SHARED PHASE AND RETENTION CONTROL FOR EFFICIENT ADAPTIVE SPECTRAL RECURRENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

As new evidence arrives, a sequence model must update both what it remembers and how that memory influences subsequent predictions. While Transformers incur computation and cache costs that scale with context length, fixed-state recurrent models offer constant-memory inference. However, linear and spectral recurrences traditionally rely on static, time-invariant transitions: fixed dynamics can append new content, but cannot dynamically revise how stored representations decay, rotate, or evolve. Although recent selective architectures introduce input-dependent transitions, they typically assign independent controls to every memory mode, coupling control cost directly to state capacity. We show that high-dimensional spectral memory does not require high-dimensional control, and introduce Shared Phase and Retention Control for Efficient Adaptive Spectral Recurrence (SPARC). SPARC employs just two input-dependent scalar signals to coordinate memory retention and phase rotation across heterogeneous complex modes, while preserving mode-specific baseline timescales and frequencies. Its diagonal, affine recurrence supports parallel associative scans for sequence-level BPTT, as well as exact structured Real-Time Recurrent Learning (RTRL) for online credit assignment. Across partially observable continuous control, POPGym, and sequence classification, SPARC achieves a 9.09% relative return improvement on Walker-P and a 1.36% relative accuracy gain on FordA over second-best methods. On an NVIDIA Blackwell GPU, our implementation reduces recurrent-mixer training latency by 18.2%–34.2% in fixed-token workloads and accelerates scans by 3.1–$4.7\times$ over an optimized RG-LRU baseline. These results show that two shared control signals can efficiently govern adaptive spectral memory across both online and full-sequence learning settings.

1 INTRODUCTION

A system receiving continuous inputs must not only preserve the past, but also revise how past information affects future computation. Memory therefore involves both writing new information and transforming existing information, including its retention and evolution. Transformers address this need by retaining token-level representations and establishing content-dependent interactions through self-attention, enabling flexible access to the past and highly parallel training (Vaswani et al., 2017). However, accessing historical representations incurs computational and memory costs that grow with context length. For fixed model width, dense global attention requires quadratic computation in sequence length, while the full key–value (KV) cache for autoregressive inference grows linearly (De et al., 2024; Botev et al., 2024). Fixed-state recurrent models offer an alternative: compressing history into a continuously updated hidden state provides inference-state storage independent of history length and linear-time stream processing (Gu et al., 2020). Recent developments, including the structured state-space formulation of S4 (Gu et al., 2021), the parallel scan formulation of S5 (Smith et al., 2022), and the diagonal spectral parameterization, initialization, and normalization of LRU (Orvieto et al., 2023), establish a practical foundation for efficient long-range sequence modeling. Building on Real-Time Recurrent Learning (RTRL) (Williams & Zipser, 1989), RTU exploits structured sensitivities in spectral recurrence for online credit assignment, allowing recurrent memory to continue learning during interaction in partially observable environments (Elelimy et al., 2024).

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

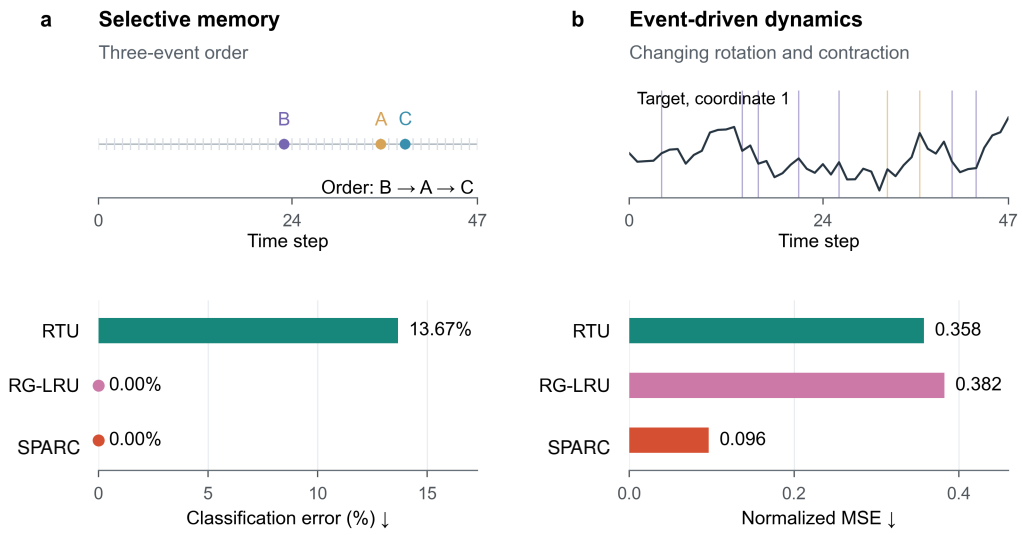

Figure 1: **Two complementary memory requirements.** (a) Recall relevant events in order despite distractors. (b) Track event-driven changes in contraction and rotation. Task details are in Appendix C.3.

Efficiently compressing history, however, does not by itself explain how new evidence should transform stored information. In a linear recurrence with a fixed state transition, a write determined only by the current layer input can add information, but cannot generally change how arbitrary existing states evolve. When an observation indicates a memory reset, a regime change, or a switch in latent dynamics, adding new content may therefore be insufficient to transform stored information as required. In a complex-valued spectral representation, this transformation has two complementary components: transition magnitude determines how strongly information is retained, whereas phase determines how the memory state rotates in the complex plane. Input-dependent gates have long been used to regulate recurrent memory (Gers et al., 2000; Cho et al., 2014). RG-LRU modulates retention and writing through input-dependent gates (De et al., 2024); Mamba makes state-space parameters content-selective (Gu & Dao, 2023); and GateLoop uses data-controlled complex transitions whose magnitude and phase vary with the input (Katsch, 2023). These approaches allow recurrent memory to accumulate information and adapt how it propagates, raising a design question about the relationship between state capacity and the dimensionality of dynamic control.

Figure 1 contrasts two memory requirements. In event-order recall, adaptive retention lets RG-LRU and SPARC achieve zero error, whereas RTU reaches 13.7%. Tracking event-driven contraction and rotation additionally requires adapting state dynamics: SPARC controls both retention and phase and achieves NMSE 0.096, compared with 0.358 for RTU and 0.382 for RG-LRU.

Generating independent controls for every mode makes control cost grow with memory size. We therefore ask: **does a high-dimensional memory require equally high-dimensional control?**

We introduce *SPARC: Shared Phase and Retention Control for Efficient Adaptive Spectral Recurrence*. SPARC uses shared phase and retention control to adapt heterogeneous spectral memory modes. Two signals from the current input coordinate state evolution, while each mode retains its learned decay, frequency, input projection, and write response. The same signals also regulate new input through a modal write gate and retention-dependent normalization.

SPARC remains diagonal and affine in the previous state, enabling associative scans for full-sequence BPTT (Martin & Cundy, 2018; Smith et al., 2022). Its diagonal state dependence and shared controllers also permit exact structured RTRL for recurrent-layer sensitivities at fixed parameters (Williams & Zipser, 1989; Elelimy et al., 2024).

Our main contributions are as follows:

1. **Shared retention–phase control.** We introduce a spectral recurrence in which two input-dependent signals are shared across spectral modes to jointly modulate memory retention and phase evolution, while preserving mode-specific baseline timescales and frequencies.
2. **Exact structured RTRL.** We derive exact RTRL updates that exploit the diagonal state Jacobian, per-mode parameter dependencies, and low-dimensional shared controllers. This structure avoids the full state–parameter influence matrix required by general dense recurrence while preserving complete recurrent-layer temporal sensitivities at fixed parameters and supplied layer inputs.
3. **Efficient GPU implementations.** We develop optimized GPU implementations of RG-LRU (De et al., 2024) and SPARC. Shared control reduces coefficient-generation and backpropagation costs, yielding 18.2%–34.2% lower recurrent-mixer training latency in fixed-token workloads and 3.1–$4.7\times$ faster scans than our optimized RG-LRU baseline on a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU.
4. **Evaluation across tasks and training regimes.** We evaluate SPARC on partially observable continuous control, POPGym, and sequence classification. SPARC improves Walker-P return and FordA accuracy by 9.09% and 1.36%, respectively, relative to the second-best methods.

2 Related Work

2.1 Spectral Recurrence and Parallel Computation

Structured recurrence combines long-range memory with efficient computation. S4 uses structured state matrices (Gu et al., 2021), S5 uses multi-input, multi-output layers and parallel scans (Smith et al., 2022), and LRU uses stable diagonal complex dynamics (Orvieto et al., 2023). LinOSS builds oscillatory memory from discretized harmonic oscillators (Rusch & Rus, 2025), while RTU exploits spectral sensitivities for online reinforcement learning (Elelimy et al., 2024). Affine composition enables parallel prefix scans (Blelloch, 1990; Martin & Cundy, 2018), but practical efficiency also depends on coefficient generation, memory access, and differentiation. Mamba addresses these costs through selective scans, fusion, and recomputation (Gu & Dao, 2023); Mamba-2 uses state space duality to organize blockwise computation (Dao & Gu, 2024). SPARC preserves scan-compatible spectral recurrence while reducing dynamic-control overhead through two shared signals. With controls disabled, linear content, and matched write parameters, it recovers the recurrent forms of LRU and linear RTU.

2.2 Input-Dependent Recurrent Dynamics

Input-dependent recurrent models differ in the dynamics they control and how those controls are parameterized. RG-LRU uses real-valued gates for retention and writing (De et al., 2024), while GateLoop uses complex transitions with input-dependent magnitude and phase (Katsch, 2023). Mamba introduces selective dynamics through input-dependent discretization and input–output projections, including structured, low-rank selection (Gu & Dao, 2023). Mamba-3 uses complex-valued updates whose rotations admit an interpretation as data-dependent rotary embeddings (Lahoti et al., 2026). SPARC jointly adapts retention and phase through two signals shared across spectral modes, each retaining its learned baseline decay and frequency. This separates dynamic-control dimensionality from recurrent-state capacity.

2.3 Online Learning for Recurrent Models

Recurrent training balances temporal credit assignment against computation and memory. BPTT differentiates through an unrolled graph; truncated BPTT limits credit assignment to a finite window (Werbos, 1990). RTRL propagates state–parameter sensitivities forward but is prohibitively expensive for dense recurrent networks (Williams & Zipser, 1989). Approximations include UORO's unbiased rank-one estimator (Tallec & Ollivier, 2017), KF-RTRL's stochastic Kronecker factors (Mujika et al., 2018), SnAp's finite-step influence sparsity (Menick et al., 2020), and e-prop's local eligibility traces with learning signals (Bellec et al., 2020). Structured architectures instead restrict state dependencies to make sensitivities more efficient to represent (Zucchet et al., 2023;

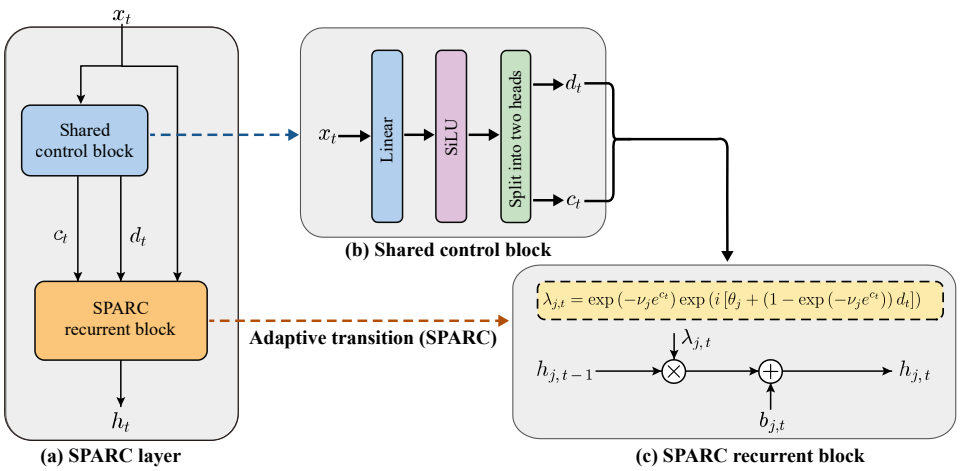

Figure 2: **Overview of SPARC.** (a) The layer input supplies a shared control block and the recurrent block, which produces the updated state. (b) Two scalar control heads coordinate retention and phase across spectral modes. The Linear–SiLU pathway illustrates a possible controller architecture; Eq. (31) specifies the tanh-projection parameterization used in this paper. (c) Mode j multiplies its previous state by an input-dependent complex transition and adds the current write contribution.

Elelimy et al., 2024). SPARC's diagonal state Jacobian, mode-local parameters, and shared controllers permit exact structured RTRL without temporal truncation or stochastic approximation, at fixed parameters and supplied layer inputs.

3 Method

SPARC builds on a diagonal spectral recurrence whose modes retain their own decay rates and frequencies, while two shared input-dependent signals modulate retention and phase. Figure 2 traces the input flow through the shared controls and mode-wise state update. This structure connects to fixed spectral recurrence while preserving the affine form needed for parallel sequence computation and exact structured RTRL.

3.1 Spectral Recurrence Preliminaries

Linear recurrent models compress history into a fixed-dimensional state. Consider

$$\mathbf{h}_t = A\mathbf{h}_{t-1} + B\mathbf{x}_t, \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^D$, $\mathbf{h}_t \in \mathbb{C}^H$, $A \in \mathbb{C}^{H\times H}$, and $B \in \mathbb{C}^{H\times D}$. The transition propagates existing information; the input projection writes new content. For diagonalizable $A = V\Lambda V^{-1}$, express the state in its eigenbasis, retaining the notation $\mathbf{h}_t$, and let $\widetilde{B} = V^{-1}B$. Then

$$\mathbf{h}_t = \boldsymbol{\lambda} \odot \mathbf{h}_{t-1} + \widetilde{B}\mathbf{x}_t. \tag{2}$$

In this basis, each mode evolves independently through a complex transition. A stable transition can be written as

$$\lambda_j = \exp(-\nu_j + i\theta_j), \qquad \nu_j > 0, \tag{3}$$

where $e^{-\nu_j}$ controls retention and θ_j controls rotation. After k steps, a stored contribution is multiplied by $e^{-k\nu_j}e^{ik\theta_j}$: decay determines its remaining magnitude, while phase determines its orientation and hence its projection onto a readout. Distinct modal decays and frequencies therefore provide heterogeneous memory responses, as in LRU and the linear RTU core (Orvieto et al., 2023; Elelimy et al., 2024).

