise multiplication. Equations (8) and (9) have complementary roles: $\boldsymbol{U}_j$ tracks *which directions* queries occupy, while $\boldsymbol{a}_j$ tracks *how much demand* lies along each direction. The factorized moment surrogate is $\boldsymbol{U}_j \operatorname{diag}(\boldsymbol{a}_j)\boldsymbol{U}_j^{\mathsf{T}}$, so the heat assigned to an incoming write key is

$$\bar{h}_j(\boldsymbol{k}_t) = \sum_{i=1}^{r} a_{j,i}(\boldsymbol{u}_{j,i}^{\mathsf{T}}\boldsymbol{k}_t)^2. \tag{10}$$

The magnitude of Eq. (10) depends on query count and geometry. To keep β interpretable across streams, we maintain one causal scale

$$s_j = \max\left\{ s_{j-1}, \sum_{i=1}^{r} a_{j,i}(\boldsymbol{u}_{j,i}^{\mathsf{T}}\boldsymbol{q}_j)^2, \epsilon \right\}, \qquad h_j(\boldsymbol{k}) = \frac{\bar{h}_j(\boldsymbol{k})}{s_j}, \tag{11}$$

with a small numerical $\epsilon > 0$. This maximum-observed scale is a normalization device, not an additional learned parameter. It uses only information available at query j.

4.3 Read-driven rate and memory update

For write t, QD^3 uses the most recent query state $j(t)$ and sets

$$\widetilde{\eta}_t = min\left\{\eta\big[1 + \beta h_{j(t)}(\boldsymbol{k}_t)\big],\ c\right\}, \qquad \beta \geq 0, \tag{12}$$

followed by

$$\boldsymbol{M}_t = \boldsymbol{M}_{t-1} + \widetilde{\eta}_t\big(\boldsymbol{v}_t - \boldsymbol{M}_{t-1}\boldsymbol{k}_t\big)\boldsymbol{k}_t^{\mathsf{T}}. \tag{13}$$

The constant 1 in Eq. (12) matters. It retains the ordinary Delta update during cold start and for directions with no query evidence. The coefficient β controls only the additional effort assigned by the workload trace. Setting $\beta = 0$ makes $\widetilde{\eta}_t = \eta$ whenever $\eta \leq c$, so Eq. (13) is exactly Eq. (3); this equality is also tested numerically in both implementations.

The cap c is a safeguard for the residual correction, not a cosmetic clamp. For unit-norm keys, the following property is immediate.

Proposition 1 (Written-address non-expansion) *Let $\|\boldsymbol{k}_t\| = 1$ and $0 \leq \widetilde{\eta}_t \leq 2$. If $\boldsymbol{e}_t$ is the pre-update residual in Eq. (2), the residual at the same key after Eq. (13) is*

$$\boldsymbol{e}_t^{+} = (1 - \widetilde{\eta}_t)\boldsymbol{e}_t, \tag{14}$$

and therefore $\|\boldsymbol{e}_t^{+}\| \leq \|\boldsymbol{e}_t\|$.

