# LINEAR-TO-NONLINEAR TRANSITIONS IN THE LEARNING DYNAMICS OF RECURRENT NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

How do recurrent neural networks (RNNs) acquire the temporal computations required by a task? We use the best causal linear (BCL) predictor—determined by input and target second-order statistics—as a task-defined reference for answering this question. First, we connect the BCL predictor to its minimal recurrent realization, providing an exact functional benchmark and task-defined poles. Second, we show that an overparameterized linear RNN realizes this predictor through a small set of residue-carrying modes while most of its spectrum remains behaviorally weak. Recurrent gain accelerates learning only up to a near-marginal regime, where predictive modes approach the BCL poles as nearby unused modes move away. Third, on a nonlinear memory task, a nonlinear RNN reaches BCL-level performance before improving beyond the optimal-linear loss and forming stable memory states around the same slow scaffold. A controlled filtered-cubic task shows that this linear-first ordering depends on the training regime. The BCL thus links task statistics, recurrent dynamics, and the emergence of nonlinear computation during learning.

## 1 INTRODUCTION

Prediction from a sequence of inputs is a core ability of intelligence from AI to biological brains. Autoregressive language models learn by predicting the next token from its preceding context (Bengio et al., 2003); mammals accumulate self-motion signals to navigate (Gil et al., 2018); and influential theories of predictive coding propose that cortical systems interpret sensory input by comparing it with internally generated predictions (Rao & Ballard, 1999). Understanding how such temporal computations form during learning would reveal not only whether a learner solves a task, but what it learns, in what order, and through which internal dynamics.

A central obstacle is that a sequential prediction task constrains an input–output function, not a unique internal mechanism. This ambiguity is amplified by modern training practice, where effective optimization often relies on heavily overparameterized models. Excess capacity can facilitate gradient-based optimization, including in theoretical analyses of RNNs (Allen-Zhu et al., 2019), while dense networks can contain much smaller subnetworks that achieve comparable performance (Frankle & Carbin, 2019). Empirical scaling laws quantify how performance improves with model size, data, and compute (Kaplan et al., 2020; Hoffmann et al., 2022), but do not explain which internal computations are responsible for these improvements or whether additional capacity serves representation, optimization, or both. Moving beyond brute-force scaling therefore requires a mechanistic account of what learning constructs inside these systems. Training loss and raw recurrent spectra cannot provide this account: they do not reveal which temporal structure is required by the task, which network modes implement it, or when the learned computation begins to exploit nonlinear dependencies.

We take a step toward this account with a task-first framework for analyzing learning in RNNs. For a stationary sequential prediction problem under squared loss, the input autocovariance and input–target cross-covariance determine the best causal linear (BCL) predictor through the Wiener–Hopf equations (Wiener, 1949; Kailath et al., 2000). Although the target may depend nonlinearly on its input history, the BCL is the exact optimum among causal linear predictors. We can therefore compare a network's learned input–output map with the BCL at any training checkpoint: proximity to the BCL means that the network reproduces the task's best linear prediction, whereas improvement beyond its



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.

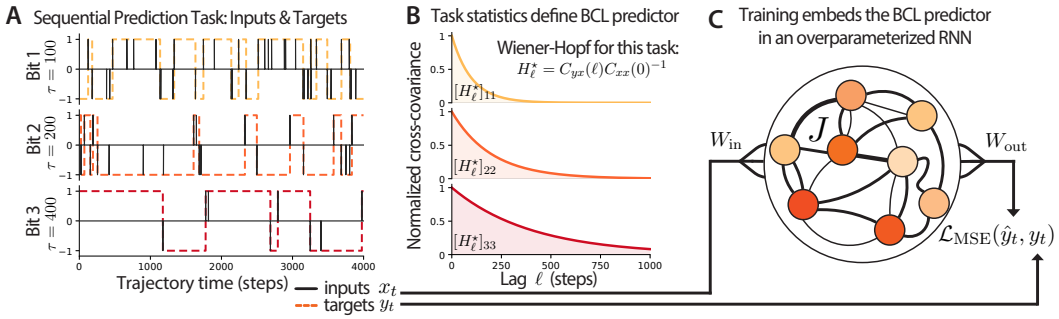



Figure 1: **Task statistics provide a linear reference for recurrent learning. A.** A heterogeneous flip-flop task with three temporal scales. **B.** Second-order task statistics determine the best causal linear (BCL) predictor through the Wiener–Hopf equations. **C.** Training embeds the BCL predictor in an overparameterized RNN, providing a task-defined reference for analyzing recurrent learning.

optimal loss requires predictive structure unavailable to causal linear models. A minimal recurrent realization of the BCL defines BCL poles, while input–output residues identify the predictive modes that implement the same operator in a larger trained network. This isolates the functionally relevant recurrent subsystem from the excess capacity that may facilitate its discovery. Our contributions are:

**Contributions.**

1. **We define a task-based linear reference (fig. 1).** We connect the Wiener–Hopf predictor to its minimal recurrent realization, obtaining an exact functional benchmark and a set of BCL poles determined by the sequential prediction problem rather than by a particular network parameterization.
2. **We show how an overparameterized linear RNN learns this reference (fig. 2).** Residue coupling isolates a small set of task-coupled predictive modes whose timescales approach the BCL timescales as the corresponding behavioral components are acquired. Increasing recurrent gain accelerates learning up to a near-marginal regime, where learning instead becomes slower and more variable. In that regime, the modes that ultimately match the BCL poles move toward them while nearby unused modes shift away, revealing selective spectral reorganization as a candidate mechanism for the gain-dependent learning-speed tradeoff.
3. **We identify a linear-first regime in nonlinear RNN learning (fig. 3).** On a nonlinear memory task, channelwise performance reaches the BCL benchmarks while predictive origin-Jacobian modes acquire BCL-like timescales. These modes later undergo stability transitions; the final network improves below the optimal-linear loss and contains stable nonlinear memory fixed points organized around the same slow scaffold. A controlled filtered-cubic task shows that this ordering depends on the training regime.