A fixed spectrum applies this same retention and rotation at every step. To let the current input modify the evolution of stored information, consider

$$h_{j,t} = \lambda_{j,t}(\mathbf{x}_t)h_{j,t-1} + b_{j,t}(\mathbf{x}_t). \tag{4}$$

The distinction between transition and writing is seen by applying the same input to two states produced by different histories:

$$h_{j,t} - h'_{j,t} = \lambda_{j,t}(\mathbf{x}_t)(h_{j,t-1} - h'_{j,t-1}). \tag{5}$$

The input-only write cancels. It can add information, but cannot change how differences between historical states are transformed. This is why input-dependent writing alone cannot generally replace transition control in the same state representation. Appendix A gives the precise condition and the effect of accumulated phase on historical contributions.

3.2 Shared Adaptive Spectral Dynamics

SPARC computes two shared scalar controls, c_t and d_t, from affine input projections followed by scaled $\tanh$ activations. Appendix C.1 gives their parameterization and ranges. Each mode retains its learned decay $\nu_j > 0$ and frequency θ_j, with transition

$$\rho_{j,t} = \exp(-\nu_j e^{c_t}), \qquad \lambda_{j,t} = \rho_{j,t} \exp\big(i[\theta_j + (1 - \rho_{j,t})d_t]\big). \tag{6}$$

The retention control rescales modal decay rates together; the phase control adjusts their rotation. The factor $1 - \rho_{j,t}$ reduces phase adjustments for persistent modes. Every transition remains contractive because $|\lambda_{j,t}| = \rho_{j,t} < 1$.

The input is projected into real and imaginary content coordinates,

$$\mathbf{u}_t = \varphi(B^{\mathrm{Re}}\mathbf{x}_t) + i\varphi(B^{\mathrm{Im}}\mathbf{x}_t), \tag{7}$$

where both matrices have shape $H \times D$. The elementwise activation φ is $\tanh$ or the identity and acts only on the input. Writing combines this content with retention-dependent normalization and a sigmoid gate:

$$b_{j,t} = g_j \sqrt{1 - \rho_{j,t}^2}\, q_{j,t}\, u_{j,t}. \tag{8}$$

Here $g_j > 0$ is a learned modal gain, and $q_{j,t}$ is a sigmoid of a mode-specific linear combination of the two controls. Appendix B gives their full definitions and derivatives.

The state update is

$$\mathbf{h}_t = \boldsymbol{\lambda}_t \odot \mathbf{h}_{t-1} + \mathbf{b}_t. \tag{9}$$

The coefficients depend only on the current input and parameters, preserving affine dependence on the previous state. Sharing the controls leaves modal decays, frequencies, input projections, and write responses distinct.

Connection to fixed spectral recurrences. Zero-initialized controllers give $c_t = d_t = 0$, recovering the fixed spectrum in Eq. (3). With linear content and matched effective write parameters, this has the recurrent form of LRU; the corresponding decay-dependent write scale also recovers the linear RTU core (Orvieto et al., 2023; Elelimy et al., 2024).

3.3 Parallel BPTT and Structured RTRL

Transition and write coefficients depend only on supplied inputs and parameters, so they can be generated before recurrent propagation. Represent each update by $F_t = (\boldsymbol{\lambda}_t, \mathbf{b}_t)$, with composition

$$F_2 \circ F_1 = \big(\boldsymbol{\lambda}_2 \odot \boldsymbol{\lambda}_1,\ \boldsymbol{\lambda}_2 \odot \mathbf{b}_1 + \mathbf{b}_2\big). \tag{10}$$

The operation is associative, with identity $(\mathbf{1}, \mathbf{0})$. A work-efficient prefix scan therefore evaluates the recurrent states with $O(TH)$ work and $O(\log T)$ temporal depth for a length-T sequence (Blelloch, 1990; Martin & Cundy, 2018).

Backpropagating through coefficient generation and the scan gives full-sequence BPTT without temporal truncation. For online interaction, the same cell runs one step at a time and propagates recurrent sensitivities forward.

For a real recurrent parameter block $\boldsymbol{\psi}$, define the complex row Jacobian $Z^{\boldsymbol{\psi}}_{j,t} = \partial h_{j,t}/\partial \boldsymbol{\psi}$. Holding supplied layer inputs fixed, the chain rule gives

$$Z^{\boldsymbol{\psi}}_{j,t} = \lambda_{j,t} Z^{\boldsymbol{\psi}}_{j,t-1} + h_{j,t-1} \frac{\partial \lambda_{j,t}}{\partial \boldsymbol{\psi}} + \frac{\partial b_{j,t}}{\partial \boldsymbol{\psi}}. \tag{11}$$

The first term propagates historical sensitivity; the remaining terms capture the current transition and write effects. With a parameter-independent initial state, $Z^{\psi}_{j,0} = 0$, these updates give exact recurrent-layer derivatives at fixed parameters. Mode-local parameters influence only their own mode, while shared controllers reuse instantaneous input Jacobians. Contracting the traces with state gradients from the readout yields recurrent-parameter gradients. Appendix B gives the local derivatives and qualifications for upstream encoders and stored-trace PPO.

Shared control uses $2(D + 1)$ controller parameters and $O(D)$ projection work per step, versus $2H(D + 1)$ and $O(HD)$ for independent dense controls per mode. Dense content projections leave the complete cell at $O(HD + H)$ work; structured RTRL uses the same order of work and storage. Appendix B.7 details the comparison, including RG-LRU's block-diagonal gates; Section 5 reports measured latency and memory use.

3.4 GPU-Efficient Implementation

Google DeepMind's released RecurrentGemma implementation provides a TPU-optimized Pallas scan and a reference PyTorch implementation (Botev et al., 2024). We implement GPU kernels for both RG-LRU and SPARC. SPARC divides time into chunks and modes into tiles. A prefix scan over local affine summaries supplies chunk entrance states, followed by parallel replay. The backward pass similarly uses a reverse scan for boundary cotangents, local adjoint replay, and gradient reduction.

Fusing content activation, write normalization, and gating, while recomputing modal transitions inside the scan, stores two $B \times T$ control arrays instead of a $B \times T \times H$ complex transition tensor. Modal control gradients are reduced within each tile; recurrent accumulation and scan summaries use FP32. Appendix G provides the pseudocode (Algorithm 1), memory layout, reset handling, and single-step kernel.

4 Experiments

We evaluate whether shared spectral control supports useful recurrent representations in three settings: partially observable continuous control, POPGym memory and decision-making tasks, and sequence classification. The reinforcement-learning experiments use an RTU recurrent actor–critic backbone with stored online sensitivities; classification uses an LRU residual sequence backbone and full-sequence BPTT. All benchmark tasks use three training seeds.

4.1 Experimental Setup

Tasks. Continuous control uses Ant-P, Walker-P, Hopper-P, and Cheetah-P (-P denotes masked observations), with $4{,}999{,}168$ environment steps per run. POPGym uses Autoencode, CountRecall, RepeatFirst, RepeatPrevious, Noisy Pendulum, and HigherLower, with $3{,}999{,}744$ steps per run (Morad et al., 2023), testing recall, memory updating, and noisy control.

Classification uses FordA, StarLightCurves, and UWaveGestureLibraryAll from UCR (Dau et al., 2019), plus permuted MNIST and sequential grayscale CIFAR-10. Inputs are scalar sequences without two-dimensional spatial modules.

Compared methods and state size. We compare SPARC with RTU, LRU, RG-LRU, GateLoop, Mamba-3, and GRU (Elelimy et al., 2024; Orvieto et al., 2023; De et al., 2024; Katsch, 2023; Lahoti et al., 2026; Cho et al., 2014), matching real recurrent-state budgets at 384 coordinates for continuous control and 128 for POPGym. Parameter counts remain architecture-dependent. SPARC uses $\tanh$ content; Appendix D gives architecture and initialization details.

Recurrent actor–critic backbone. Following RTU (Elelimy et al., 2024), RL models use a width-64 observation encoder and policy/value heads, resetting recurrent states and sensitivities at episode boundaries. Appendix D.2 specifies the backbone and action distributions.

PPO optimization. All methods use PPO (Schulman et al., 2017), rollout length 2048, and matched task-specific learning rates. SPARC, RTU, LRU, RG-LRU, and GateLoop use stored re-

Table 1: Partially observable continuous control.

Method	Ant-P	Walker-P	Hopper-P	Cheetah-P
RTU	3823.8	899.8	1266.7	2547.3
LRU	<u>4476.5</u>	655.5	**1363.4**	**2688.9**
RG-LRU	1953.7	865.5	1224.4	2539.5
GateLoop	4105.6	<u>911.9</u>	<u>1325.1</u>	2466.1
Mamba-3	1355.7	680.2	863.6	2019.9
GRU	1921.1	760.1	631.3	2174.1
SPARC	**4507.8**	**994.8**	1282.9	<u>2563.8</u>

Mean return across three training seeds. Higher is better. Best means are bold; second-best means are underlined.

current sensitivities; GRU and Mamba-3 use length-4 TBPTT. Appendices D.2 and B.6 detail optimization and stored-trace credit assignment.

Full-sequence backbone and optimization. Classification uses the LRU residual backbone (Orvieto et al., 2023): four width-64 layers and sequence-mean pooling, with 64 complex modes per SPARC layer. All methods use full-sequence BPTT for 20,000 updates at batch size 32; Appendix D.4 gives architecture and optimizer details.

Evaluation metrics. RL reports mean return over the final 100 rollouts; classification reports selected internal-validation accuracy (%). Scores average three training seeds. Preprocessing and splits are shared across methods. Appendices D.5 and D.6 detail checkpoint selection and aggregation.

4.2 Partially Observable Reinforcement Learning

SPARC achieves the highest mean returns on Ant-P (4507.8) and Walker-P (994.8) in Table 1. The Walker-P result improves on GateLoop's 911.9 by 9.09%, while Ant-P exceeds LRU by 0.70%. SPARC also ranks second on Cheetah-P with a return of 2563.8. The Walker-P phase ablation in Section 4.4 suggests that adapting rotation contributes to this performance alongside selective retention. Figure 5 in Appendix D.7 shows the learning curves.

On POPGym, SPARC obtains the highest mean returns on CountRecall, RepeatFirst, and Noisy Pendulum (Figure 3). The largest gain over the strongest baseline occurs on Noisy Pendulum, where return increases from GRU's 0.311 to 0.461. CountRecall improves by 0.050 over RTU, while RepeatFirst reaches 0.829 compared with GRU's 0.811. These gains cover both recall and memory updates under noisy observations. Table 6 in Appendix D.7 gives the complete results.

4.3 Sequence Modeling under Full-Sequence BPTT

Under full-sequence BPTT, SPARC achieves the highest mean accuracy on FordA and permuted MNIST and matches the highest displayed mean on StarLightCurves (Table 2). FordA accuracy rises from RTU and LRU's 95.38% to 96.68%, a relative gain of 1.36%. On permuted MNIST, SPARC reaches 96.87%, narrowly ahead of LRU's 96.82%, while StarLightCurves accuracy is 99.67%. With all methods trained in the same residual backbone, these results support SPARC's use in multilayer sequence models as well as recurrent policies.

4.4 Ablation Studies

Nine retrained variants isolate retention, phase, their coupling, and write components. Appendix E provides the selected comparisons (Table 7) and complete 16-task results.

Removing adaptive phase lowers Walker-P return from 994.79 to 837.70. Removing the phase clock lowers RepeatFirst return from 0.829 to 0.288 and sequential CIFAR-10 accuracy from 59.92% to 49.03%.

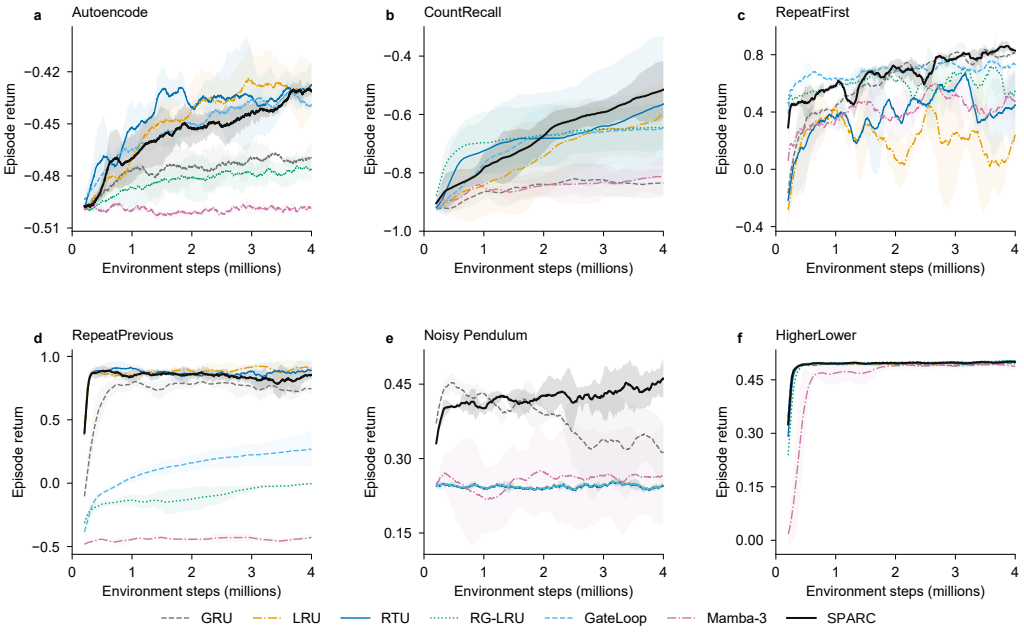

Figure 3: Learning curves for Autoencode, CountRecall, RepeatFirst, RepeatPrevious, Noisy Pendulum, and HigherLower. Curves and shading use the same three-seed averaging and sample-standard-deviation convention as Figure 5.

Table 2: Full-sequence classification.

Method	**FordA**	**SLC**	**UWave**	**pMNIST**	**CIFAR-10**
RTU	<u>95.38</u>	**99.67**	93.33	96.75	59.32
LRU	<u>95.38</u>	**99.67**	92.22	<u>96.82</u>	<u>60.11</u>
RG-LRU	92.89	**99.67**	**98.52**	85.48	51.01
GateLoop	91.97	99.00	91.85	62.31	52.46
Mamba-3	90.58	<u>99.33</u>	<u>98.15</u>	88.51	48.65
GRU	93.54	99.00	94.81	90.92	**62.69**
SPARC	**96.68**	**99.67**	96.30	**96.87**	59.92

Mean accuracy (%) across three seeds under full-sequence BPTT. SLC: StarLightCurves; pMNIST: permuted MNIST; CIFAR-10: sequential grayscale inputs. Evaluation follows Appendix D.5. Higher is better; best means are bold and second-best means underlined.