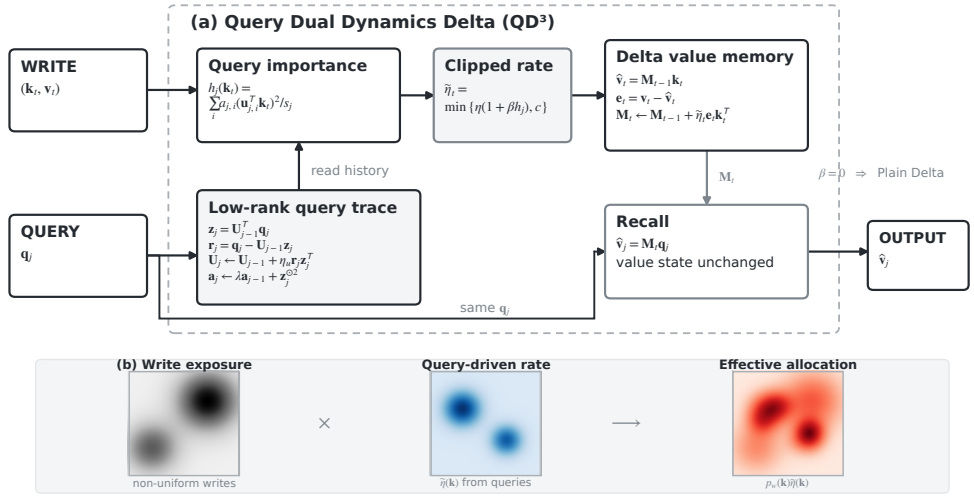

Fig. 1 Query Dual Dynamics Delta. (a) Query history determines the clipped rate of later residual Delta writes. (b) Query history sets the per-write rate $\widetilde{\eta}(\mathbf{k})$, whereas write exposure $p_w(\mathbf{k})$ determines how often it is applied; their product shows effective update opportunity. The exact update also contains the residual. Maps are schematic, not additional state.

Proof Multiplying Eq. (13) by $\boldsymbol{k}_t$ and using $\boldsymbol{k}_t^{\mathsf{T}}\boldsymbol{k}_t = 1$ gives $\boldsymbol{M}_t\boldsymbol{k}_t = \boldsymbol{M}_{t-1}\boldsymbol{k}_t + \widetilde{\eta}_t\boldsymbol{e}_t$. Subtracting from $\boldsymbol{v}_t$ yields the stated identity. The norm inequality follows from $|1-\widetilde{\eta}_t| \le 1$ on $[0,2]$. □

We use $c = 1.9$, leaving a margin below the endpoint. Proposition 1 is local to the written address; as with every shared fast-weight matrix, interference at other keys is still possible. It nevertheless explains why workload amplification is clipped before entering the Delta rule.

Figure 1 summarizes the complete event flow. Query events update $(\boldsymbol{U}, \boldsymbol{a}, s)$ and are answered from $\boldsymbol{M}$ without directly changing it. Write events consult the frozen query state to update $\boldsymbol{M}$ and do not train the query trace. This separation prevents write frequency from leaking into the quantity meant to represent read demand.

4.4 State and computational cost

The value memory contains $d_v d_k$ scalars. QD^3 adds $d_k r + r + 1$ scalars for $\boldsymbol{U}$, $\boldsymbol{a}$, and s, independent of the number of events. Its additional state fraction is therefore

$$\frac{d_k r + r + 1}{d_v d_k} \approx \frac{r}{d_v} \quad \text{when } d_k, d_v \text{ are not small.} \tag{15}$$

A query needs $O(d_v d_k)$ work for recall and $O(d_k r)$ for the trace update. A write retains the $O(d_v d_k)$ Delta prediction and outer product and adds an $O(d_k r)$ heat calculation. The full-trace reference adds d_k^2 state and $O(d_k^2)$ trace work. The low-rank construction is consequently most attractive when $r \ll \min(d_k, d_v)$.

5 Experimental design

5.1 Tasks and evaluation metric

The experiments isolate memory quality: there is no learned encoder, classifier, or end-to-end task head. This avoids attributing to the write rule gains that could instead come from additional model training.

5.1.1 Synthetic spatial field

Write locations are sampled uniformly from $[0, 1]^2$. A fixed 64-dimensional random Fourier map produces unit keys, and an eight-dimensional smooth harmonic field produces values with additive Gaussian noise of standard deviation 0.05. Query and evaluation locations are sampled independently. In the concentrated condition they come from two Gaussian hotspots; in the uniform condition they cover the square uniformly. A moving-hotspot condition switches to a disjoint pair halfway through the query stream. Each run has 3,000 writes and 5,000 evaluation samples. This controlled source is included to expose the mechanism under known independent distributions, rather than as an application benchmark.

5.1.2 Real-data-derived controlled workloads

Two structurally different streams are constructed from source-code text. The token workload contains 3,000 identifiers: short-window co-occurrence features are reduced to $d_k = 128$ keys, long-window profiles give $d_v = 64$ values, and observed token counts define read frequency. The module-import graph contains 578 modules: co-import structure gives $d_k = 24$ keys and $d_v = 48$ values, while import counts define the read distribution. Writes traverse every item in random order for three and six passes, respectively. These tasks retain real heavy-tailed workloads and geometry while allowing exact control of the read distribution. They are used for mechanism confirmation, not presented as end-to-end software applications.