Together, these results make the BCL predictor a bridge between task statistics and recurrent dynamics, allowing us to quantify when nonlinear computation emerges during learning.

## 2 Related Work

For stationary sequence prediction, the Wiener–Hopf equations determine the optimal causal linear predictor from the input autocovariance and target–input cross-covariance (Wiener, 1958). Wiener–Volterra expansions extend this task-level description to nonlinear functionals, with orthogonal Hermite orders under Gaussian inputs. Once a linear input–output operator is known, Ho–Kalman realization theory and the SVD-based Eigensystem Realization Algorithm construct a minimal state-space implementation from its Markov parameters (Ho & Kalman, 1966; Juang & Pappa, 1985). We compose these classical results into a task-defined reference: task statistics specify the BCL predictor, and its Hankel structure specifies a minimal recurrent realization against which larger trained networks can be compared.

Deep linear networks learn singular modes on timescales determined by their strength and initialization (Saxe et al., 2014), while early nonlinear learning can sometimes follow a simpler linearized model (Hu et al., 2020). Stagewise acquisition can also reflect hierarchical, algebraic, or statistical structure in the target (Abbe et al., 2021; Bardone & Goldt, 2024; Marchetti et al., 2026; Fischer et al., 2022; Üre et al., 2026). These results motivate, but do not establish, a universal linear-first principle. The same input–output map can also have many recurrent implementations: training can embed low-rank structure within a random spectral bulk (Schuessler et al., 2020), while parameterization and readout scale influence whether learning changes recurrent features (Bauer et al., 2026). Recent analyses of linear RNNs show ordered acquisition of finite-horizon task directions (Proca et al., 2025), the formation of spectral outliers that implement integration and oscillatory filters (Bordelon et al., 2025), and recurrent restructuring that amplifies task-relevant frequencies (Clark et al., 2026). Long timescales can also arise through trainable unit-level constants, random recurrent dynamics, or curriculum-dependent collective mechanisms (Quax et al., 2020; Shi et al., 2025; Khajehabdollahi et al., 2024), and different operational definitions of neural timescale need not isolate the same mechanism (Raut et al., 2020; Spitmaan et al., 2020; Zeraati et al., 2026). Learning can further reorganize recurrent dynamics through stability changes that affect gradients, loss, and the state-space neural tangent kernel (Doya, 1992; Eisenmann et al., 2023; Hazelden & Shea-Brown, 2026). We anchor these phenomena to the stationary causal predictor, using input–output residues to isolate predictive modes and testing the narrower, regime-dependent claim that nonlinear sequence learning passes through its task-defined optimal linear component.

## 3 THE BCL PREDICTOR AS A REFERENCE FOR RECURRENT LEARNING

We study RNNs trained to predict target sequences from input sequences. To separate the computation required by the task from the particular network that implements it, we first define the task's best causal linear (BCL) predictor. We then construct a minimal linear RNN that realizes this predictor and develop the input–output diagnostics used below to analyze much larger trained networks.

### 3.1 CLASSICAL BACKGROUND: THE BCL PREDICTOR

**The BCL predictor** A sequence-prediction task is specified by a distribution $P$ over length-$T$ input–target trajectories $(x_{1:T}, y_{1:T})$, where $x_t \in \mathbb{R}^{d_x}$ and $y_t \in \mathbb{R}^{d_y}$. A causal predictor $f$ maps the inputs available at time $t$ to a prediction $\widehat{y}_t = f(x_{1:t})$. We evaluate it by the mean-squared error

$$\mathcal{L}_{\mathrm{MSE}}(f) \coloneqq \mathbb{E}_{(x_{1:T}, y_{1:T})\sim P}\left[\frac{1}{T}\sum_{t=1}^{T} \|y_t - f(x_{1:t})\|_2^2\right]. \tag{1}$$

For the stationary analysis below, we regard these trajectories as finite windows from centered input and target processes whose second-order statistics depend only on temporal lag. A causal linear filter with coefficients $H = \{H_k\}_{k\geq 0}$ acts on the input history as

$$f_H(x_{\leq t}) = \sum_{k=0}^{\infty} H_k x_{t-k}, \qquad H_k \in \mathbb{R}^{d_y \times d_x}. \tag{2}$$

Let $\mathcal{F}_{\mathrm{lin}}$ denote this class of predictors. The BCL predictor is the task-defined input–output map

$$f_{\mathrm{BCL}} \in \operatorname*{arg\,min}_{f_H \in \mathcal{F}_{\mathrm{lin}}} \mathcal{L}_{\mathrm{MSE}}(f_H), \qquad f_{\mathrm{BCL}} = f_{H^\star}, \tag{3}$$

where $H^\star = \{H_k^\star\}_{k\geq 0}$ denotes one set of optimal filter coefficients. This definition concerns the function implemented by a model, not a particular parameterization of that function.