Neutralizing the write gate lowers RepeatFirst return to 0.663 while FordA remains at 96.68%; linear content lowers RepeatFirst to 0.123. Diagnostics show stronger gate variation and content saturation on RepeatFirst than on FordA (Appendix F).

5 Computational Efficiency

We compare SPARC with our optimized RG-LRU implementation on a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU (Figure 4). Both use matched workloads, BF16 inputs and outputs, and FP32 recurrent accumulation. We time scans after coefficient generation and complete recurrent-mixer forward–backward passes including coefficient generation. These timings exclude the surrounding model and optimizer; Appendix G.1 specifies measurement scope, workload shapes, and normalization.

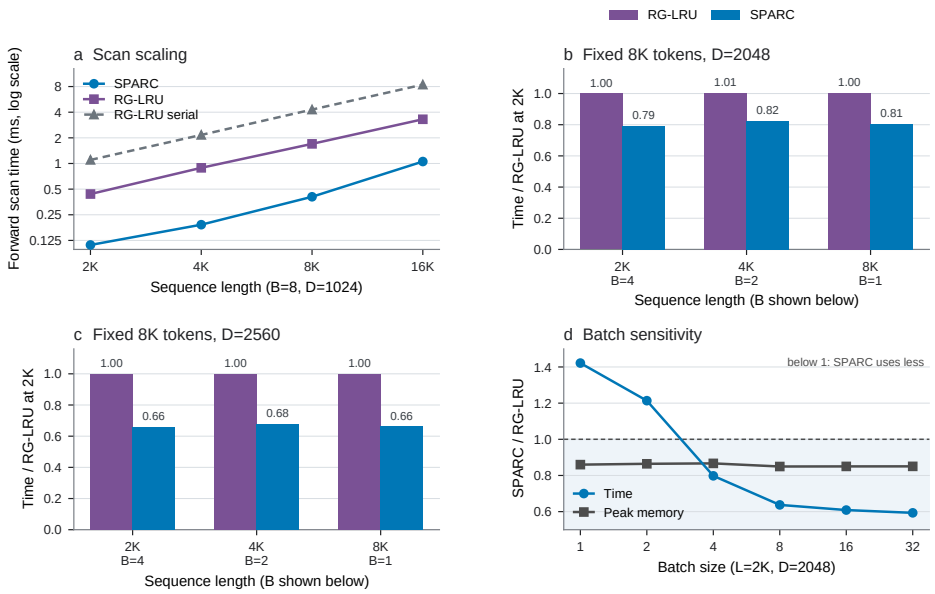

Figure 4: GPU efficiency on a single NVIDIA RTX PRO 6000 Blackwell Server Edition GPU. (a) Scan-only timing after coefficient generation. (b,c) Recurrent-mixer forward-plus-backward timing at fixed token count, for widths 2048 and 2560. (d) Runtime and peak allocated-memory ratios, SPARC divided by RG-LRU, as batch size varies. Ratios below one favor SPARC.

At batch size 8 and width 1024, SPARC's scan is 3.1–4.7× faster for sequence lengths 2048–16384. With 8192 tokens per batch, complete-mixer training latency falls by 18.2%–21.0% at width 2048 and 32.1%–34.2% at width 2560. These gains include the control pathway and its gradients, alongside the scan optimizations in Section 3.4. In the batch-size sweep, SPARC is 42.2% and 21.4% slower at batch sizes 1 and 2, respectively, but its runtime ratio is approximately 0.6 at batch sizes 8 and above. Peak allocated memory is roughly 13%–15% lower for most tested batch sizes. Lower memory use therefore does not guarantee lower latency in the smallest-batch workloads. Appendices G.2 and G.3 report sequential-reference comparisons and numerical checks.

6 Limitations and Future Work

Shared control coordinates changes across memory modes, which limits their ability to adapt independently. The ablations also show task-dependent gains and variation across training seeds, motivating further study of control grouping, initialization, and training stability. Our evaluation covers control and sequence classification at moderate model sizes. Extending SPARC to large language models, including hybrids with local attention as in Griffin (De et al., 2024), will require evaluating language-modeling quality, long-context behavior, and training efficiency at scale.

7 Conclusion

SPARC uses shared phase and retention control to adapt heterogeneous spectral memory modes. Two input-dependent signals adjust the evolution of a high-dimensional state while preserving distinct modal timescales and frequencies. The resulting affine recurrence supports parallel full-sequence BPTT and exact recurrent-layer sensitivities at fixed parameters. Across the evaluated tasks, SPARC improves Walker-P return by 9.09% and FordA accuracy by 1.36% over the strongest competing means. On a single Blackwell GPU, its optimized mixer reduces forward–backward latency by 18.2%–34.2% at a fixed token budget. These results support shared retention and phase control as an efficient mechanism for adapting recurrent memory.

AI USE STATEMENT

In this work, generative AI tools were used to assist in research execution and ideation, primarily for software implementation, code debugging, and preliminary technical discussions. Additionally, these tools were used to aid and polish the manuscript to improve clarity, grammar, and overall readability. All AI-generated code and drafted text were thoroughly inspected, tested, and verified by the authors, and generative AI was not used to produce core scientific claims. The authors take full responsibility for the entirety and integrity of this paper.

REFERENCES

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, 11(1):3625, 2020.

Guy E Blelloch. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University, November 1990.

Aleksandar Botev, Soham De, Samuel L Smith, Anushan Fernando, George-Cristian Muraru, Ruba Haroun, Leonard Berrada, Razvan Pascanu, Pier Giuseppe Sessa, Robert Dadashi, et al. Recurrentgemma: Moving past transformers for efficient open language models. *arXiv preprint arXiv:2404.07839*, 2024.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1724–1734, 2014.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Esraa Elelimy, Adam White, Michael Bowling, and Martha White. Real-time recurrent learning using trace units in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:17006–17043, 2024.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33: 1474–1487, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Tobias Katsch. Gateloop: Fully data-controlled linear recurrence for sequence modeling. *arXiv preprint arXiv:2311.01927*, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aakash Sunil Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, volume 2026, pp. 86173–86202, 2026.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *International Conference on Learning Representations*, 2018.

Jacob Menick, Erich Elsen, Utku Evci, Simon Osindero, Karen Simonyan, and Alex Graves. A practical sparse approximation for real time recurrent learning. *arXiv preprint arXiv:2006.07232*, 2020.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. Popgym: Benchmarking partially observable reinforcement learning. *arXiv preprint arXiv:2303.01859*, 2023.

Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning*, pp. 26670–26698. PMLR, 2023.

T Konstantin Rusch and Daniela Rus. Oscillatory state-space models. In *International Conference on Learning Representations*, volume 2025, pp. 22303–22322, 2025.

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

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

Corentin Tallec and Yann Ollivier. Unbiased online recurrent optimization. *arXiv preprint arXiv:1702.05043*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Nicolas Zucchet, Robert Meier, Simon Schug, Asier Mujika, and Joao Sacramento. Online learning of long-range dependencies. *Advances in Neural Information Processing Systems*, 36:10477–10493, 2023.

A Why Writing Cannot Generally Replace Transition Control

The distinction in Eq. (5) applies to a common state representation and to all reachable histories at a fixed current input.

Fix an input $\mathbf{x}$ and let $\mathcal{R}_{\mathbf{x}}$ be the set of previous states reachable immediately before that input. Suppose an input-dependent transition and a fixed transition agree after changing only the input-dependent write:

$$A(\mathbf{x})\mathbf{h} + \mathbf{b}(\mathbf{x}) = A_0\mathbf{h} + \mathbf{b}'(\mathbf{x}) \qquad \text{for every } \mathbf{h} \in \mathcal{R}_{\mathbf{x}}. \tag{12}$$

Then

$$[A(\mathbf{x}) - A_0]\mathbf{v} = 0 \quad \text{for every } \mathbf{v} \in \operatorname{span}(\mathcal{R}_{\mathbf{x}} - \mathcal{R}_{\mathbf{x}}). \tag{13}$$

In particular, if this difference span is the full state space, then $A(\mathbf{x}) = A_0$.

Apply Eq. (12) to two reachable states and subtract. The input-only writes cancel, giving $[A(\mathbf{x}) - A_0](\mathbf{h} - \mathbf{h}') = 0$. Linearity extends this equality to the span of all reachable-state differences. A matrix that vanishes on the full state space is zero.

Thus an input-only write can reproduce an adaptive transition on all reachable histories only when the transition difference vanishes on their difference span.

Accumulated phase. For a fixed input sequence, expanding Eq. (9) gives

$$h_{j,t} = \left(\prod_{\ell=1}^{t} \lambda_{j,\ell}\right) h_{j,0} + \sum_{k=1}^{t} \left(\prod_{\ell=k+1}^{t} \lambda_{j,\ell}\right) b_{j,k}. \tag{14}$$

An empty product is one. The contribution of one past write is therefore

$$\Delta h_{j,t}^{(k)} = \left(\prod_{\ell=k+1}^{t} \rho_{j,\ell}\right) e^{\,i[(t-k)\theta_j + \sum_{\ell=k+1}^{t}(1-\rho_{j,\ell})\kappa_p p_\ell]} b_{j,k}. \tag{15}$$

The product of retention factors determines how much of a past write remains, while accumulated phase rotates its contribution. A readout of the real state component therefore changes with phase even when the retained magnitude is fixed. Phase differences between past writes also determine whether their contributions reinforce or cancel.

The baseline term depends on elapsed time, while the additional term depends on intervening inputs. Retention-only control does not directly alter this rotation angle within a mode.

B Explicit Structured RTRL Derivatives

B.1 Parameter blocks and optimization semantics

The modal gain in Eq. (8) is $g_j = e^{\zeta_j}$. To preserve the experimental parameterization, its log coordinate is represented by

$$\zeta_j = \beta_j + \log 2 - \frac{1}{2}\log(1 - e^{-2\nu_j}), \qquad \nu_j = e^{a_j}. \tag{16}$$

The optimized gain and decay coordinates are β_j and a_j; ζ_j is derived from them. Substitution into Eq. (8) gives exactly

$$b_{j,t} = e^{\beta_j}\sqrt{\frac{1 - e^{-2e_{j,t}}}{1 - e^{-2\nu_j}}}\,[2\sigma(s_j p_t + v_j r_t)]\,u_{j,t}. \tag{17}$$

The independently optimized parameters comprise the real and imaginary content matrices, the retention and phase controller weights, and five parameter vectors with one entry per mode: log decay, log frequency, reference log gain, and the phase and retention coefficients of the write gate. The content matrices have shape $H \times D$, the controller vectors have length $D + 1$, and each modal vector has length H. Derivatives hold the supplied input fixed. The reference log gain is optimized independently of log decay; the compact gain in Eq. (16) includes the decay-dependent normalization.

Using the normalized controls from Appendix C.1, the sigmoid write gate is

$$q_{j,t} = \sigma(s_j p_t + v_j r_t), \qquad 0 < q_{j,t} < 1. \tag{18}$$

The local derivatives below are substituted directly into Eq. (11).

B.2 TRANSITION DERIVATIVES

Differentiating the expanded transition in Eq. (32) gives

$$\begin{aligned}
\frac{\partial \lambda_{j,t}}{\partial p_t} &= i\kappa_p(1-\rho_{j,t})\lambda_{j,t},\\
\frac{\partial \lambda_{j,t}}{\partial r_t} &= \kappa_r e_{j,t}(-1+i\rho_{j,t}\kappa_p p_t)\lambda_{j,t},\\
\frac{\partial \lambda_{j,t}}{\partial a_j} &= e_{j,t}(-1+i\rho_{j,t}\kappa_p p_t)\lambda_{j,t},\\
\frac{\partial \lambda_{j,t}}{\partial \vartheta_j} &= i\theta_j\lambda_{j,t}.
\end{aligned} \tag{19}$$

Changing log decay affects both the retained magnitude and the phase correction through the retention-coupled clock. The content projections, reference gain, and write-gate coefficients affect only the write pathway, so their direct derivatives of the transition are zero.

B.3 WRITE DERIVATIVES

The logarithm of the positive amplitude multiplying $q_{j,t}u_{j,t}$ is

$$\beta_j + \log 2 + \frac{1}{2}\log(1-e^{-2e_{j,t}}) - \frac{1}{2}\log(1-e^{-2\nu_j}). \tag{20}$$

Its derivative with respect to a_j therefore contains both the effective-decay term and the baseline-normalization term. Combining these with the sigmoid derivative gives

$$\begin{aligned}
\frac{\partial b_{j,t}}{\partial p_t} &= b_{j,t}(1-q_{j,t})s_j,\\
\frac{\partial b_{j,t}}{\partial r_t} &= b_{j,t}\left[\frac{\kappa_r e_{j,t}}{e^{2e_{j,t}}-1} + (1-q_{j,t})v_j\right],\\
\frac{\partial b_{j,t}}{\partial a_j} &= b_{j,t}\left[\frac{e_{j,t}}{e^{2e_{j,t}}-1} - \frac{\nu_j}{e^{2\nu_j}-1}\right],\\
\frac{\partial b_{j,t}}{\partial \beta_j} &= b_{j,t}, \qquad \frac{\partial b_{j,t}}{\partial \vartheta_j} = 0,\\
\frac{\partial b_{j,t}}{\partial s_j} &= b_{j,t}(1-q_{j,t})p_t, \qquad \frac{\partial b_{j,t}}{\partial v_j} = b_{j,t}(1-q_{j,t})r_t.
\end{aligned} \tag{21}$$

These formulas follow from product differentiation and remain valid when $u_{j,t} = 0$; they do not require taking the logarithm of complex content.

For the two content-projection rows,

$$\begin{aligned}
\frac{\partial b_{j,t}}{\partial B^{\mathrm{Re}}_{j,:}} &= e^{\zeta_j}\sqrt{1-\rho_{j,t}^2}\, q_{j,t}\varphi'((B^{\mathrm{Re}}\mathbf{x}_t)_j)\mathbf{x}_t^\top,\\
\frac{\partial b_{j,t}}{\partial B^{\mathrm{Im}}_{j,:}} &= ie^{\zeta_j}\sqrt{1-\rho_{j,t}^2}\, q_{j,t}\varphi'((B^{\mathrm{Im}}\mathbf{x}_t)_j)\mathbf{x}_t^\top.
\end{aligned} \tag{22}$$

Here $\varphi'(z) = 1-\tanh^2(z)$ for bounded content and $\varphi'(z) = 1$ for linear content. Other rows have zero direct effect on mode j.

B.4 CONTROLLER TRACES, LOSS GRADIENTS, AND RESETS

For example, the normalized phase-controller trace between episode resets is

$$Z^{\mathbf{w}_p}_{j,t} = \lambda_{j,t}Z^{\mathbf{w}_p}_{j,t-1} + \left(h_{j,t-1}\frac{\partial \lambda_{j,t}}{\partial p_t} + \frac{\partial b_{j,t}}{\partial p_t}\right)(1-p_t^2)\bar{\mathbf{x}}_t^\top, \tag{23}$$

where $\partial\lambda_{j,t}/\partial p_t = i\kappa_p(1-\rho_{j,t})\lambda_{j,t}$. The radial-controller trace follows the same chain rule with r_t and $\mathbf{w}_r$.