5.1.3 WikiText-2 held-out token sequence

WikiText-2 is a public corpus of curated Wikipedia articles [5]. The training split fixes a 3,000-word vocabulary and builds a unit-normalized 256-dimensional short-window co-occurrence key for each word; normalized 64-dimensional long-window context profiles are the stored values. The write stream makes two independently shuffled passes over these 3,000 associations. Query events and evaluation items, however, come from a contiguous, disjoint window of the held-out validation or test token sequence: a query prefix is immediately followed by the scored portion. A shuffled-query control keeps the same query multiset and changes only its order. The prepared vocabulary covers 84.1% of held-out test tokens, and the most frequent 10% of vocabulary items account for 72.5% of covered test occurrences. Each run evaluates 3,000 held-out tokens.

For N evaluation associations $(\boldsymbol{k}_n, \boldsymbol{v}_n)$, lower normalized root mean squared error (NRMSE) is better:

$$\mathrm{NRMSE} = \frac{\sqrt{N^{-1}\sum_{n=1}^{N}\|\boldsymbol{M}\boldsymbol{k}_n - \boldsymbol{v}_n\|^2}}{\sqrt{N^{-1}\sum_{n=1}^{N}\|\boldsymbol{v}_n\|^2}}. \tag{16}$$

Sampling evaluation items from the query workload makes Eq. (16) a Monte Carlo estimate of Eq. (4), normalized for comparability.

5.2 Baselines and controls

Plain Delta is Eq. (3). The main source control, *Write Trace*, is deliberately matched to QD^3: it uses the same Oja update, rank choices, energy decay, heat normalization, number of trace events, rate equation, and validation budget, but supplies the contemporaneous write key to Eqs. (7)–(9). A difference between these two methods can therefore be attributed to the information source rather than additional state or computation.

The secondary *Dual Trace* control maintains independent query and write traces. Their normalized heats h_q and h_w are combined as $h_q/(h_q + h_w + \epsilon)$ before the same clipped rate equation. Per-trace ranks $\{1, 2, 4\}$ give total ranks $\{2, 4, 8\}$, matching the auxiliary-state and 75-candidate validation budgets of QD^3. This control asks whether the same limited state is better divided between demand and write exposure.

The *Full Query Trace* uses Eq. (5) without low-rank approximation. The *Stationary Query Moment* receives the true stationary query second moment and uses the same scalar gain; it is an offline reference, not an upper bound. The *Weighted Linear* reference solves ridge-stabilized least squares under the evaluation weights. It is optimal only within a stationary linear-map class and is not a bound on arbitrary memory systems.

Mechanism ablations replace the learned low-rank trace with a coordinate-wise diagonal moment, the most recent query alone, or a fixed random rank-four subspace. Rank is swept over $r \in \{1, 2, 4, 8, 16\}$. Uniform queries test the case in which the query source carries no distributional preference beyond broad coverage. Query-frequency power $\alpha \in \{0, 0.25, 0.5, 0.75, 1, 1.25\}$ continuously changes the controlled discrete workloads from uniform ($\alpha = 0$) to and beyond their observed skew ($\alpha = 1$).

5.3 Selection, pairing, and implementation checks

Pilot seeds inspected during development are excluded. For the synthetic and controlled workloads, eight new validation seeds (2001–2008) select hyperparameters and 16 locked test seeds (10001–10016) are evaluated once. WikiText-2 uses validation seeds 4001–4008 and disjoint test seeds 30001–30016. Every method within a seed receives the same write order, query sequence, and evaluation samples. Each test seed occupies a disjoint block of the held-out token sequence before its contiguous window is selected.

The main search uses $\eta \in \{0.02, 0.05, 0.1, 0.2, 0.4\}$, $\beta \in \{1, 3, 10, 30, 100\}$, and $r \in \{2, 4, 8\}$ for low-rank methods. WikiText-2 uses $\eta \in \{0.02, 0.05, 0.1\}$, $\beta \in \{1, 3, 10, 30\}$, and $r \in \{1, 2, 4\}$. Each method is tuned separately on validation NRMSE. Fixed operator settings are $\eta_u = 0.05$, $\lambda = 0.999$, $c = 1.9$, and one query event per three writes.