**The BCL only depends on the task's second-order statistics** The input autocovariance and target–input cross-covariance for a given lag $\ell$ are

$$C_{xx}(\ell) \coloneqq \mathbb{E}[x_t x_{t-\ell}^\top] \in \mathbb{R}^{d_x \times d_x}, \qquad C_{yx}(\ell) \coloneqq \mathbb{E}[y_t x_{t-\ell}^\top] \in \mathbb{R}^{d_y \times d_x}, \tag{4}$$

where the expectations are taken under $P$. By classical Wiener filtering theory (Wiener, 1949; Kailath et al., 2000), the BCL coefficients satisfy the Wiener–Hopf equations

$$C_{yx}(\ell) = \sum_{k=0}^{\infty} H_k^\star C_{xx}(\ell - k), \qquad \ell \geq 0. \tag{5}$$

Thus, the task distribution determines the BCL predictor through the second-order statistics $(C_{xx}, C_{yx})$. The BCL is generally distinct from the unrestricted minimum-MSE predictor, which may be nonlinear.

### 3.2 Theory: From the BCL predictor to a minimal RNN

The Wiener–Hopf equations specify the optimal linear input–output map, but not how that map should be represented by recurrent dynamics. Here, we propose to connect the two using the classical Ho–Kalman realization theorem and its SVD-based Eigensystem Realization Algorithm (ERA) (Ho & Kalman, 1966; Juang & Pappa, 1985). Specifically, we propose the following theorem that specializes this construction to the BCL predictor and makes explicit the task-to-RNN correspondence used throughout our analyses.

**Theorem 1** (Minimal RNN realization of the BCL predictor)**.** *Suppose the block-Hankel matrices associated with $H^\star$ have finite maximum rank $r$. Choose numbers of block rows $n_o$ and block columns $n_c$ and form*

$$[\mathcal{H}_0]_{ij} = H^\star_{i+j}, \qquad [\mathcal{H}_1]_{ij} = H^\star_{i+j+1}, \qquad \operatorname{rank}(\mathcal{H}_0) = r, \tag{6}$$

*where $0 \le i < n_o$ and $0 \le j < n_c$. Let $\mathcal{H}_0 = U_r \Sigma_r V_r^\top$ be its rank-$r$ singular value decomposition and define*

$$\mathcal{O} = U_r \Sigma_r^{1/2}, \qquad \mathcal{C} = \Sigma_r^{1/2} V_r^\top, \qquad J = \Sigma_r^{-1/2} U_r^\top \mathcal{H}_1 V_r \Sigma_r^{-1/2}. \tag{7}$$

*Let $W_{\mathrm{out}}$ be the first block row of $\mathcal{O}$, and let $W_{\mathrm{in}}$ be the first block column of $\mathcal{C}$. Then*

$$H^\star_k = W_{\mathrm{out}} J^k W_{\mathrm{in}}, \qquad k \ge 0. \tag{8}$$

*Consequently, the width-$r$ linear RNN*

$$h_t = J h_{t-1} + W_{\mathrm{in}} x_t, \qquad \widehat{y}_t = W_{\mathrm{out}} h_t \tag{9}$$

*implements the BCL predictor. No linear RNN of width smaller than $r$ can realize the same filter exactly.*

The proof and the full block-matrix construction are given in Appendix A. Combining the Wiener–Hopf characterization with the realization theorem gives the chain

$$\underbrace{P}_{\text{task distribution}} \longrightarrow \underbrace{(C_{xx}, C_{yx})}_{\text{second-order statistics}} \longrightarrow \underbrace{H^\star}_{\text{BCL filter}} \longrightarrow \underbrace{(J, W_{\mathrm{in}}, W_{\mathrm{out}})}_{\text{minimal linear RNN}}. \tag{10}$$

The individual steps draw on classical prediction and realization theory. Our contribution is to use their composition as a task-defined reference for RNN learning: it identifies the smallest recurrent system capable of implementing the task's optimal linear computation and provides an input–output reference for analyzing much larger trained networks.

Our theorem establishes representational possibility. It does not imply that gradient descent will find the BCL predictor, nor does it specify how an overparameterized RNN will embed the minimal realization among its additional recurrent modes. We next define coordinate-invariant comparisons between this reference and a trained network.

### 3.3 Methods: Comparing trained RNNs with the BCL reference

Directly comparing the recurrent matrix of a trained RNN with that of the minimal BCL realization is not meaningful. A minimal realization is unique only up to an invertible change of hidden-state coordinates, while a trained network may be much wider and may contain many modes with little effect on its predictions. We therefore compare trained RNNs with the BCL at three complementary levels: functional behavior, predictive modes and their timescales, and latent-state geometry. Behavioral and representational comparisons are widely used to compare task-trained RNNs (Huang et al., 2025); the BCL additionally provides a task-defined reference for identifying the predictive modes and timescales learned by a network.

More explicitly, under a change of hidden coordinates $h \mapsto Sh$, a linear realization transforms as $(J, W_{\text{in}}, W_{\text{out}}) \mapsto (SJS^{-1}, SW_{\text{in}}, W_{\text{out}}S^{-1})$. The transfer matrix, its poles and residues, and the resulting predictions are unchanged. These objects therefore characterize an equivalence class of realizations rather than a preferred neural basis. Latent-state plots are not invariant in this sense and are used only after fixing an explicit projection.