The shared input-controller Jacobians are

$$\frac{\partial r_t}{\partial \mathbf{w}_r} = (1 - r_t^2)\bar{\mathbf{x}}_t^\top, \qquad \frac{\partial p_t}{\partial \mathbf{w}_p} = (1 - p_t^2)\bar{\mathbf{x}}_t^\top. \tag{24}$$

At episode boundaries, let $m_t \in \{0, 1\}$ indicate whether the previous state belongs to the current episode. Resetting to zero gives

$$\mathbf{h}_t = (m_t \boldsymbol{\lambda}_t) \odot \mathbf{h}_{t-1} + \mathbf{b}_t. \tag{25}$$

The mask preserves affine composition and resets accumulated sensitivities. The controller derivative includes both transition and write pathways, giving

$$\begin{aligned} Z_{j,t}^{\mathbf{w}_r} &= m_t \lambda_{j,t} Z_{j,t-1}^{\mathbf{w}_r} + \left(m_t h_{j,t-1} \frac{\partial \lambda_{j,t}}{\partial r_t} + \frac{\partial b_{j,t}}{\partial r_t} \right) (1 - r_t^2)\bar{\mathbf{x}}_t^\top, \\ Z_{j,t}^{\mathbf{w}_p} &= m_t \lambda_{j,t} Z_{j,t-1}^{\mathbf{w}_p} + \left(m_t h_{j,t-1} \frac{\partial \lambda_{j,t}}{\partial p_t} + \frac{\partial b_{j,t}}{\partial p_t} \right) (1 - p_t^2)\bar{\mathbf{x}}_t^\top. \end{aligned} \tag{26}$$

For any parameter block, the corresponding reset-aware rule is

$$Z_{j,t}^{\boldsymbol{\psi}} = m_t \lambda_{j,t} Z_{j,t-1}^{\boldsymbol{\psi}} + m_t h_{j,t-1} \partial_{\boldsymbol{\psi}} \lambda_{j,t} + \partial_{\boldsymbol{\psi}} b_{j,t}. \tag{27}$$

The mask is supplied by the environment and is not differentiated. At a reset, historical state and sensitivity vanish, while the current write can still create a nonzero parameter derivative. Gradients are obtained by the real-coordinate contraction in Eq. (28), and storage is exactly the trace count in Eq. (29) for this explicit representation.

At zero initialization, $r_t = p_t = 0$ and $s_j = v_j = 0$. Direct derivatives of the modal response coefficients initially vanish because they multiply zero controls. The controllers can nevertheless receive gradients through the transition and, for the radial controller, through retention-dependent writing. The write-gate pathway is therefore initially neutral without permanently disabling adaptation.

B.5 Loss contraction and trace storage

For a real instantaneous loss $\mathcal{L}_t$, ordinary backpropagation through the current readout supplies the real-coordinate state gradients. Their contraction with the traces is

$$\frac{\partial \mathcal{L}_t}{\partial \boldsymbol{\psi}} = \sum_{j=1}^{H} \Big(\frac{\partial \mathcal{L}_t}{\partial \operatorname{Re} h_{j,t}} \operatorname{Re} Z_{j,t}^{\boldsymbol{\psi}} + \frac{\partial \mathcal{L}_t}{\partial \operatorname{Im} h_{j,t}} \operatorname{Im} Z_{j,t}^{\boldsymbol{\psi}} \Big) + \left. \frac{\partial \mathcal{L}_t}{\partial \boldsymbol{\psi}} \right|_{\text{direct}}. \tag{28}$$

The final term includes parameter dependence outside the recurrent state. This expression avoids reliance on a particular complex-gradient convention.

The two content matrices require $2HD$ complex trace entries, the five modal parameter vectors require $5H$, and the two shared controllers require $2H(D + 1)$. The total is

$$2HD + 5H + 2H(D + 1) = 4HD + 7H, \tag{29}$$

with $O(HD+H)$ update work per step. Shared control therefore does not mean that only two traces are stored: each mode has its own accumulated response to the controller parameters.

B.6 Exactness scope and stored-trace PPO

For fixed recurrent parameters and supplied inputs, Eq. (11) is the full recurrent-layer chain rule without temporal truncation or stochastic approximation. If an upstream-only parameter block $\boldsymbol{\omega}$ changes the input, its total derivative additionally satisfies

$$\frac{dh_{j,t}}{d\boldsymbol{\omega}} = \lambda_{j,t} \frac{dh_{j,t-1}}{d\boldsymbol{\omega}} + \left(h_{j,t-1} \frac{\partial \lambda_{j,t}}{\partial \mathbf{x}_t} + \frac{\partial b_{j,t}}{\partial \mathbf{x}_t} \right) \frac{d\mathbf{x}_t}{d\boldsymbol{\omega}}. \tag{30}$$

Omitting historical upstream sensitivities does not yield exact end-to-end temporal gradients for the encoder or an entire stack of recurrent layers.

The RL implementation uses local encoder gradients and stores recurrent sensitivities during rollout collection. PPO reuses those stored quantities rather than replaying the full history after every minibatch update, following the stored-trace approach studied with RTU (Elelimy et al., 2024). They therefore retain collection-time semantics instead of becoming exact sensitivities under each updated parameter vector. If traces are carried across parameter changes, they also inherit the usual stale-trace qualification. Storing all rollout traces uses $O(THD)$ memory even though the running online trace state is $O(HD + H)$.

B.7 Control-path complexity

SPARC separates dynamic-control dimensionality from the number of modes. For input width D and H complex modes, two independent dense controls per mode require $2H(D + 1)$ parameters and $O(HD)$ projection work per step; SPARC uses $2(D + 1)$ parameters and $O(D)$ work. Its two outputs per step require $2T$ stored values over a length-T sequence, versus $2HT$ for per-mode controls. Sharing also reduces backward projection work.

Modal coefficient generation remains $O(H)$ and dense content projection $O(HD)$, so the complete cell still costs $O(HD + H)$ per step. Exact structured RTRL has the same work and storage order as other diagonal cores with local parameter dependencies (Elelimy et al., 2024). Sharing therefore reduces the control pathway within the same overall asymptotic order. Practical gains depend on the competing controller: RG-LRU's block-diagonal gates already reduce projection cost (De et al., 2024). Section 5 measures latency and memory use, including coefficient generation.

C Component Analysis and Controlled Diagnostics

This section explains the components and defines the corresponding interventions. Retrained ablation results are reported in Appendix E and checkpoint diagnostics in Appendix F.

C.1 Control ranges and retention-coupled phase

The two scalar controls are computed from affine projections of the current layer input. With $\bar{\mathbf{x}}_t = [\mathbf{x}_t; 1]$ and $\mathbf{w}_r, \mathbf{w}_p \in \mathbb{R}^{D+1}$,

$$\begin{aligned} r_t &= \tanh\big(\mathbf{w}_r^\top \bar{\mathbf{x}}_t\big), \\ p_t &= \tanh\big(\mathbf{w}_p^\top \bar{\mathbf{x}}_t\big). \end{aligned} \tag{31}$$

The scaled controls in Section 3.2 are $c_t = \kappa_r r_t$ and $d_t = \kappa_p p_t$. Each mode learns log decay a_j and log frequency ϑ_j, giving $\nu_j = \exp(a_j) > 0$ and $\theta_j = \exp(\vartheta_j) > 0$. The expanded transition used in the derivative formulas is

$$\begin{aligned} e_{j,t} &= \nu_j \exp(\kappa_r r_t), \\ \rho_{j,t} &= \exp(-e_{j,t}), \\ \lambda_{j,t} &= \rho_{j,t} \exp\big(i[\theta_j + (1 - \rho_{j,t})\kappa_p p_t]\big). \end{aligned} \tag{32}$$

The shared decay rescaling preserves $e_{j,t}/e_{k,t} = \nu_j/\nu_k$.

The constants $\kappa_r = \log 16$ and $\kappa_p = \pi/2$ are fixed hyperparameters. The first allows the effective decay to vary from just above one sixteenth to just below sixteen times its baseline, centered at the unmodified spectrum when $r_t = 0$. Under a held-constant control, the exponential decay timescale $1/e_{j,t}$ changes reciprocally. This gives a substantial adjustment range without requiring an unbounded controller output. Smaller ranges restrict adaptation; larger ranges permit more extreme forgetting and more rapid variation.

The phase range permits a signed correction of less than a quarter turn before multiplication by the retention factor. It is large enough to change orientation appreciably without allowing an unrestricted per-step correction.

Role of the phase clock. The factor $1 - \rho_{j,t}$ protects nearly persistent modes from large observation-driven phase corrections. For small effective decay, $1 - e^{-e_{j,t}}$ is approximately $e_{j,t}$, so the correction decreases with the decay. For rapidly forgotten modes, larger phase changes act

on a previous-state contribution that is already attenuated. Removing this factor tests the retention dependence of phase adaptation.

C.2 Content, write gate, normalization, and gain

Input activation and saturation. The identity mapping preserves the amplitude of each projected input and avoids activation saturation. A componentwise $\tanh$ instead limits the real and imaginary input contributions to $(-1, 1)$, which can reduce the effect of extreme projected observations. The trade-off is that $\varphi'(z) = 1 - \tanh^2(z)$ becomes small at large $|z|$, potentially weakening content-projection gradients and discarding magnitude information. The checkpoint analysis in Appendix F measures projected-input magnitudes, activation saturation, and derivatives alongside task scores; saturation is markedly task-dependent.

Modal write response. The gate $q_{j,t} = \sigma(s_j p_t + v_j r_t)$ couples writing to the same two controls used by the transition, but gives each mode its own response. Modes can therefore write more or less strongly under a common retention or phase signal. To remove adaptive gating without changing the initialization scale, the matched intervention is $q_{j,t} = 1/2$. Setting $q_{j,t} = 1$ while leaving the gain unchanged would double the reference write and confound the comparison.

Retention-dependent normalization. The factor $\sqrt{1 - \rho_{j,t}^2}$ reduces the instantaneous write when a mode is highly persistent and increases it when old content is attenuated more strongly. In the implemented coordinates, the denominator in Eq. (17) makes this factor equal to one relative to the reference gain when $r_t = 0$. This is related to decay-dependent normalizations used in recurrent models (Orvieto et al., 2023; De et al., 2024), but temporally correlated inputs and input-dependent coefficients prevent a general constant-variance interpretation. A matched ablation replaces the adaptive square-root factor with $\sqrt{1 - e^{-2\nu_j}}$, retaining its baseline amplitude.

Independent reference gain. The parameter β_j sets a learned reference write scale independently of the decay parameter a_j. For nonlinear content, an external gain generally cannot be absorbed into the content matrix over the full input range. The gain-learning ablation holds its initialized value fixed. Since ζ_j depends on ν_j through Eq. (16), freezing ζ_j is not the same intervention as freezing β_j.

Control sharing. Shared controllers use $2(D+1)$ parameters. Replacing them by two independent controllers per complex mode uses $2H(D + 1)$ parameters and increases control-projection work from $O(D)$ to $O(HD)$. The state still has H complex modes in both cases. The increased parameter count and projection work affect latency. Accumulated sensitivity storage remains $O(HD)$ for both shared and unshared coordinate-local controls.

C.3 Programmatically generated diagnostic tasks

Figure 1 evaluates two memory requirements with generated sequences: preserving relevant events among distractors and tracking a state with changing dynamics.

Selective event-order retention. Each sequence contains three relevant events, labeled A, B, and C, among distractor inputs. The classification target is their order of occurrence: the example in Figure 1(a) has target B–A–C. Recovering this order requires remembering the earlier event identities through the distractors and incorporating each later event in sequence. Classification error is the fraction of evaluation sequences with an incorrect order prediction. SPARC and RG-LRU both achieve zero error, compared with 13.7% for RTU.

Event-driven dynamical tracking. The prediction target is a latent-state trajectory generated under event-dependent contraction and rotation. Event times mark changes in how the state evolves: contraction changes its retained magnitude, while rotation changes its direction and thus its coordinates. The model receives the generated input sequence and predicts the evolving state throughout the sequence. Figure 1(b) shows the first target coordinate and marks the event times. SPARC ob-

Table 3: Component-removal definitions.

Variant	Intervention and scope
Fixed shared controls	Set $r_t = p_t = 0$ in every pathway; retain the learned modal spectrum and the selected content activation.
No adaptive retention	Set $e_{j,t} = \nu_j$ only in the transition, including its phase clock; retain the original adaptive write normalization and gate.
No adaptive phase	Set $\lambda_{j,t} = \rho_{j,t} e^{i\theta_j}$; retain p_t in the write gate.
No phase clock	Replace $(1 - \rho_{j,t})\kappa_p p_t$ by $\kappa_p p_t$; keep retention and all write terms.
Static phase clock	Use $(1 - e^{-\nu_j})\kappa_p p_t$; retain adaptive retention and writing.
Static transition	Set $\lambda_j = e^{-\nu_j + i\theta_j}$; retain the original adaptive write pathway.
Constant write gate	Set $q_{j,t} = 1/2$; keep the reference gain and retention-dependent normalization.
Fixed write normalization	Replace $\sqrt{1 - \rho_{j,t}^2}$ by $\sqrt{1 - e^{-2\nu_j}}$ in Eq. (8).
Frozen reference gain	Hold $\beta_j = \beta_j^{(0)}$; continue to compute ζ_j using Eq. (16).
Linear input content	Use $\varphi(z) = z$ instead of $\tanh(z)$, without adding a recurrent-state nonlinearity.
Unshared control	Use separate $r_{j,t}, p_{j,t}$ controllers for each mode with the same ranges; report the added parameters and computation.

Appendix E reports measured variants; fixed shared controls and unshared control were not run. Constant-gate and fixed-normalization variants preserve the baseline write amplitude.

tains NMSE 0.096, compared with 0.358 for RTU and 0.382 for RG-LRU. NMSE is the summed squared prediction error divided by the summed squared target magnitude over the evaluation set.

Training and evaluation. Each task uses 384 training and 512 evaluation sequences of length 48, with batch size 64 and 128 real recurrent-state coordinates. Training runs for 1200 updates on event order and 450 on dynamical tracking, using AdamW (Loshchilov & Hutter, 2017) with learning rate 0.01, zero weight decay, and gradient-norm clipping at one. Methods share the same generated task data. All 512 evaluation sequences contribute to each reported score. The upper panels of Figure 1 display the input and target of evaluation sequence zero.