We report test means, sample standard deviations, and paired differences with two-sided 95% Student-t confidence intervals over 16 seeds. The pre-specified primary comparisons are QD^3 versus Plain Delta and versus Write Trace; full-trace comparisons are secondary. The accompanying reproducibility package (Online Resource 1) contains the complete validation grids, selected settings, per-seed observations, manifests, file hashes, prepared arrays, and executable source. Unit tests cover finite outputs, batched/individual agreement, deterministic replay, exact $\beta = 0$ reduction, source isolation, and agreement among NumPy, Torch, and CUDA implementations to within 2×10^{-4}.

6 Results

6.1 Concentrated and held-out workloads

Table 2 reports the four principal non-uniform workloads. On the controlled spatial field, QD^3 lowers NRMSE from 0.7305 to 0.5040, a 31.0% reduction. Write Trace does not improve Plain Delta, whereas the low-rank query trace is statistically indistinguishable from the full online query moment (paired difference -0.0007, 95% CI $[-0.0042, 0.0029]$). Thus, the spatial gain is neither a general consequence of adaptive rates nor dependent on storing a dense moment.

The two real-data-derived workloads reproduce the effect across text statistics and graph structure. QD^3 reduces error by 18.7% on token profiles and 32.1% on the module-import graph. Write frequency is informative here: Write Trace already obtains reductions of 16.8% and 27.8%. The remaining paired improvements of QD^3 are nevertheless consistent: the QD^3–Write Trace differences are -0.0144 (95% CI $[-0.0216, -0.0071]$; 14/16 wins) and -0.0391 (95% CI $[-0.0460, -0.0321]$; 16/16 wins), respectively. Reads therefore supply information beyond a strong frequency-based baseline.

On held-out WikiText-2 sequences, the effect is smaller but remains paired and repeatable. QD^3 reduces NRMSE by 2.62% relative to Plain Delta and by 1.42% relative to Write Trace; the paired difference is -0.0024 (95% CI $[-0.0034, -0.0014]$; 14/16 wins). Its mean differs from the full trace by only 8.6×10^{-5}, with a confidence interval spanning zero. Shuffling the order of the same query multiset changes QD^3's gain from 2.62% to 2.46%; frequency and key geometry, rather than a fragile local token ordering effect, account for most of the result.

Figure 2 places the central positive conditions beside their uniform controls. Under uniform spatial queries, QD^3 and Write Trace are indistinguishable (paired difference -0.00021, 95% CI $[-0.00084, 0.00042]$), and both are slightly worse than Plain Delta. In the uniform WikiText-2 workload, both adaptive low-rank methods improve Plain Delta by about 1.2%, but their difference is only 3.1×10^{-5} NRMSE. The same

Table 2 Blind-test NRMSE on non-uniform workloads, reported as mean (sample standard deviation) over 16 paired seeds. Gain is the relative reduction of QD^3 from Plain Delta. Paired QD^3–Write Trace confidence intervals are reported in the surrounding text.

Workload	Plain	Write trace	QD^3	Full trace	Gain (%)
Spatial, concentrated	.7305 (.0450)	.7347 (.0534)	**.5040** (.0455)	.5047 (.0467)	31.01
Source-token profiles	.7718 (.0391)	.6421 (.0243)	**.6278** (.0164)	.6143 (.0096)	18.66
Module-import graph	.8923 (.0219)	.6447 (.0282)	**.6056** (.0190)	.6035 (.0171)	32.13
WikiText-2, ordered	.1719 (.0157)	.1698 (.0160)	**.1674** (.0147)	.1673 (.0148)	2.62