**1. Functional behavior.** We first ask whether the network has learned the input–output computation specified by the BCL. Define its excess loss as $\Delta\mathcal{L}_{\text{BCL}}(f) \coloneqq \mathcal{L}_{\text{MSE}}(f) - \mathcal{L}_{\text{MSE}}(f_{\text{BCL}})$. For any stable causal linear predictor with filter coefficients $H$, the projection property of the BCL gives

$$\Delta\mathcal{L}_{\text{BCL}}(H) = D^2_{\text{pred}}(H, H^\star) \geq 0, \tag{11}$$

where $D_{\text{pred}}$ is the prediction-weighted distance between input–output operators defined in Appendix B.1. Thus, for linear RNNs, approaching the BCL loss is equivalent to approaching the BCL operator on the task distribution.

For nonlinear RNNs, loss below the BCL value certifies the use of predictive structure unavailable to causal linear models. Equality of losses alone, however, does not establish equality of input–output maps. We therefore use the BCL loss as a behavioral benchmark, not as proof that a nonlinear network implements exactly the same operator.

**2. Predictive modes and timescales.** The one-sided $z$-transform of the BCL filter coefficients defines the transfer matrix $\widetilde{H}^\star(z) = \sum_{k=0}^{\infty} H^\star_k z^{-k}$. We call the poles of $\widetilde{H}^\star(z)$ the *BCL poles*. Each BCL pole corresponds to an eigenvalue $\lambda_j$ of the recurrent matrix that is both controllable ($(w_j^\dagger W_{\text{in}} \neq 0)$ and observable ($(W_{\text{out}} v_j) \neq 0$), where $v_j$ and $w_j$ are the $j^{\text{th}}$ right and left eigenvectors. In a minimal realization, every mode is controllable and observable Kailath (1980), so the BCL poles coincide exactly with the eigenvalues of the minimal recurrent matrix $J^\star$. Here and below, the superscript $\star$ distinguishes BCL quantities from quantities associated with a particular trained network, so $\text{poles}(\widetilde{H}^\star) = \text{spec}(J^\star)$.

This equivalence is a standard result from minimal realization theory (Ho & Kalman, 1966; Kailath, 1980). BCL poles are determined by the input–output map and do not depend on the hidden-state coordinates chosen to realize it.

Network modes and their timescales can be defined for any linear RNN, whether or not it realizes the BCL. Suppose the trained recurrent matrix $J$ is diagonalizable, with eigenvalues $\lambda_j$, right eigenvectors $v_j$, and left eigenvectors $w_j$ normalized by $w_j^\dagger v_\ell = \delta_{j\ell}$. Each eigenpair $(\lambda_j, v_j)$ defines a *dynamical mode*: in the absence of input, a state component along $v_j$ evolves proportionally to $\lambda_j^k$. The network transfer matrix decomposes as

$$\widetilde{H}_{\text{RNN}}(z) = W_{\text{out}}(I - z^{-1}J)^{-1}W_{\text{in}} = \sum_{j=1}^{N} \frac{R_j}{1 - \lambda_j z^{-1}}, \qquad R_j = (W_{\text{out}} v_j)(w_j^\dagger W_{\text{in}}). \tag{12}$$

The residue $R_j$ measures whether mode $j$ is both excited by the input and visible through the readout. A mode can therefore be slow or internally prominent while having negligible effect on prediction. We call a dynamical mode *predictive* when it has appreciable input–output influence under the task distribution, as quantified by the residue-based predictive-strength measure in Appendix B.2.

This terminology distinguishes task-defined and network-defined objects: $\lambda^\star_\ell$ is a BCL pole, whereas $\lambda_j$ is the eigenvalue of a mode in a particular trained network. For tasks with channel-specific BCL components, we assign predictive modes to output channels using residue coupling and refer to the selected modes as *task-coupled modes*. This assignment does not use proximity to the BCL poles; whether a task-coupled mode approaches its corresponding BCL pole is therefore an empirical result of training.

For analyses of the local spectral neighborhood, we instead use a distinct geometric assignment: a one-to-one minimum-distance match between network eigenvalues and BCL poles. We call the selected eigenvalues *pole-matched modes*. Unlike task-coupled modes, pole-matched modes are selected using pole proximity and therefore do not, by themselves, establish input–output relevance. We keep these terms separate below.

For any stable pole or network eigenvalue $\lambda = \rho e^{\mathrm{i}\omega}$, we define

$$T^{\mathrm{decay}}(\lambda) = -\frac{\Delta t}{\log \rho}, \qquad T^{\mathrm{osc}}(\lambda) = \frac{2\pi \Delta t}{|\omega|}, \tag{13}$$

with $T^{\mathrm{osc}} = \infty$ for a positive real value. Applying these definitions to $\lambda_\ell^\star$ gives a ***BCL timescale***; applying them to $\lambda_j$ gives a ***network-mode timescale***. A network-mode timescale is relevant to the learned predictor only when the corresponding mode is predictive.

For a predictive mode $j(\ell)$ assigned to BCL component $\ell$, we quantify spectral agreement by the dimensionless Euclidean distance $d_\ell^{\mathrm{pole}} = |\lambda_{j(\ell)} - \lambda_\ell^\star|$ in the discrete-time spectral plane. We compare both this distance and the corresponding BCL and network-mode timescales during training.