C.4 Matched component-removal definitions

Table 3 specifies interventions that distinguish these mechanisms. For retention or phase interventions, the other uses of the shared signal are retained where indicated; otherwise setting a controller to zero would alter both transition and writing. The measured variants were retrained with matched task protocols, initialization sources, and state sizes. Appendix E gives their numerical results.

The retrained comparisons evaluate task performance under each intervention; checkpoint diagnostics measure how the trained cell uses the same components.

D Experimental Details and Hyperparameters

D.1 Seeds and reporting convention

All benchmark and ablation results are averaged over three training seeds, with the same task-specific seed set used across methods. Data splits and pixel permutations are fixed independently of training randomness. The classification data split and checkpoint-selection rule are specified below.

Methods retain their native recurrent equations, internal gating, normalization, and initialization within the common task wrapper. The matched RL state budgets correspond to 192 complex modes for SPARC and LRU in continuous control, and 64 for SPARC in POPGym. SPARC uses $\tanh$ content; the identity mapping is evaluated as a separate ablation.

Table 4: Shared reinforcement-learning hyperparameters.

Setting	Continuous control	POPGym
Environment interaction steps	4,999,168	3,999,744
Physical rollout length	2048	2048
PPO epochs per rollout	4	10
Minibatches per epoch	32	32
Real-valued recurrent state coordinates	384	128
SPARC complex modes	192	64
Encoder width	64	64
Policy-head / value-head width	64/64	64/64
Discount factor	0.99	0.99
GAE parameter	0.95	0.95
PPO clipping parameter	0.2	0.2
Value-loss coefficient	0.5	0.5
Entropy coefficient	0	0
Global gradient-norm clipping	0.5	0.5
Optimizer	Adam	Adam
Optimizer ϵ	10^{-5}	10^{-5}
Observation normalization	Enabled	Disabled
Reward normalization	Enabled	Disabled
Previous action / reward inputs	Absent / absent	Absent / absent
RTRL minibatch sequence length	1	1
GRU / Mamba-3 TBPTT length	4	4

Learning rates are fixed across methods within each task.

D.2 Reinforcement-learning architecture and optimization

The observation encoder, policy head, and value head each have width 64. The encoder uses $\tanh$; SPARC's real and imaginary states are concatenated and passed through ReLU before the two heads. Continuous policies use a diagonal Gaussian with learned state-independent log standard deviation and action clipping; discrete policies use categorical logits. No previous-action or previous-reward features are supplied to the recurrent cell. The encoded input and SPARC head features are

$$\mathbf{x}_t = \tanh(W_{\mathrm{enc}}\mathbf{o}_t + \mathbf{b}_{\mathrm{enc}}), \qquad \mathbf{z}_t = \mathrm{ReLU}([\mathrm{Re}\,\mathbf{h}_t; \mathrm{Im}\,\mathbf{h}_t]). \tag{33}$$

Continuous control applies the partial-observation mask and enables observation and reward normalization. POPGym uses the native task observations without an added mask or injected-noise wrapper, and disables both normalizations. Native task noise is retained. The POPGym environment configurations are the Easy variants. Noisy Pendulum uses the continuous-action policy, while the other five tasks use categorical policies.

The learning rate is 10^{-4} for all listed RL tasks except Walker-P (3×10^{-5}), RepeatFirst (10^{-3}), and HigherLower (10^{-5}).

SPARC, RTU, LRU, RG-LRU, and GateLoop use stored recurrent sensitivities and minibatch sequence length 1. GRU and Mamba-3 use TBPTT with length 4: GRU has dense state-dependent gates, and Mamba-3 maintains additional recurrent caches, requiring sensitivity structures beyond the mode-local diagonal updates used here.

Both SPARC controllers use the full task learning rate. Stored-gradient and stored-target options are enabled in PPO. Recurrent states and eligibility traces reset at episode boundaries. RL states use complex64, with single-precision real parameters and projections.

D.3 Spectral and controller initialization

Shared initialization. The controller vectors and biases, together with s_j and v_j, are initialized to zero. Consequently $r_t = p_t = 0$, $q_{j,t} = 1/2$, and

$$e^{\zeta_j}\sqrt{1 - e^{-2\nu_j}}\, q_{j,t} = e^{\beta_j}. \tag{34}$$

Content projections have no bias, and recurrent states start at zero. This initialization sets the reference write amplitude to e^{β_j}.

Reinforcement learning. SPARC reuses the paired RTU arrays for the encoder, readout, modal spectrum, and content projections. For independent uniform draws $U_j, V_j \in (0, 1)$,

$$\rho_j^{(0)} = \sqrt{U_j}, \qquad \nu_j^{(0)} = -\log \rho_j^{(0)}, \qquad \theta_j^{(0)} = 6.28 V_j. \tag{35}$$

The log coordinates are $a_j^{(0)} = \log \nu_j^{(0)}$ and $\vartheta_j^{(0)} = \log \theta_j^{(0)}$. The reference gain is initialized as

$$\beta_j^{(0)} = \log\left(\sqrt{1 - e^{-2\nu_j^{(0)}}} + 10^{-8}\right), \tag{36}$$

and then optimized independently of a_j. The compact log gain is obtained from Eq. (16), including its factor-of-two conversion.

Classification. The squared spectral radius is uniform on $[0.9^2, 0.999^2]$ and the phase is uniform on $[0, 2\pi)$. Sampling squared radius uniformly gives an area-uniform annulus, following LRU; placing this annulus near the unit circle initializes slowly decaying modes useful for long-range propagation (Orvieto et al., 2023). Shared arrays, including the readout and reference gain, are copied from the paired LRU initializer. The real and imaginary content matrices use its scaling by $1/\sqrt{2d_{\text{model}}}$, with $d_{\text{model}} = 64$.

D.4 Full-sequence architecture and optimization

Classification uses four unidirectional residual sequence layers of model width 64, pre-layer normalization, the full-GLU activation block, dropout 0.1, and sequence-mean pooling. SPARC has 64 complex modes per layer. The learned complex readout matrix C and real skip coefficients $\mathbf{D}_{\text{skip}}$ give

$$\mathbf{y}_t = \mathrm{Re}(C\mathbf{h}_t) + \mathbf{D}_{\text{skip}} \odot \mathbf{x}_t. \tag{37}$$

They belong to the regular-parameter optimizer group. A scalar-input embedding feeds the stack; each residual layer contains the recurrent mixer, readout, and activation block.

Training uses $20{,}000$ full-sequence updates, batch size 32, and validation every 500 updates. The regular learning rate warms up from 10^{-6} to 1.95×10^{-3} over $2{,}000$ updates, then uses cosine decay to 10^{-6}. The recurrent learning rate is one quarter of the regular rate throughout. Recurrent parameters use Adam (Kingma & Ba, 2014) without weight decay, and the remaining parameters use AdamW (Loshchilov & Hutter, 2017) with weight decay 0.05.

D.5 Classification data and preprocessing

The classification protocol splits the official training partition into fit and internal-validation subsets; reported accuracies are measured on internal validation. All inputs have one scalar channel, and Table 5 gives the exact sizes and sequence lengths.

UCR preprocessing uses a single global z-score transformation fitted only on the training-fit subset. Pixel normalization also uses training-fit examples only. No data augmentation is applied. Permuted MNIST uses one fixed pixel permutation; grayscale CIFAR-10 uses row-major ordering. The data splits and pixel permutation remain fixed across models and training runs; exact seeds are retained in the experiment configurations.

For each run, checkpoint selection maximizes internal validation accuracy over the fixed budget and breaks ties by lower validation loss. Table 2 averages the selected internal-validation accuracies across three seeds.

Table 5: Classification data configurations.

Dataset	Length	Classes	Fit	Validation
FordA	500	2	3240	361
StarLightCurves	1024	3	900	100
UWaveGestureLibraryAll	945	8	806	90
Permuted MNIST	784	10	54,001	5999
Seq. CIFAR-10 Gray	1024	10	45,000	5000

Length is the number of sequence positions. Fit and validation counts refer to examples in an internal split of the official training partition.

D.6 Score aggregation and curve construction

For training run $n \in \{1, \dots, N_{\text{run}}\}$ with $N_{\text{run}} = 3$ and rollout k, let $C_{n,k}$ count completed episodes and $R_{n,k,\ell}$ denote their undiscounted returns. The rollout mean and final-window score are

$$m_{n,k} = \frac{1}{C_{n,k}} \sum_{\ell=1}^{C_{n,k}} R_{n,k,\ell}, \qquad J_n = \frac{1}{100} \sum_{k=K_n-99}^{K_n} m_{n,k}, \tag{38}$$

where K_n is the final rollout index. Every reporting-window rollout must have a positive completed-episode count and finite returns. Each rollout receives equal weight in the final-window average.

Benchmark tables use the three-seed mean

$$\bar{J} = \frac{1}{N_{\text{run}}} \sum_{n=1}^{N_{\text{run}}} J_n. \tag{39}$$

For classification, J_n instead denotes the selected internal validation accuracy.

Learning curves first apply the same 100-rollout moving average within each run:

$$\widetilde{m}_{n,k} = \frac{1}{100} \sum_{\ell=k-99}^{k} m_{n,\ell}, \qquad k \geq 100. \tag{40}$$

At each shared environment-step position, the curve is the mean of the three smoothed seed trajectories and shading is their sample standard deviation. Curves start at rollout 100 and retain the original environment-step and return scales.

D.7 Additional benchmark results

Table 6 gives the complete POPGym means. Figure 5 shows continuous-control learning curves using the aggregation above.

E Component Ablations

We retrain nine variants to examine the contributions of adaptive retention, phase, their coupling, and the write pathway. Each uses SPARC's task-specific backbone, training budget, initialization, and three training seeds. The evaluation covers the 15 main benchmark tasks plus StatelessCartPoleEasy, abbreviated CartPole. Tables report mean scores and sample standard deviations, together with differences computed between matched seeds. A positive difference means the variant scores higher.

The transition ablations leave the adaptive write normalization and gate active, so that changes in performance can be assessed while the model can still control incoming information. All remaining parameters are retrained under each intervention. These comparisons therefore measure performance after adaptation to the modified cell; Appendix F separately examines component activity and interventions at fixed checkpoints.

Table 7 summarizes selected variants on four representative tasks before the complete comparisons below.

Table 6: Partially observable memory tasks in POPGym.

Method	Autoenc.	CountRec.	Repeat1st	RepeatPrev.	NoisyPend.	HigherLow.
RTU	**-0.427**	-0.565	0.448	0.891	0.244	0.501
LRU	-0.431	-0.597	0.242	**0.908**	0.246	0.498
RG-LRU	-0.476	-0.644	0.550	-0.004	0.246	**0.503**
GateLoop	-0.438	-0.647	0.727	0.268	0.245	0.498
Mamba-3	-0.499	-0.811	0.471	-0.428	0.269	0.487
GRU	-0.469	-0.835	0.811	0.744	0.311	0.495
SPARC	-0.431	**-0.515**	**0.829**	0.855	**0.461**	0.500

Mean return across three training seeds. Columns follow the task order in Figure 3. Higher is better. Best means are bold; second-best means are underlined.

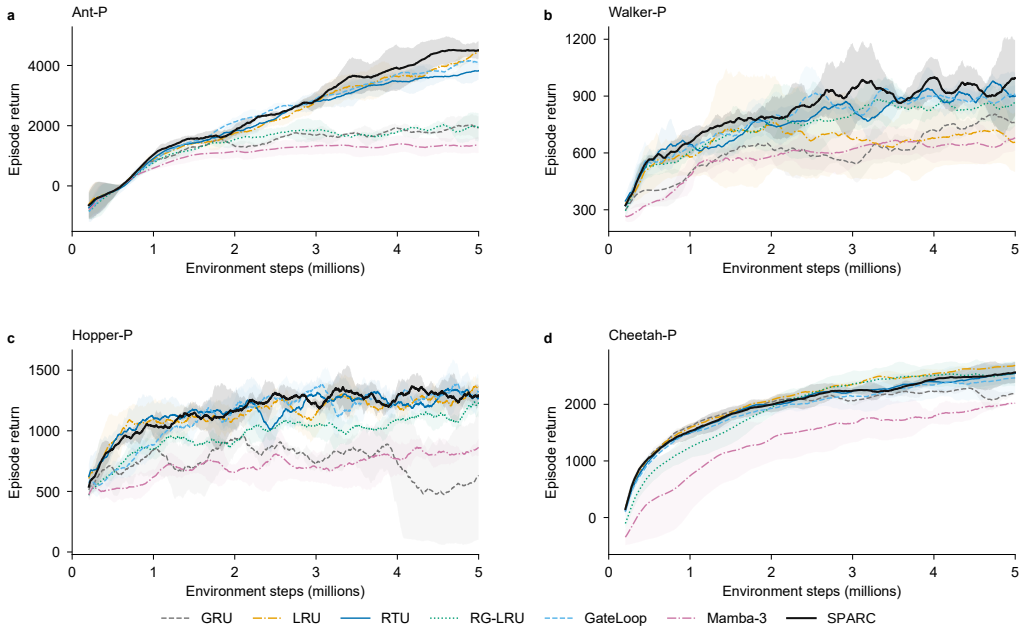

Figure 5: Learning curves on Ant-P, Walker-P, Hopper-P, and Cheetah-P. Each curve averages three seed trajectories after applying a 100-rollout moving average within each seed. Shading denotes one sample standard deviation across the smoothed trajectories.

E.1 Adaptive phase

The retention-only variant sets $\lambda_{j,t} = \rho_{j,t} e^{i\theta_j}$, removing the input-dependent phase correction while retaining baseline rotation, adaptive retention, and the full write pathway. The phase controller still participates in the write gate.

Removing adaptive phase lowers Walker-P return from 994.79 to 837.70, RepeatFirst return from 0.829 to 0.756, and FordA accuracy from 96.68% to 96.03% (Table 8). These losses occur despite retaining selective forgetting and writing, indicating a contribution from input-dependent rotation on these tasks. The Walker-P difference has substantial seed variation (-157.09 ± 233.46), however, and the retention-only variant improves UWave by 1.11 percentage points and RepeatPrevious by 0.083. Adaptive phase thus provides an additional useful control of memory evolution, with benefits that depend on the task.

Table 7: Selected component ablations.

Variant	RepeatFirst	Walker-P	FordA	CIFAR-10
SPARC	**0.829**	**994.79**	**96.68**	**59.92**
Neutral write gate	0.663	909.56	**96.68**	59.35
Linear content	0.123	921.90	96.12	59.46
Fixed write normalization	0.628	<u>943.45</u>	96.40	58.73
No phase clock	0.288	928.64	<u>96.49</u>	49.03
Retention-only transition	<u>0.756</u>	837.70	96.03	<u>59.79</u>

RepeatFirst and Walker-P: return; FordA and CIFAR-10: accuracy (%). Higher is better; best means are bold and second-best means underlined. Transition-only variants retain the original write pathway. Paired differences, sample spreads, and all variants are in Appendix E.