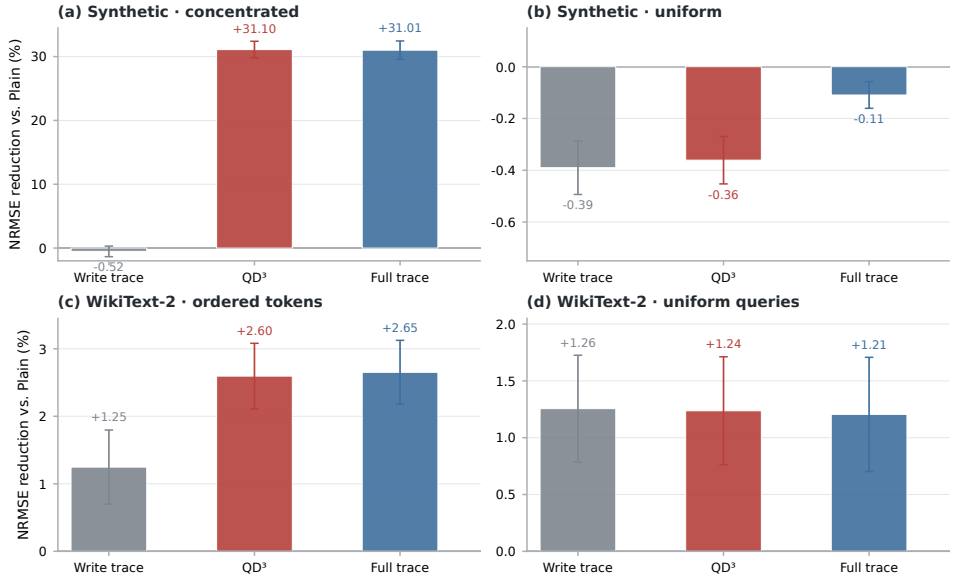

Fig. 2 Blind-test NRMSE reduction relative to Plain Delta (mean and paired 95% confidence interval; 16 seeds; each panel has its own labeled vertical scale). On the controlled spatial field, query-derived traces improve concentrated-query recall while the matched write trace does not; their distinction disappears under uniform queries. On WikiText-2, contiguous held-out queries give QD^3 a 2.62% reduction, whereas QD^3 and Write Trace are nearly identical under uniform queries. Lower NRMSE is better.

source equality appears on uniform token and import workloads: the paired QD^3–Write Trace confidence intervals both include zero. The relevant boundary is therefore precise: when queries carry no extra allocation information, the query trace loses its advantage over its matched write-derived counterpart. Our claim concerns workload-aware allocation, not universal improvement under identical event distributions.

6.2 Where the gain comes from

The query-skew sweep in Fig. 3a tests the proposed cause continuously rather than only at two endpoints. At $\alpha = 0$, QD^3 has no measurable advantage over Write Trace. As read demand becomes more concentrated, the advantage grows monotonically to

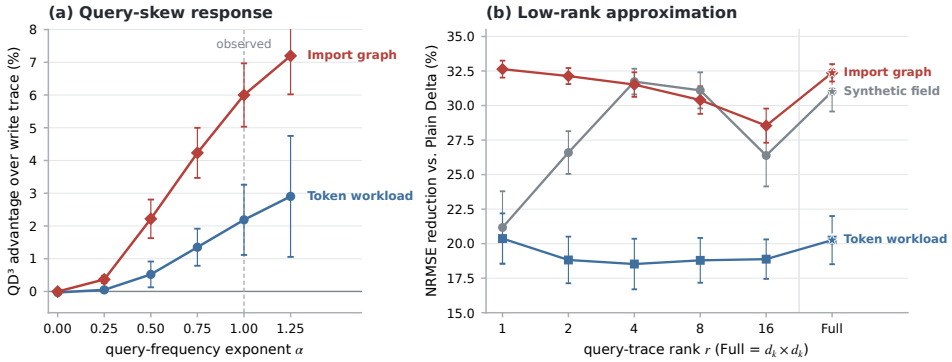

Fig. 3 Mechanism tests on blind seeds. (a) The additional NRMSE reduction of QD^3 over the estimator- and capacity-matched Write Trace grows with the query-frequency exponent and begins near zero for uniform queries. (b) A small adaptive query subspace recovers the full online second-moment result; optimal rank is task dependent and selected only on validation seeds.

Table 3 Ablation results as percentage NRMSE reduction from Plain Delta. "Random" is a fixed random rank-four subspace; "Dual" is the state-matched query–write control; "validation-selected QD^3" repeats the headline QD^3 configuration from Table 2, with hyperparameters chosen only on validation data.

Workload	Diag.	Last	Random	Dual	Val.-selected QD^3	Full
Spatial, concentrated	−0.82	16.09	−0.64	18.40	31.01	30.91
Source-token profiles	18.30	20.12	1.70	8.84	18.66	20.40
Module-import graph	27.50	30.71	3.00	10.06	32.13	32.37

2.90% on token profiles and 7.20% on the import graph at $\alpha = 1.25$. This is the pattern expected if the second state measures workload mismatch: changing the query distribution changes the value of query history while leaving the write population fixed.