A nonlinear RNN does not generally possess a single global linear transfer function or global set of poles. The nonlinear experiments below use the bias-free voltage-form update

$$h_t = (I - A)h_{t-1} + A[W_{\mathrm{rec}}\phi(h_{t-1}) + W_{\mathrm{in}}x_t], \qquad \widehat{y}_t = W_{\mathrm{out}}\phi(h_t), \tag{14}$$

where $A = \mathrm{diag}(\alpha_1, \ldots, \alpha_N)$ and $\phi = \tanh$; the matched linear network replaces $\phi$ by the identity. Here the weight matrices include the fixed parameterization-dependent scale factors. More generally, for an update $h_t = F_\theta(h_{t-1}, x_t)$, we analyze the local linearization about the operating point used in each experiment. In the nonlinear analyses below, this is the origin Jacobian $J_\theta^{(0)} = \partial_h F_\theta(h, x)|_{(h,x)=(0,0)}$. The corresponding local input and output maps are $B_\theta^{(0)} = \partial_x F_\theta|_{(0,0)}$ and $C_\theta^{(0)} = \partial_h \widehat{y}|_{h=0}$. Its eigenpairs define ***origin-Jacobian modes*** and local network timescales, while residues formed from $(J_\theta^{(0)}, B_\theta^{(0)}, C_\theta^{(0)})$ identify which modes contribute to local prediction. We do not refer to these eigenvalues as global poles of the nonlinear RNN.

**3. Latent-state geometry.** Finally, we examine how the predictive modes identified above are embedded in the network's state space. Because latent geometry depends on the chosen coordinates, we do not interpret it as a coordinate-invariant distance from the minimal BCL realization. Instead, we project network activity, residue-selected modes, activity centroids, and fixed points into a common low-dimensional activity basis. This reveals how a BCL-like linear scaffold is embedded within a trained network and, for nonlinear RNNs, how that scaffold relates to the formation of nonlinear memory states. The projection and fixed-point procedures are described in Appendix C.

These comparisons organize the remaining analyses. Section 4 uses functional agreement and predictive network modes to study how an overparameterized linear RNN learns the BCL reference. Section 5 combines functional, local-dynamical, and geometric diagnostics to determine when nonlinear learning follows a linear-first trajectory and when it departs from that trajectory.

# 4 Results: How Overparameterized RNNs Learn Linear Predictors

We now use the functional and modal diagnostics of Section 3 to ask how training embeds a low-order BCL predictor within a much larger linear RNN. The input–output operator determines prediction, whereas its realization determines which recurrent modes carry that operator and how they are situated within the surrounding spectral bulk.

## 4.1 A controlled multiscale memory task

Our primary benchmark is the heterogeneous signed $n$-bit flip-flop, a widely used task for studying persistent memory in task-trained RNNs (Sussillo & Barak, 2013; Maheswaranathan et al., 2019). We choose it because it is analytically tractable yet contains the ingredients needed here: controlled long task timescales, nonlinear input-dependent memory, and an exact linear reference.

At each time step and independently across channels, the input $x_{t,m}$ is $+1$ or $-1$, each with probability $p_m/2$, and is zero otherwise. The target retains the sign of the most recent nonzero pulse:

$$y_{t,m} = \begin{cases} x_{t,m}, & x_{t,m} \neq 0, \\ y_{t-1,m}, & x_{t,m} = 0. \end{cases} \tag{15}$$

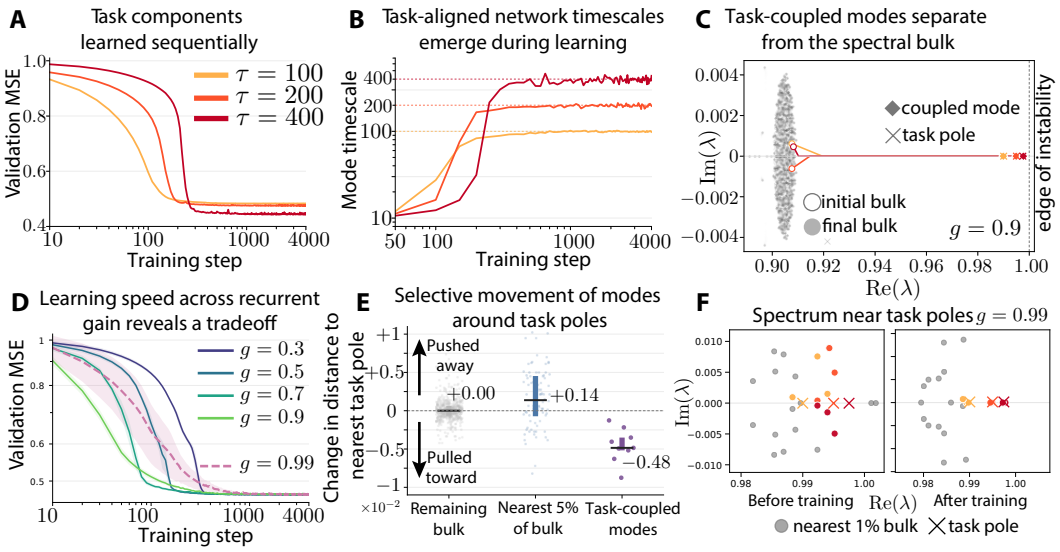



Figure 2: **Learning long timescales selectively reorganizes the recurrent spectrum. A.** Component-wise validation MSE in a representative MSE-trained linear RNN. **B.** Decay timescales of its residue-selected task-coupled modes; dotted lines denote the BCL timescales. **C.** Trajectories of the same task-coupled modes through the initial and final spectral bulk; crosses mark the BCL poles. **D.** Validation MSE across recurrent gains (mean ± SEM; three seeds). **E.** Signed change in distance to the nearest BCL pole at $g = 0.99$; positive and negative values denote movement away from and toward the BCL poles, respectively. **F.** Near-pole spectra before and after training at $g = 0.99$; colored modes are pole-matched, gray modes are the initially nearest $1\%$ of the unused bulk, and black crosses mark the BCL poles.