Table 8: Retention-only transition.

Task	SPARC	Variant	Δ (variant $-$ SPARC)
Autoencode	-0.431 ± 0.003	$\mathbf{-0.424} \pm 0.023$	0.007 ± 0.021
CountRecall	-0.515 ± 0.098	$\mathbf{-0.493} \pm 0.148$	0.022 ± 0.051
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.756 ± 0.051	-0.073 ± 0.058
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.938} \pm 0.051$	0.083 ± 0.019
Noisy Pendulum	0.461 ± 0.039	$\mathbf{0.504} \pm 0.053$	0.043 ± 0.020
CartPole	$\mathbf{0.959} \pm 0.026$	0.921 ± 0.016	-0.038 ± 0.010
HigherLower	$\mathbf{0.500} \pm 0.003$	0.499 ± 0.001	-0.001 ± 0.003
Ant-P	4507.80 ± 292.89	$\mathbf{4554.24} \pm 319.76$	46.44 ± 26.87
Walker-P	$\mathbf{994.79} \pm 198.05$	837.70 ± 112.75	-157.09 ± 233.46
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1273.43 ± 81.61	-9.51 ± 153.98
Cheetah-P	2563.76 ± 184.38	$\mathbf{2629.61} \pm 3.15$	65.85 ± 181.23
FordA	$\mathbf{96.68} \pm 0.28$	96.03 ± 0.42	-0.65 ± 0.42
StarLightCurves	$\mathbf{99.67} \pm 0.58$	$\mathbf{99.67} \pm 0.58$	0.00 ± 1.00
UWave	96.30 ± 1.70	$\mathbf{97.41} \pm 0.64$	1.11 ± 1.92
Permuted MNIST	96.87 ± 0.11	$\mathbf{96.93} \pm 0.17$	0.06 ± 0.15
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.79 ± 1.29	-0.13 ± 0.17

Mean $\pm$ sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant $-$ SPARC difference. Higher is better; the larger mean in each pair is bold.

E.2 Adaptive retention

The phase-only variant uses $\exp(-\nu_j + i[\theta_j + (1 - e^{-\nu_j})\kappa_p p_t])$ as its transition. Each mode keeps a learned, input-independent decay while its phase remains adaptive. The write gate and dynamic write normalization are retained. Because the phase clock also uses the baseline decay, this intervention removes both adaptive retention and its influence on the size of the phase correction.

Mean return decreases on all four continuous-control tasks, including Ant-P from 4507.80 to 3930.02 and Walker-P from 994.79 to 863.20 (Table 9). UWave accuracy falls from 96.30% to 91.85%, the largest classification loss. RepeatFirst also drops to 0.326, with a large sample standard deviation of 0.515. Adaptive phase and writing therefore do not fully compensate for removing retention control in these comparisons, although Autoencode, RepeatPrevious, and sequential CIFAR-10 obtain higher means with the phase-only transition.

E.3 Static transition with adaptive writing

To test the value of adapting existing memory when writing is already adaptive, we fix the transition to $\lambda_j = \exp(-\nu_j + i\theta_j)$. The modal spectrum remains trainable, and both shared controllers still

Table 9: Phase-only transition.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	-0.431 ± 0.003	$\mathbf{-0.424} \pm 0.023$	0.007 ± 0.021
CountRecall	$\mathbf{-0.515} \pm 0.098$	-0.568 ± 0.229	-0.053 ± 0.131
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.326 ± 0.515	-0.503 ± 0.522
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.902} \pm 0.024$	0.047 ± 0.056
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.457 ± 0.097	-0.004 ± 0.102
CartPole	$\mathbf{0.959} \pm 0.026$	0.947 ± 0.005	-0.012 ± 0.032
HigherLower	$\mathbf{0.500} \pm 0.003$	0.498 ± 0.004	-0.002 ± 0.003
Ant-P	$\mathbf{4507.80} \pm 292.89$	3930.02 ± 70.94	-577.78 ± 363.83
Walker-P	$\mathbf{994.79} \pm 198.05$	863.20 ± 111.82	-131.59 ± 104.36
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1220.37 ± 80.70	-62.57 ± 139.22
Cheetah-P	$\mathbf{2563.76} \pm 184.38$	2535.70 ± 61.05	-28.06 ± 123.33
FordA	$\mathbf{96.68} \pm 0.28$	95.48 ± 0.42	-1.20 ± 0.58
StarLightCurves	$\mathbf{99.67} \pm 0.58$	99.00 ± 1.00	-0.67 ± 0.58
UWave	$\mathbf{96.30} \pm 1.70$	91.85 ± 2.80	-4.44 ± 2.94
Permuted MNIST	$\mathbf{96.87} \pm 0.11$	96.77 ± 0.07	-0.10 ± 0.06
Seq. CIFAR-10 Gray	59.92 ± 1.41	$\mathbf{60.16} \pm 1.45$	0.24 ± 1.30

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

regulate the write pathway. Thus, incoming information can be selected and scaled according to the input, while every stored contribution propagates through an input-independent transition.

SPARC exceeds this variant on Ant-P (4507.80 versus 3794.64), RepeatFirst (0.829 versus 0.534), and UWave (96.30% versus 92.59%; Table 10). These comparisons provide empirical examples where adaptive writing alone does not recover the full model's performance. The static transition nevertheless improves CountRecall and Noisy Pendulum, reaching -0.397 and 0.532, respectively, so the advantage of adapting memory propagation is not uniform across the recall and control tasks.

E.4 Retention-coupled phase clock

The phase clock scales the shared phase correction according to each mode's current retention. Replacing $(1 - \rho_{j,t})\kappa_p p_t$ by $\kappa_p p_t$ removes this scaling, so even highly persistent modes receive the full correction. Adaptive retention, baseline rotation, and all write terms remain in place.

This change produces large losses across several task families (Table 11): RepeatFirst return falls from 0.829 to 0.288, Ant-P return from 4507.80 to 2848.36, and sequential CIFAR-10 accuracy from 59.92% to 49.03%. The CIFAR-10 paired difference is -10.89 ± 2.16 percentage points. In checkpoint diagnostics, the clock reduces RepeatFirst's phase correction to 3.29% of its uncoupled value (Appendix F). Together with the retraining results, this supports limiting observation-driven rotation in persistent modes as a useful part of the phase-control design.

E.5 Input dependence of the phase clock

The static-clock variant retains modal scaling of the phase correction but computes it from the learned baseline decay: $(1 - e^{-\nu_j})\kappa_p p_t$. Retention, the phase signal, and writing remain input-dependent. Comparing this variant with SPARC tests whether the phase scale benefits from following current retention; comparing it with the uncoupled variant distinguishes that effect from modal scaling itself.

On sequential CIFAR-10, a static clock reaches 59.68%, close to SPARC's 59.92% and substantially above the uncoupled variant's 49.03% (Tables 12 and 11). Baseline scaling therefore recovers most of the loss on this task. RepeatFirst shows a larger gap to SPARC (0.383 versus 0.829), although the static-clock result varies widely across seeds (0.547 sample SD). CountRecall improves to -0.389.

Table 10: Static transition.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.441 ± 0.019	-0.009 ± 0.016
CountRecall	-0.515 ± 0.098	$\mathbf{-0.397} \pm 0.161$	0.119 ± 0.064
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.534 ± 0.196	-0.294 ± 0.189
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.900} \pm 0.077$	0.045 ± 0.061
Noisy Pendulum	0.461 ± 0.039	$\mathbf{0.532} \pm 0.031$	0.070 ± 0.050
CartPole	$\mathbf{0.959} \pm 0.026$	0.949 ± 0.012	-0.010 ± 0.038
HigherLower	$\mathbf{0.500} \pm 0.003$	0.499 ± 0.004	-0.001 ± 0.001
Ant-P	$\mathbf{4507.80} \pm 292.89$	3794.64 ± 383.71	-713.16 ± 676.60
Walker-P	$\mathbf{994.79} \pm 198.05$	884.66 ± 53.96	-110.13 ± 163.06
Hopper-P	1282.94 ± 78.06	$\mathbf{1318.28} \pm 203.47$	35.34 ± 169.24
Cheetah-P	$\mathbf{2563.76} \pm 184.38$	2557.46 ± 147.79	-6.30 ± 36.60
FordA	$\mathbf{96.68} \pm 0.28$	96.21 ± 0.16	-0.46 ± 0.42
StarLightCurves	$\mathbf{99.67} \pm 0.58$	$\mathbf{99.67} \pm 0.58$	0.00 ± 1.00
UWave	$\mathbf{96.30} \pm 1.70$	92.59 ± 2.31	-3.70 ± 0.64
Permuted MNIST	$\mathbf{96.87} \pm 0.11$	$\mathbf{96.87} \pm 0.12$	-0.01 ± 0.18
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.51 ± 0.63	-0.41 ± 0.87

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

Table 11: No phase clock.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.454 ± 0.029	-0.022 ± 0.027
CountRecall	$\mathbf{-0.515} \pm 0.098$	-0.627 ± 0.151	-0.112 ± 0.145
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.288 ± 0.242	-0.541 ± 0.234
RepeatPrevious	$\mathbf{0.855} \pm 0.033$	0.661 ± 0.120	-0.194 ± 0.136
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.384 ± 0.017	-0.077 ± 0.056
CartPole	$\mathbf{0.959} \pm 0.026$	0.945 ± 0.008	-0.013 ± 0.018
HigherLower	$\mathbf{0.500} \pm 0.003$	0.499 ± 0.002	-0.001 ± 0.002
Ant-P	$\mathbf{4507.80} \pm 292.89$	2848.36 ± 945.33	-1659.44 ± 652.44
Walker-P	$\mathbf{994.79} \pm 198.05$	928.64 ± 118.58	-66.16 ± 89.52
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1181.06 ± 87.00	-101.88 ± 120.14
Cheetah-P	$\mathbf{2563.76} \pm 184.38$	2440.37 ± 80.43	-123.39 ± 103.95
FordA	$\mathbf{96.68} \pm 0.28$	96.49 ± 0.70	-0.18 ± 0.58
StarLightCurves	$\mathbf{99.67} \pm 0.58$	99.33 ± 1.15	-0.33 ± 0.58
UWave	$\mathbf{96.30} \pm 1.70$	$\mathbf{96.30} \pm 1.70$	0.00 ± 2.22
Permuted MNIST	$\mathbf{96.87} \pm 0.11$	96.44 ± 0.10	-0.43 ± 0.20
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	49.03 ± 2.48	-10.89 ± 2.16

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

The additional benefit of following current retention is thus less consistent than the benefit of retaining a phase clock, particularly in classification.

Table 12: Static phase clock.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.452 ± 0.015	-0.021 ± 0.017
CountRecall	-0.515 ± 0.098	$\mathbf{-0.389} \pm 0.068$	0.126 ± 0.081
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.383 ± 0.547	-0.446 ± 0.555
RepeatPrevious	$\mathbf{0.855} \pm 0.033$	0.845 ± 0.053	-0.010 ± 0.071
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.413 ± 0.087	-0.048 ± 0.069
CartPole	$\mathbf{0.959} \pm 0.026$	0.811 ± 0.016	-0.148 ± 0.042
HigherLower	$\mathbf{0.500} \pm 0.003$	0.499 ± 0.003	-0.001 ± 0.003
Ant-P	$\mathbf{4507.80} \pm 292.89$	4155.32 ± 981.54	-352.48 ± 1274.43
Walker-P	$\mathbf{994.79} \pm 198.05$	944.61 ± 116.43	-50.18 ± 256.05
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1263.91 ± 113.93	-19.04 ± 140.13
Cheetah-P	2563.76 ± 184.38	$\mathbf{2628.95} \pm 183.48$	65.19 ± 0.90
FordA	$\mathbf{96.68} \pm 0.28$	96.03 ± 0.97	-0.65 ± 1.12
StarLightCurves	$\mathbf{99.67} \pm 0.58$	99.33 ± 1.15	-0.33 ± 0.58
UWave	$\mathbf{96.30} \pm 1.70$	95.19 ± 1.70	-1.11 ± 0.00
Permuted MNIST	96.87 ± 0.11	$\mathbf{96.91} \pm 0.28$	0.03 ± 0.18
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.68 ± 1.40	-0.24 ± 0.27

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

E.6 Modal write gate

We replace the modal gate $\sigma(s_j p_t + v_j r_t)$ by its neutral value $1/2$, preserving the reference write amplitude while removing each mode's learned gating response to the shared controls. The transition, reference gain, and retention-dependent write normalization remain active. In the implementation and diagnostics, the equivalent normalized gate is $2\sigma(s_j p_t + v_j r_t)$, whose neutral value is one.

Neutralizing the gate lowers RepeatFirst return from 0.829 to 0.663 and Ant-P return from 4507.80 to 3942.01, while FordA remains at 96.68% (Table 13). The contrast between RepeatFirst and FordA also appears in the fitted models: with other write factors held fixed, gating multiplies recorded write energy by 1.816 on RepeatFirst and 1.011 on FordA. The retrained results and checkpoint measurements both indicate greater use of this pathway on RepeatFirst, where a neutral gate leaves a performance gap even after the remaining parameters are retrained.

E.7 Learned reference write gain

The reference gain sets the write amplitude of each mode at zero control, independently of its learned decay. We freeze β_j at initialization while continuing to train the spectrum, content projections, and dynamic write factors. The derived coordinate ζ_j is still recomputed using Eq. (16); fixing ζ_j would change the reference amplitude as the decay is trained and would test a different constraint.

With the reference gain frozen, Walker-P return decreases from 994.79 to 856.69, RepeatFirst from 0.829 to 0.721, and FordA accuracy from 96.68% to 96.12% (Table 14). Learning a persistent modal write scale thus remains useful even when writing already adapts to the input. The gain is less consequential on StarLightCurves and UWave, whose means are unchanged at the reported precision. RepeatPrevious instead improves to 0.907 with the gain frozen.