Figure 3b and Table 3 examine representation. A fixed random rank-four subspace is ineffective, so state count alone is not sufficient. A diagonal trace works when demand is aligned with input coordinates but fails on the rotated spatial features. The most recent query captures some locality, yet accumulated geometry is consistently stronger on the spatial task. The state-matched Dual Trace improves on Plain Delta under all three non-uniform workloads, but its gains of 18.40%, 8.84%, and 10.06% remain below the validation-selected QD^3 configurations used for the headline comparisons. Dividing a fixed trace budget between demand and exposure is therefore less effective here than dedicating it to query history. Learned ranks between one and four recover essentially all of the full-trace benefit on the two real-data-derived workloads. Rank sensitivity is task dependent, and all headline configurations are selected only on validation data.

6.3 Changing demand

The moving-hotspot experiment changes the query center halfway through the stream and evaluates the final region. Plain Delta obtains 0.7528 ± 0.0241 NRMSE and Write Trace 0.7630 ± 0.0346. QD^3 reaches 0.5600 ± 0.0390, a 25.6% reduction from Plain Delta. It also outperforms the dense full trace (0.5814 ± 0.0484) by 0.0214 NRMSE, with paired 95% CI $[-0.0329, -0.0100]$. The low-rank bottleneck and exponential energy decay together favor dominant recent directions, which is useful when demand moves. We interpret this result as evidence of online tracking, not as a general guarantee that low rank will always outperform a full moment.

7 Discussion

7.1 Design interpretation

The results identify more than an adaptive-rate effect because Write Trace supplies that matched comparison. They support the intended information flow: reads estimate demand, the estimate allocates correction to later writes, and the value state remains responsible for association. Equation (4) specifies the population the memory should serve, while Eq. (6) supplies an online geometric proxy using addresses alone. This proxy is not a density-ratio estimator or a replacement for the weighted-linear solution; its contribution is to make read-aware allocation causal, inspectable, and maintainable in $O(d_k r)$ state. Uniform controls complete the interpretation: matched read and write traces agree when their information is interchangeable and separate when access concentrates elsewhere in key space.

7.2 Application value and deployment scope

The appropriate deployment setting has three observable properties: capacity remains fixed while associations continue to arrive, reads occur before some later writes, and access is more concentrated than the stored population. The closest use case is a linear-attention or fast-weight sequence layer, whose recurrent matrix is already a continuous superposition with no token list to evict. Other natural cases are persistent agent or retrieval memory, where repeated requests reveal semantic demand, and robot or sensor maps, where revisited regions should receive more accurate later updates. In all three cases the access summary remains $O(d_k r)$ rather than growing into a log.

These conditions can be checked before deployment from capacity pressure and access statistics. A build-once store has no causal read signal, while uniform access offers no source-specific allocation advantage. The next step is therefore not to broaden the claim, but to integrate the operator into chunkwise training kernels and measure downstream language, retrieval, and embodied-memory behavior.

8 Conclusion

Query Dual Dynamics Delta turns past access into a compact causal signal for future storage. A low-rank trace learns query directions and energy, a normalized and clipped heat modulates the ordinary residual Delta write, and $\beta = 0$ exactly recovers the

base rule. Across controlled and held-out workloads, the query source improves recall when demand is concentrated and consistently exceeds an estimator-matched write source. The central design lesson is consequently precise: in a continuously updated fixed-capacity memory, reading can guide later write allocation without retaining an ever-growing access history.

Statements and Declarations

Funding. No funding was received for conducting this study.
Competing interests. On behalf of all authors, the corresponding author states that there is no conflict of interest.
Author contributions. The first author conceived the method, designed and conducted the experiments, analyzed the results, and drafted the manuscript. The second author reviewed and revised the manuscript. Both authors read and approved the final manuscript.
Ethics approval. Not applicable. The study uses public text and locally derived software-structure statistics and does not involve human participants or animals.
Consent to participate. Not applicable.
Consent for publication. Not applicable.
Data availability. WikiText-2 is publicly available. The exact prepared arrays used for all reported experiments are included in the reproducibility package accompanying this submission.
Materials availability. The prepared workload arrays and complete per-seed result tables are included in the reproducibility package.
Code availability. Source code for the memory operators, validation and test evaluation, statistical analysis, and implementation checks is included with this submission and will be deposited in a public repository upon acceptance.