Different pulse probabilities $p_m$ produce different memory timescales across channels. Because the input is temporally white, the BCL predictor is analytic:

$$\widehat{y}^{\mathrm{BCL}}_{t,m} = (1 - p_m)\widehat{y}^{\mathrm{BCL}}_{t-1,m} + x_{t,m}. \tag{16}$$

Channel $m$ therefore contributes one BCL pole $\lambda^\star_m = 1 - p_m$, with decay timescale $\tau_m = -1/\log(1 - p_m) \approx 1/p_m$ for small $p_m$. Figure 2 uses three channels with nominal timescales $\tau_m \in \{100, 200, 400\}$. The covariance derivation, exact BCL loss, and minimal $n$-dimensional realization are given in Appendix F.1.

### 4.2 Learning a sparse recurrent realization

The Wiener–Hopf solution specifies the task's optimal linear input–output map, but not how an overparameterized recurrent network will realize it. We find that training concentrates this low-order computation in a small set of task-coupled modes while leaving most of the recurrent spectrum behaviorally weak (Figure 2).

**Task-coupled modes emerge with the corresponding behavioral components.** We first examine a representative $N = 512$ linear RNN trained with MSE and initialized with recurrent gain $g = 0.5$ and scale $s = 0.08$. The validation MSEs of the three task components decrease sequentially, in the order $\tau = 100$, 200, and 400 (fig. 2A). Thus, in this representative run, the shorter-timescale components are learned before the longer-timescale components.

To connect this behavioral progression to the recurrent dynamics, we assign a predictive mode to each output at the final checkpoint using matched-channel residue coupling, as in Section 3.3, and track the selected modes backward through optimization using adjacent right-eigenvector overlap. Their network-mode timescales approach the three BCL timescales in the same order as the corresponding component MSEs decrease (fig. 2B). Because time is measured in update steps, the decay timescale in eq. (13) reduces here to $T^{\mathrm{decay}}(\lambda) = -1/\log|\lambda|$.

Panel 2C shows the corresponding spectral picture in the same representative network. The task-coupled modes separate from the initial bulk and approach the analytic BCL poles $\lambda_m^\star = 1 - p_m = \exp(-1/\tau_m)$, while the bulk remains concentrated farther from the edge of instability. Training therefore does not shift the entire spectrum uniformly toward marginal stability; it selectively positions a small number of predictive modes near the poles required by the BCL filter.

**Recurrent gain produces a learning-speed tradeoff.** We next vary the recurrent gain while holding the initialization scale fixed at $s = 1$ in $N = 512$ networks. Across three seeds, increasing the gain from $g = 0.3$ to $g = 0.9$ generally accelerates the decrease in validation MSE (fig. 2D). This trend is consistent with higher-gain initializations placing more candidate modes near the slow BCL poles. However, the near-marginal $g = 0.99$ networks do not continue this trend: they learn more slowly than the $g = 0.9$ networks and exhibit greater variability across seeds. Thus, bringing the initial spectrum closer to the BCL poles is beneficial only up to a point. Because changing $g$ can affect both spectral placement and optimization conditioning, the sweep alone does not identify the cause of this reversal. It motivates testing whether near-marginal networks also reorganize the unused modes surrounding the BCL poles.

**Training selectively clears the spectrum around the BCL poles.** For each $g = 0.99$ network, we first assign one final eigenvalue to each BCL pole by one-to-one minimum-distance matching and track these pole-matched modes backward across checkpoints by minimum displacement. After removing them, we match the remaining eigenvalues between initialization and the end of training by minimizing their total displacement in the complex plane. For each resulting mode trajectory $i$, we compute its signed change in distance to the nearest BCL pole, $\Delta d_i = \min_m |\lambda_i^{\text{final}} - \lambda_m^\star| - \min_m |\lambda_i^{\text{init}} - \lambda_m^\star|$. Thus, $\Delta d_i < 0$ indicates movement toward the nearest BCL pole, whereas $\Delta d_i > 0$ indicates movement away. The points in fig. 2E show individual modes pooled across the three $g = 0.99$ networks; the thick bars and horizontal ticks show the interquartile ranges and medians, respectively.

In the $g = 0.99$ networks, the bulk as a whole moves little (fig. 2E). The pole-matched modes move strongly toward the BCL poles. Importantly, the initially nearest $5\%$ of unused bulk modes show the opposite tendency: their distribution of displacements is shifted away from the poles. The before-and-after spectra make this selectivity visible directly (fig. 2F). Pole-matched modes, colored by their associated BCL timescale, occupy the analytic pole locations after training, whereas the initially nearest $1\%$ of unused modes are more dispersed around their neighborhood.

Together, the residue and spectral analyses support a two-sided picture of realization learning. Optimization concentrates prediction in a small controllable-and-observable subsystem near the BCL poles, while the trained spectrum leaves nearby unused modes farther from those poles. This selective reorganization is a candidate mechanism for the learning-speed tradeoff at very high gain, although the gain sweep does not exclude additional effects from gradient scaling or conditioning.