E.8 Retention-dependent write normalization

The normalization links write amplitude to the current retention: persistent modes receive smaller writes, while stronger forgetting permits larger writes. We remove this dependence by replacing

Table 13: Neutral write gate.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	-0.431 ± 0.003	$\mathbf{-0.430} \pm 0.026$	0.001 ± 0.025
CountRecall	-0.515 ± 0.098	$\mathbf{-0.511} \pm 0.059$	0.004 ± 0.073
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.663 ± 0.067	-0.166 ± 0.074
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.867} \pm 0.033$	0.012 ± 0.001
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.425 ± 0.030	-0.036 ± 0.022
CartPole	$\mathbf{0.959} \pm 0.026$	0.915 ± 0.078	-0.044 ± 0.104
HigherLower	$\mathbf{0.500} \pm 0.003$	$\mathbf{0.500} \pm 0.003$	-0.000 ± 0.002
Ant-P	$\mathbf{4507.80} \pm 292.89$	3942.01 ± 65.70	-565.79 ± 358.59
Walker-P	$\mathbf{994.79} \pm 198.05$	909.56 ± 305.84	-85.23 ± 276.07
Hopper-P	1282.94 ± 78.06	$\mathbf{1316.95} \pm 90.72$	34.00 ± 162.90
Cheetah-P	2563.76 ± 184.38	$\mathbf{2600.65} \pm 69.98$	36.89 ± 114.40
FordA	$\mathbf{96.68} \pm 0.28$	$\mathbf{96.68} \pm 0.28$	0.00 ± 0.28
StarLightCurves	$\mathbf{99.67} \pm 0.58$	$\mathbf{99.67} \pm 0.58$	0.00 ± 0.00
UWave	$\mathbf{96.30} \pm 1.70$	95.93 ± 2.31	-0.37 ± 0.64
Permuted MNIST	$\mathbf{96.87} \pm 0.11$	96.86 ± 0.03	-0.02 ± 0.10
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.35 ± 1.78	-0.57 ± 0.40

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

Table 14: Frozen reference gain.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.447 ± 0.024	-0.016 ± 0.021
CountRecall	$\mathbf{-0.515} \pm 0.098$	-0.613 ± 0.114	-0.098 ± 0.051
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.721 ± 0.016	-0.107 ± 0.024
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.907} \pm 0.031$	0.052 ± 0.052
Noisy Pendulum	0.461 ± 0.039	$\mathbf{0.464} \pm 0.031$	0.003 ± 0.041
CartPole	$\mathbf{0.959} \pm 0.026$	0.950 ± 0.027	-0.009 ± 0.000
HigherLower	$\mathbf{0.500} \pm 0.003$	$\mathbf{0.500} \pm 0.003$	-0.000 ± 0.001
Ant-P	$\mathbf{4507.80} \pm 292.89$	4353.05 ± 19.52	-154.75 ± 312.41
Walker-P	$\mathbf{994.79} \pm 198.05$	856.69 ± 55.47	-138.11 ± 146.46
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1226.34 ± 229.83	-56.60 ± 300.35
Cheetah-P	2563.76 ± 184.38	$\mathbf{2623.83} \pm 83.09$	60.07 ± 101.30
FordA	$\mathbf{96.68} \pm 0.28$	96.12 ± 0.28	-0.55 ± 0.00
StarLightCurves	$\mathbf{99.67} \pm 0.58$	$\mathbf{99.67} \pm 0.58$	0.00 ± 1.00
UWave	$\mathbf{96.30} \pm 1.70$	$\mathbf{96.30} \pm 1.70$	0.00 ± 0.00
Permuted MNIST	$\mathbf{96.87} \pm 0.11$	96.69 ± 0.36	-0.18 ± 0.36
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.72 ± 1.65	-0.20 ± 0.27

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

$\sqrt{(1-e^{-2e_{j,t}})/(1-e^{-2\nu_j})}$ with one in the implemented coordinates. Equivalently, the compact write uses $\sqrt{1-e^{-2\nu_j}}$ in place of $\sqrt{1-\rho_{j,t}^2}$. The reference gain and gate remain trainable, and the zero-control amplitude is preserved.

The largest continuous-control loss occurs on Ant-P, from 4507.80 to 2983.48, with substantial variation across seeds (Table 15). RepeatFirst falls from 0.829 to 0.628, and sequential CIFAR-10 accuracy from 59.92% to 58.73%; CountRecall improves to -0.452. The losses support coordinating writing with retention beyond the adjustment available through the learned gain and gate. This interpretation concerns control of write amplitude: correlated inputs and adaptive coefficients prevent a general constant-variance interpretation of the normalization.

Table 15: Fixed write normalization.

Task	**SPARC**	**Variant**	**Δ (variant $-$ SPARC)**
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.477 ± 0.021	-0.045 ± 0.019
CountRecall	-0.515 ± 0.098	$\mathbf{-0.452} \pm 0.036$	0.063 ± 0.077
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.628 ± 0.049	-0.200 ± 0.057
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.878} \pm 0.065$	0.023 ± 0.037
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.366 ± 0.145	-0.095 ± 0.133
CartPole	$\mathbf{0.959} \pm 0.026$	0.891 ± 0.065	-0.067 ± 0.039
HigherLower	0.500 ± 0.003	$\mathbf{0.501} \pm 0.004$	0.000 ± 0.001
Ant-P	$\mathbf{4507.80} \pm 292.89$	2983.48 ± 1285.42	-1524.33 ± 992.53
Walker-P	$\mathbf{994.79} \pm 198.05$	943.45 ± 63.39	-51.34 ± 158.83
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1242.74 ± 103.53	-40.20 ± 43.20
Cheetah-P	$\mathbf{2563.76} \pm 184.38$	2407.62 ± 144.89	-156.14 ± 39.49
FordA	$\mathbf{96.68} \pm 0.28$	96.40 ± 1.11	-0.28 ± 1.27
StarLightCurves	$\mathbf{99.67} \pm 0.58$	99.33 ± 0.58	-0.33 ± 1.15
UWave	$\mathbf{96.30} \pm 1.70$	$\mathbf{96.30} \pm 0.64$	0.00 ± 1.92
Permuted MNIST	96.87 ± 0.11	$\mathbf{96.89} \pm 0.17$	0.02 ± 0.10
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	58.73 ± 1.90	-1.19 ± 0.63

Mean $\pm$ sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant $-$ SPARC difference. Higher is better; the larger mean in each pair is bold.

E.9 Content nonlinearity and saturation

Replacing $\tanh$ content with the identity removes the bound on projected input amplitudes while preserving both controller nonlinearities and all transition and write factors. This tests whether bounding new content helps learning within the same adaptive recurrence.

On RepeatFirst, mean return falls from 0.829 to 0.123, and the linear variant's sample standard deviation rises to 0.857 (Table 16). The fitted SPARC model is also strongly saturated on this task: $(95.73 \pm 1.79)\%$ of real and imaginary content activations exceed 0.95 in magnitude, compared with $(3.01 \pm 0.15)\%$ on FordA. Bounding content therefore materially changes the write pathway used on RepeatFirst, although these diagnostics alone do not explain the variation between runs. Linear content remains effective elsewhere, improving permuted MNIST from 96.87% to 97.07% and StarLightCurves from 99.67% to 100.00%. These gains show that bounded content is not necessary for strong classification performance with the adaptive recurrence.

E.10 Aggregate comparison and control dimensionality

To summarize performance across tasks with different score scales, we rank each candidate against the same six baseline methods using unrounded means and competition ranks, then average with equal weight over the 16 tasks. SPARC obtains the lowest mean rank, 1.9375, with the retention-only variant closest at 2.0625. This small aggregate gap is consistent with the phase ablation: adaptive phase contributes on particular tasks, while selective retention alone remains a strong alternative.

Table 16: Linear content.

Task	SPARC	Variant	Δ (variant − SPARC)
Autoencode	$\mathbf{-0.431} \pm 0.003$	-0.442 ± 0.012	-0.011 ± 0.009
CountRecall	-0.515 ± 0.098	$\mathbf{-0.465} \pm 0.060$	0.050 ± 0.081
RepeatFirst	$\mathbf{0.829} \pm 0.008$	0.123 ± 0.857	-0.705 ± 0.865
RepeatPrevious	0.855 ± 0.033	$\mathbf{0.867} \pm 0.070$	0.012 ± 0.041
Noisy Pendulum	$\mathbf{0.461} \pm 0.039$	0.425 ± 0.060	-0.037 ± 0.079
CartPole	$\mathbf{0.959} \pm 0.026$	0.944 ± 0.001	-0.015 ± 0.027
HigherLower	$\mathbf{0.500} \pm 0.003$	$\mathbf{0.500} \pm 0.002$	-0.000 ± 0.001
Ant-P	$\mathbf{4507.80} \pm 292.89$	4143.82 ± 100.87	-363.98 ± 393.76
Walker-P	$\mathbf{994.79} \pm 198.05$	921.90 ± 52.32	-72.89 ± 157.41
Hopper-P	$\mathbf{1282.94} \pm 78.06$	1242.39 ± 107.85	-40.55 ± 70.63
Cheetah-P	2563.76 ± 184.38	$\mathbf{2594.39} \pm 31.84$	30.63 ± 152.54
FordA	$\mathbf{96.68} \pm 0.28$	96.12 ± 0.00	-0.55 ± 0.28
StarLightCurves	99.67 ± 0.58	$\mathbf{100.00} \pm 0.00$	0.33 ± 0.58
UWave	$\mathbf{96.30} \pm 1.70$	$\mathbf{96.30} \pm 1.70$	0.00 ± 2.94
Permuted MNIST	96.87 ± 0.11	$\mathbf{97.07} \pm 0.10$	0.19 ± 0.03
Seq. CIFAR-10 Gray	$\mathbf{59.92} \pm 1.41$	59.46 ± 1.06	-0.46 ± 0.50

Mean ± sample SD. Classification: accuracy (%); other tasks: return. Δ: paired variant − SPARC difference. Higher is better; the larger mean in each pair is bold.

Removing the phase clock gives the largest deterioration in mean rank (3.5000); using a static clock partially recovers performance (2.8750). Phase-only and static transitions obtain 3.1875 and 2.5625. Among write interventions, neutral gates, frozen reference gains, linear content, and fixed normalization obtain 2.3750, 2.6250, 2.5000, and 3.1250, respectively. The full design therefore gives the strongest aggregate performance in this panel, even though individual variants improve some tasks. The per-task differences and seed spreads remain necessary for interpreting these ranks.

All measured variants use shared control. Two shared projections require $2(D+1)$ parameters, compared with $2H(D+1)$ for separate controllers per complex mode. Since the unshared variant was not run, this parameter comparison describes the structural saving; the ablations establish the roles of components within SPARC, without measuring the performance cost of sharing itself.

F Checkpoint and Training Diagnostics

F.1 Sampling, aggregation, and diagnostic conventions

Checkpoint replay uses a fixed environment stream from reset for the first 1024 RL steps, or the first eight internal-validation examples and all four classification layers. Statistics are averaged within each independently trained run before aggregating runs. Diagnostics use the same three training seeds as the ablations. Recorded-input interventions hold the observed input stream fixed.

We measure the normalized gate $2\sigma(s_j p_t + v_j r_t)$, reference gain e^{β_j}, and dynamic gain ratio $\sqrt{(1-e^{-2e_{j,t}})/(1-e^{-2\nu_j})}$. The product of the last two factors excludes the gate. A gate value of one is neutral. Entries are run means; temporal standard deviations are calculated within each fragment before run-level aggregation.

F.2 Gate activity and content saturation

The gate and content activation regulate different aspects of writing. The normalized gate can attenuate or amplify projected content, while $\tanh$ bounds each real and imaginary activation. Table 17 combines their activity statistics with their effects on write energy. Gate energy compares the original write with a neutral gate; content energy compares $\tanh$ with linear content while retaining the same gate and gain weights.

RepeatFirst combines substantial gate variation with $(95.73 \pm 1.79)\%$ content saturation. Its gate increases recorded write energy by a factor of 1.816, whereas $\tanh$ retains only 0.76% of the corresponding linear-content energy. FordA has much weaker gate variation and retains 51.45% of linear-content energy. These measurements describe how trained models use the write pathway; the retrained comparisons in Appendix E test whether other parameters compensate when a component is removed.

Table 17: Write-gate activity and content saturation.

Task	Gate SD	Saturation (%)	Gate ratio	Content (%)
Autoencode	0.0913 ± 0.0299	32.67 ± 3.70	1.171	15.49
CountRecall	0.0426 ± 0.0274	23.50 ± 5.50	1.252	25.38
RepeatFirst	0.4508 ± 0.0382	95.73 ± 1.79	1.816	0.76
RepeatPrevious	0.0081 ± 0.0013	31.52 ± 4.34	1.244	16.43
Noisy Pendulum	0.0890 ± 0.0203	75.65 ± 1.48	1.079	4.23
CartPole	0.0072 ± 0.0015	28.33 ± 5.32	1.149	26.49
HigherLower	0.0028 ± 0.0015	0.01 ± 0.02	1.014	75.86
Ant-P	0.0075 ± 0.0060	0.74 ± 0.93	1.020	58.37
Walker-P	0.0044 ± 0.0007	1.20 ± 0.77	1.026	55.15
Hopper-P	0.0148 ± 0.0078	9.82 ± 2.93	1.128	34.79
Cheetah-P	0.0146 ± 0.0020	6.09 ± 0.79	1.032	42.82
FordA	0.0082 ± 0.0044	3.01 ± 0.15	1.011	51.45
StarLightCurves	0.0131 ± 0.0020	1.04 ± 0.72	1.004	55.66
UWave	0.0223 ± 0.0037	4.45 ± 1.42	1.050	47.08
Permuted MNIST	0.0094 ± 0.0024	2.66 ± 0.57	1.006	57.53
Seq. CIFAR-10 Gray	0.0132 ± 0.0018	3.33 ± 1.30	1.008	48.01

Gate SD is the per-mode temporal standard deviation. Gate ratio and content (%) report write-energy ratios. Gate variability and saturation report run means $\pm$ sample SD. Saturation is the fraction of real/imaginary activations with magnitude above 0.95. Energy ratios hold the other write factors fixed.

F.3 Static and dynamic write scales

The reference gain sets a fixed scale for each mode at a checkpoint, while the dynamic gain ratio responds to retention. Their effects on writing depend on the content reaching each mode. For RepeatFirst, the mean reference gain is 3.574, the dynamic gain ratio is 0.324, and their mean product is 1.153; the corresponding FordA values are 0.289, 0.702, and 0.203. The dynamic normalization scales the recorded write energy to 21.2% of its value without this factor on RepeatFirst and 59.2% on FordA.

F.4 Retention and phase activity

Table 18 relates the two shared controls to their effects on the existing state. The decay multiplier rescales the learned baseline decay. History-energy survival is the summed energy after the transition divided by the previous-state energy; its baseline comparison uses the same states and learned spectrum. The phase offset is the mean absolute additional rotation. The clock ratio compares the coupled phase offset with the corresponding uncoupled offset.

RepeatFirst retains 99.12% of the recorded previous-state energy per step while its clock reduces the phase correction to 3.29% of the uncoupled value. Ant-P instead retains 17.71% and has a clock ratio of 45.33%. Shared controls thus produce different temporal responses across tasks and modes, consistent with the distinct removal effects in Appendix E.