References

[1] Ba, J., Hinton, G.E., Mnih, V., Leibo, J.Z., Ionescu, C.: Using fast weights to attend to the recent past. In: Advances in Neural Information Processing Systems, vol. 29, pp. 4331–4339 (2016). https://proceedings.neurips.cc/paper/2016/hash/9f44e956e3a2b7b5598c625fcc802c36-Abstract.html

[2] Huang, D., Niles-Weed, J., Ward, R.: Streaming k-PCA: Efficient guarantees for Oja's algorithm, beyond rank-one updates. In: Proceedings of Thirty Fourth Conference on Learning Theory. Proceedings of Machine Learning Research, vol. 134, pp. 2463–2498 (2021). https://proceedings.mlr.press/v134/huang21a.html

[3] Katharopoulos, A., Vyas, A., Pappas, N., Fleuret, F.: Transformers are RNNs: Fast autoregressive transformers with linear attention. In: Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 5156–5165 (2020). https://proceedings.mlr.press/v119/katharopoulos20a.html

[4] Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., Chen, D.: SnapKV: LLM knows what you are looking for before generation. In: Advances in Neural Information Processing Systems, vol. 37 (2024). https://doi.org/10.52202/079017-0722

[5] Merity, S., Xiong, C., Bradbury, J., Socher, R.: Pointer sentinel mixture models. In: The Fifth International Conference on Learning Representations (2017). https://openreview.net/forum?id=Byj72udxe

[6] Oja, E.: Simplified neuron model as a principal component analyzer. Journal of Mathematical Biology **15**, 267–273 (1982) https://doi.org/10.1007/BF00275687

[7] Park, S., Kim, S., Park, N.: Q-Delta: Beyond key-value associative state evolution. In: Proceedings of the 43rd International Conference on Machine Learning (2026). https://icml.cc/Downloads/2026

[8] Schmidhuber, J.: Learning to control fast-weight memories: An alternative to dynamic recurrent networks. Neural Computation **4**(1), 131–139 (1992) https://doi.org/10.1162/neco.1992.4.1.131

[9] Siems, J., Carstensen, T., Zela, A., Hutter, F., Pontil, M., Grazzi, R.: DeltaProduct: Improving state-tracking in linear RNNs via householder products. In: Advances in Neural Information Processing Systems, vol. 38 (2025). https://doi.org/10.52202/085713-5141

[10] Schlag, I., Irie, K., Schmidhuber, J.: Linear transformers are secretly fast weight programmers. In: Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 9355–9366 (2021). https://proceedings.mlr.press/v139/schlag21a.html

[11] Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P.V., Kawanabe, M.: Direct importance estimation with model selection and its application to covariate shift adaptation. In: Advances in Neural Information Processing Systems, vol. 20 (2007)

[12] Widrow, B., Marcian E. Hoff, J.: Adaptive switching circuits. In: IRE WESCON Convention Record, vol. 4, pp. 96–104 (1960). https://isl.stanford.edu/~widrow/papers/c1960adaptiveswitching.pdf

[13] Yang, S., Kautz, J., Hatamizadeh, A.: Gated delta networks: Improving Mamba2 with delta rule. In: The Thirteenth International Conference on Learning Representations (2025). https://openreview.net/forum?id=r8H7xhYPwz

[14] Yang, S., Wang, B., Zhang, Y., Shen, Y., Kim, Y.: Parallelizing linear transformers with the delta rule over sequence length. In: Advances in Neural Information Processing Systems, vol. 37 (2024). https://doi.org/10.52202/079017-3668

[15] Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., Wang, Z., Chen, B.: H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In: Advances in Neural Information Processing Systems, vol. 36 (2023). https://doi.org/10.52202/075280-1506

[16] Zhu, Y., Yang, D.H., Amiri, M.M., Murugesan, K., Pedapati, T., Chen, P.-Y.: OjaKV: Context-aware online low-rank KV cache compression with Oja's rule. In: Findings of the Association for Computational Linguistics: ACL 2026, pp. 10161–10178. Association for Computational Linguistics, San Diego, California, United States (2026). https://doi.org/10.18653/v1/2026.findings-acl.494 . https://aclanthology.org/2026.findings-acl.494/