## 5 Results: A Linear-First Regime in Nonlinear RNN Learning

We now ask whether the optimal linear predictor can appear as an intermediate stage in the training of a nonlinear recurrent network. We identify a regime in which BCL-level behavior and predictive origin-Jacobian modes with BCL-like timescales precede the final nonlinear attractor geometry.

### 5.1 A linear scaffold before nonlinear memory attractors

We train matched bias-free voltage RNNs with tanh or identity activation on a three-channel signed flip-flop task (eq. (14)).

The exact finite-horizon BCL loss provides a channelwise benchmark even though the Bayes-optimal solution is nonlinear. Both networks reach this benchmark, but only the tanh RNN subsequently improves far below it (fig. 3A). During the BCL-level regime, residue-selected origin-Jacobian modes acquire the corresponding BCL timescales (fig. 3B). They later lengthen and cross the unit circle; after the first crossing, the dashed curves denote eigenvector-tracked continuations rather than modes of a stable local transfer function. Selection and continuation details are given in Appendix B.2.1.

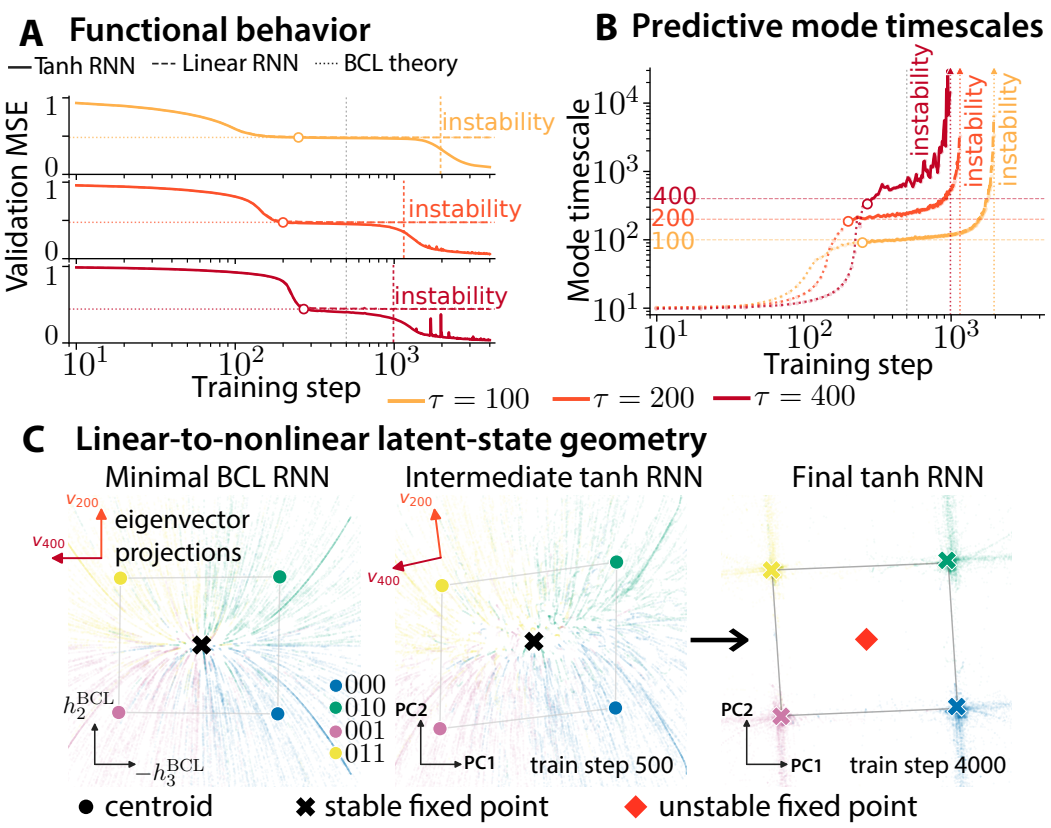



Figure 3: **A BCL-level predictive scaffold precedes nonlinear memory attractors. A.** Per-channel validation MSE for matched tanh (solid) and linear (dashed) RNNs; dotted lines are BCL losses and vertical lines mark stability transitions. **B.** Timescales of channel-selected predictive origin-Jacobian modes. Horizontal lines are BCL timescales; dashed segments continue modes after the origin destabilizes. **C.** Driven activity in the minimal BCL RNN, an intermediate BCL-level tanh RNN, and the final tanh RNN. Circles are state-conditioned centroids, crosses are stable zero-input fixed points, the diamond is the unstable origin, and arrows show slow predictive modes.

The state-space geometry shows how this linear scaffold precedes nonlinear memory (fig. 3C). The intermediate network remains near BCL-level performance with a stable origin and slow predictive directions aligned with those of the minimal BCL realization. By the end of training, the origin is unstable and driven activity is organized around stable memory fixed points, while retaining the same slow scaffold. Projection, sampling, and fixed-point details are given in Appendix C.3.

A controlled filtered-cubic experiment that separates first- and third-order acquisition shows that this linear-first ordering depends on the training regime (Appendix C.1).

## 6 Discussion

Our framework connects task statistics, recurrent realizations, and nonlinear learning through the BCL predictor. Wiener–Hopf equations determine this operator from task statistics, while residues isolate the recurrent modes that carry it. In overparameterized linear RNNs, training concentrates prediction in a small subsystem whose discovery depends non-monotonically on recurrent gain. On a nonlinear memory task, BCL-level behavior and timescales precede improvement beyond the optimal-linear loss and the formation of stable memory states around the same slow scaffold. The filtered-cubic experiment shows that this linear-first trajectory is a training regime, not a universal law.