Table 18: Retention and phase activity.

Task	Decay scale	Surv. (%)	Rel. surv.	Phase (deg)	Clock (%)
Autoencode	2.769	63.99	1.039	30.553	38.05
CountRecall	0.708	80.46	1.114	17.901	25.30
RepeatFirst	0.255	99.12	1.091	2.963	3.29
RepeatPrevious	0.223	84.66	1.584	6.026	9.74
Noisy Pendulum	4.416	54.92	0.967	30.287	41.12
CartPole	0.623	67.66	1.210	17.442	22.16
HigherLower	1.801	54.09	0.864	17.852	43.13
Ant-P	2.762	17.71	0.465	5.492	45.33
Walker-P	1.782	46.55	0.892	13.414	42.92
Hopper-P	2.856	46.14	0.860	33.468	45.57
Cheetah-P	2.261	35.86	0.708	20.096	49.86
FordA	0.591	95.15	1.043	0.776	3.00
StarLightCurves	1.216	91.40	0.991	1.778	5.77
UWave	0.895	94.86	1.024	1.459	4.02
Permuted MNIST	0.590	96.21	1.032	0.623	2.97
Seq. CIFAR-10 Gray	0.520	96.23	1.042	0.581	2.56

Run-averaged statistics on recorded inputs. Decay scale multiplies the baseline decay; survival measures history energy and is divided by baseline survival in the relative column. Phase is the absolute additional rotation; clock is the coupled-to-uncoupled phase ratio.

F.5 Fixed-input interventions

We intervene at trained checkpoints while holding the parameters and recorded inputs fixed. Neutralizing the gate gives a mean policy KL divergence of 1.7050 on RepeatFirst, compared with 0.0014 on Walker-P; the prediction-distribution KL on FordA is 0.0009. Removing the phase clock gives larger changes of 5.4257, 5.2410, and 3.9080, respectively. These measurements describe the fitted model's response to each intervention. The retrained ablations in Appendix E evaluate the corresponding effects on task performance.

F.6 Depth and training evolution

Layer statistics retain all four classification layers and average eight examples within each run. Training diagnostics compare the first and last 10% of common finite logged environment-step positions, without extrapolation, endpoint rescaling, or treating checkpoints as independent runs. Component activity varies with depth. In FordA, content saturation falls from 9.02% in layer one to 0.75% in layer four, while the temporal standard deviation of the gain ratio rises from 0.1056 to 0.3239. In UWave, the temporal standard deviation of the normalized gate increases from 0.0119 to 0.0315 across the same layers.

During RepeatFirst training, logged saturation rises from 54.152% to 95.625%, gate standard deviation from 0.331 to 0.735, and state RMS from 2.666 to 22.638. HigherLower shows much weaker gate variation (0.008 to 0.010). The logged gate standard deviation differs from the per-mode temporal statistic used in checkpoint replay. The retrained linear-content ablation evaluates its effect on task performance.

G GPU Implementation Details

Algorithm 1 divides time into chunks $\mathcal{C}_k = [s_k, e_k]$ and modes into M tiles, indexed by m. The affine updates $F_{t,m}$ are reconstructed from the shared controls within each tile; Θ denotes the recurrent parameters.

Algorithm 1 Chunked scan with backward replay.

Input: shared controls, writes, $\mathbf{h}_0$, chunks $\{\mathcal{C}_k\}_{k=1}^{K}$
Output: $\mathbf{h}_{1:T}$, $\nabla\Theta$
► **FORWARD PASS**
parallel for $(k, m) \in [K] \times [M]$: $S_{k,m} \leftarrow \mathrm{Compose}(F_{\mathcal{C}_k,m})$
parallel for $m \in [M]$: $\widehat{S}_{1:K,m} \leftarrow \mathrm{PrefixScan}(S_{1:K,m})$
parallel for $(k, m) \in [K] \times [M]$: $\mathbf{h}_{\mathcal{C}_k,m} \leftarrow \mathrm{Replay}(F_{\mathcal{C}_k,m}, \widehat{S}_{k-1,m}(\mathbf{h}_{0,m}))$
► **BACKWARD PASS**
parallel for $(k, m) \in [K] \times [M]$: $\widetilde{S}_{k,m} \leftarrow \mathrm{AdjointSummary}(\mathcal{C}_k, m)$
parallel for $m \in [M]$: $\bar{\mathbf{h}}_{e_{1:K},m} \leftarrow \mathrm{ReverseScan}(\widetilde{S}_{1:K,m})$
parallel for $(k, m) \in [K] \times [M]$: $(\bar{\mathbf{h}}_{s_k-1,m}, \Delta\Theta_{k,m}) \leftarrow \mathrm{AdjointReplay}(\mathcal{C}_k, m, \bar{\mathbf{h}}_{e_k,m})$
$\nabla\Theta \leftarrow \mathrm{Reduce}_{k,m}(\Delta\Theta_{k,m})$

Coefficient generation and layout. The complex state is stored in separate real and imaginary arrays. The two controller projections are evaluated once per token in FP32, producing $B \times T$ arrays for r_t and p_t. Content projections run in parallel over tokens and modes. Each modal tile broadcasts the controls and loads its spectral and write parameters. Content activation, retention-dependent normalization, modal gating, and real–imaginary write conversion are fused into one tiled operation with the layout required by the scan.

Chunk summaries and replay. A GPU program assigned to a time chunk and a modal tile reconstructs the transitions and composes the local updates using Eq. (10). It writes one complex-affine summary (P_k, Q_k) per chunk. A prefix scan gives the state entering each chunk, and local replay produces the complete state sequence. Episode resets set the transition to zero at the reset position, as in Eq. (25). In Algorithm 1, $\widehat{S}_{0,m}$ is the identity map and $\bar{\mathbf{h}}$ denotes a state cotangent.

Backward computation and storage. Each chunk forms a reverse summary of its cotangent propagation. A reverse scan supplies boundary cotangents, and local reverse replay reconstructs transitions and accumulates gradients. Modal contributions are reduced before updating the two token-level control gradients. The scan retains the state outputs needed by BPTT and recomputes transition coefficients during summary and replay. Static spectral values can be cached per mode, and scalar functions of the controls per token. The write backward operation recomputes content activations, the normalization ratio, and the gate from saved inputs and controls, then reduces parameter gradients over token tiles.

Precision and online execution. Inputs and visible outputs may use BF16. Controller projections, spectral parameters, exponentials, trigonometric functions, complex-affine summaries, recurrent accumulation, and reverse adjoints use FP32. Conversion to output precision occurs at write-back boundaries. For an online step, one fused kernel evaluates the two controls, reconstructs the transition and write, updates the real and imaginary states, and emits the visible activation and next FP32 state.

G.1 EFFICIENCY BENCHMARK PROTOCOL

Both implementations use matched tensor shapes, BF16 inputs and outputs, FP32 recurrent accumulation, and gradients for inputs and parameters. The scan benchmark takes precomputed recurrent coefficients as input. The complete-mixer benchmark includes control and coefficient generation, recurrent propagation, and their backward computations. Surrounding projection layers, normalization, feed-forward blocks, loss evaluation, optimizer updates, and multi-GPU communication are outside the timed region. Memory results compare peak allocated GPU memory for the matched training workloads.

The benchmark dimensions are mixer width d_{mix}, batch size B_{batch}, and sequence length T. Figure 4(a) uses $B_{\mathrm{batch}} = 8$ and $d_{\mathrm{mix}} = 1024$, varying T from 2048 to 16384. Panels (b) and (c) fix $B_{\mathrm{batch}}T = 8192$ at widths 2048 and 2560. Panel (d) fixes $T = 2048$ and $d_{\mathrm{mix}} = 2048$, varying batch size from 1 to 32.

For complete recurrent-mixer forward and backward computation, the fixed-token configurations are

$$(B_{\text{batch}}, T) \in \{(4, 2048), (2, 4096), (1, 8192)\}, \qquad B_{\text{batch}} T = 8192. \tag{41}$$

Latency reductions divide by the RG-LRU time at each matched configuration. The plotted bar heights instead use the width-specific RG-LRU time at $T = 2048$ as a common denominator. The batch sweep fixes $T = 2048$ and $d_{\text{mix}} = 2048$; its latency ratio crosses one between batch sizes 2 and 4 and is approximately 0.6 at batch sizes 8 and above. Peak allocated-memory ratios are approximately 0.85–0.87 for most batch sizes.

G.2 Performance Compared with a Naive Implementation

To quantify the computational benefit of the accelerated kernels, we compare the SPARC mixer with an eager PyTorch implementation of the same recurrence. The naive implementation computes the control and write terms, unbinds the time axis once, and propagates the state sequentially; PyTorch automatic differentiation supplies the backward pass. Identical parameters, input tensors, and output cotangents are used for each pair. Both paths accept BF16 inputs, return BF16 outputs, accumulate recurrent states in FP32, and compute input and parameter gradients. An NVIDIA RTX PRO 6000 Blackwell Server Edition GPU runs the measurements with PyTorch 2.8.0, CUDA 12.8, and Triton 3.4.0. After one warmup iteration, three timed iterations yield the reported medians. CUDA synchronization brackets the forward and backward stages, with each backward pass consuming a freshly constructed forward graph. Compilation is excluded, and speedups use the ratio of unrounded naive and accelerated medians.

At a fixed budget of 8192 tokens, longer sequences increase the cost of sequential execution substantially (Table 19). For width 2048, naive backward latency rises from 238.730 ms at $T = 2048$ to 967.711 ms at $T = 8192$, while the accelerated measurements range from 0.911 to 1.289 ms. The resulting backward speedups span 194.81–1061.91×. Width 2560 exhibits a similar trend: its backward speedup increases from 218.45× to 888.98× over the same sequence lengths. Across both widths, combined forward–backward speedups range from 137.13× to 869.62×. The measured gains show the practical benefit of the accelerated execution paths for full-sequence differentiation.

Table 19: SPARC versus its naive implementation at a fixed budget of 8192 tokens.

			Backward			Forward + backward		
B	T	d_{mix}	Naive	Accel.	Speedup	Naive	Accel.	Speedup
4	2048	2048	238.730	1.225	194.81×	358.822	2.617	137.13×
2	4096	2048	479.999	1.289	372.50×	721.780	2.510	287.54×
1	8192	2048	967.711	0.911	1061.91×	1469.467	1.690	869.62×
4	2048	2560	236.191	1.081	218.45×	358.953	1.824	196.78×
2	4096	2560	489.609	1.105	442.89×	740.985	1.846	401.39×
1	8192	2560	969.447	1.091	888.98×	1500.917	1.847	812.84×

Times are medians in milliseconds. Speedups divide the naive time by the accelerated time for the same configuration.

Increasing batch size at fixed $T = 2048$ and $d_{\text{mix}} = 2048$ produces a different scaling pattern (Table 20). Naive backward latency remains within 228.853–265.053 ms, whereas the accelerated path grows from 0.853 ms at $B = 1$ to 5.614 ms at $B = 32$. Backward acceleration therefore falls from 268.35× to 47.21× between these endpoints. Combined speedup reaches 230.69× at $B = 4$ and remains 45.68× at $B = 32$. Together, the two sweeps demonstrate substantial acceleration across the tested sequence lengths and batch sizes, with the largest measured gains occurring in long-sequence, small-batch configurations.

Table 20: SPARC versus its naive implementation across batch sizes, with $T = 2048$ and $d_{\text{mix}} = 2048$.

	Backward			Forward + backward		
B	Naive	Accel.	Speedup	Naive	Accel.	Speedup
1	228.853	0.853	268.35×	344.970	1.900	181.56×
2	240.696	0.942	255.42×	361.919	2.026	178.67×
4	236.798	0.890	265.99×	355.549	1.541	230.69×
8	239.711	1.710	140.21×	365.954	2.548	143.62×
16	244.437	3.044	80.31×	366.240	4.570	80.15×
32	265.053	5.614	47.21×	390.188	8.542	45.68×

Times are in milliseconds. The fixed-token and batch sweeps are measured in separate runs.

G.3 Numerical Agreement with the Naive Reference

Numerical validation uses an independent PyTorch reference that explicitly constructs the complex transition and write terms, advances the state through a sequential loop, and obtains derivatives through automatic differentiation. With $(B, T, d_{\text{mix}}) = (2, 64, 32)$ and FP32 inputs, the accelerated mixer uses the chunked scan backend (chunk size 32). Nonzero random controller and write-gate parameters exercise input-dependent retention, phase, and writing. Both implementations receive matching parameters, inputs, and output cotangents, allowing direct comparison of the forward output, input gradient, and recurrent parameter gradients.

For each tensor z, agreement is summarized by the maximum absolute error and an error normalized by the reference magnitude:

$$E_{\text{abs}} = \|z_{\text{accel}} - z_{\text{ref}}\|_\infty, \qquad E_{\text{norm}} = \frac{E_{\text{abs}}}{\max(1, \|z_{\text{ref}}\|_\infty)}. \tag{42}$$

The denominator keeps the normalization bounded for reference tensors with small magnitude. Table 21 reports both measures for the output and each active gradient block. Acceptance thresholds are 2×10^{-5} for output absolute error and 10^{-4} for normalized gradient error.

Forward outputs agree to a maximum absolute error of 5.66×10^{-7}, and the input-gradient normalized error is 5.28×10^{-7}. Among parameter gradients, the largest normalized error is 1.43×10^{-6} for the retention controller. The log-phase gradient has the largest absolute discrepancy, 7.63×10^{-5}, with a normalized error of 6.58×10^{-7}. The normalized errors for all reported outputs and gradients lie between 1.27×10^{-7} and 1.43×10^{-6}, approximately one to twelve times the FP32 machine epsilon, consistent with the accumulated rounding error induced by the different floating-point evaluation orders of the length-64 sequential recurrence and the chunked scan, and demonstrating agreement between the accelerated computation and the independently differentiated reference within FP32 numerical precision.

Table 21: Numerical agreement between the accelerated SPARC mixer and the naive reference.

Quantity	Maximum absolute error	Normalized error
Forward output	5.66×10^{-7}	5.66×10^{-7}
Input gradient	2.03×10^{-6}	5.28×10^{-7}
Log-decay parameter gradient	4.77×10^{-6}	1.17×10^{-6}
Log-phase parameter gradient	7.63×10^{-5}	6.58×10^{-7}
Phase-controller gradient	6.68×10^{-6}	7.09×10^{-7}
Retention-controller gradient	3.05×10^{-5}	1.43×10^{-6}
Log-write-gain gradient	1.43×10^{-6}	3.32×10^{-7}
Write-phase-response gradient	2.38×10^{-7}	2.38×10^{-7}
Write-retention-response gradient	1.27×10^{-7}	1.27×10^{-7}