### AI USE STATEMENT

We used generative AI to assist with theoretical development, hypothesis and experiment design, method implementation, and manuscript drafting and editing. We did not use generative AI to generate, clean, or reformat datasets; or to interpret results. Translation and qualitative analysis were not applicable. The authors reviewed all AI-assisted work, checked mathematical arguments, tested AI-assisted code, and verified claims and citations against original sources. We take responsibility for the final content of this work.

### REPRODUCIBILITY STATEMENT

(This section is **recommended** and does not count toward the page limit.)

It is important that the work published in ICLR is reproducible. Authors are strongly encouraged to include a paragraph-long Reproducibility Statement at the end of the main text (before references) to discuss the efforts that have been made to ensure reproducibility. This paragraph should not itself describe details needed for reproducing the results, but rather reference the parts of the main paper, appendix, and supplemental materials that will help with reproducibility. For example, for novel models or algorithms, a link to an anonymous downloadable source code can be submitted as supplementary materials; for theoretical results, clear explanations of any assumptions and a complete proof of the claims can be included in the appendix; for any datasets used in the experiments, a complete description of the data processing steps can be provided in the supplementary materials. Each of the above are examples of things that can be referenced in the reproducibility statement.

## REFERENCES

Emmanuel Abbe, Enric Boix-Adserà, Matthew Brennan, Guy Bresler, and Dheeraj Nagaraj. The staircase property: How hierarchical structure can guide deep learning. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/e2db7186375992e729165726762cb4c1-Abstract.html.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0ee8b85a85a49346fdff9665312a5cc4-Abstract.html.

Lorenzo Bardone and Sebastian Goldt. Sliding down the stairs: How correlated latent variables accelerate learning with neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 3024–3045, 2024. URL https://proceedings.mlr.press/v235/bardone24a.html.

Jan P. Bauer, Kirsten Fischer, Moritz Helias, and Agostina Palmigiano. A unified theory of feature learning in RNNs and DNNs, 2026. URL https://arxiv.org/abs/2602.15593.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. URL https://www.jmlr.org/papers/v3/bengio03a.html.

Blake Bordelon, Jordan Cotler, Cengiz Pehlevan, and Jacob A. Zavatone-Veth. Dynamically learning to integrate in recurrent neural networks, 2025. URL https://arxiv.org/abs/2503.18754.

David G. Clark, Blake Bordelon, Jacob A. Zavatone-Veth, and Cengiz Pehlevan. Structure, disorder, and dynamics in task-trained recurrent neural circuits, March 2026. URL https://doi.org/10.64898/2026.03.02.708943. ISSN: 2692-8205 Pages: 2026.03.02.708943 Section: New Results.

Fatih Dinc, Marta Blanco-Pozo, David Klindt, Francisco Acosta, Yiqi Jiang, Sadegh Ebrahimi, Adam Shai, Hidenori Tanaka, Peng Yuan, Mark J. Schnitzer, and Nina Miolane. Latent computing by biological neural networks: A dynamical systems framework, February 2025. URL https://arxiv.org/abs/2502.14337. arXiv:2502.14337 [q-bio].

Kenji Doya. Bifurcations in the learning of recurrent neural networks. In *Proceedings of the 1992 IEEE International Symposium on Circuits and Systems*, pp. 2777–2780, 1992. doi: 10.1109/ISCAS.1992.230622. URL https://doi.org/10.1109/ISCAS.1992.230622.

Lukas Eisenmann, Zahra Monfared, Niclas Göring, and Daniel Durstewitz. Bifurcations and loss jumps in RNN training. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-3090. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/df334022279996b07e0870a629c18857-Abstract-Conference.html.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, April 1990. ISSN 0364-0213. doi: 10.1207/S15516709COG1402_1. URL https://doi.org/10.1207/S15516709COG1402_1.

Kirsten Fischer, Alexandre René, Christian Keup, Moritz Layer, David Dahmen, and Moritz Helias. Decomposing neural networks as mappings of correlation functions. *Physical Review Research*, 4(4):043143, 2022. doi: 10.1103/PhysRevResearch.4.043143. URL https://doi.org/10.1103/PhysRevResearch.4.043143.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Mariana Gil, Mihai Ancau, Magdalene I. Schlesiger, Angela Neitz, Kevin Allen, Rodrigo J. De Marco, and Hannah Monyer. Impaired path integration in mice with disrupted grid cell firing. *Nature Neuroscience*, 21(1):81–91, 2018. doi: 10.1038/s41593-017-0039-3. URL https://doi.org/10.1038/s41593-017-0039-3.

James Hazelden and Eric Shea-Brown. State-space NTK collapse near bifurcations, 2026. URL https://arxiv.org/abs/2605.12763.

B. L. Ho and Rudolf E. Kalman. Effective construction of linear state-variable models from input/output functions. *Regelungstechnik*, 14(12):545–548, 1966. doi: 10.1524/auto.1966.14.112.545. URL https://doi.org/10.1524/auto.1966.14.112.545.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c6dfc6b7c601ac2978357b7a81e2d7ae-Abstract.html.

Ann Huang, Satpreet H. Singh, Flavio Martinelli, and Kanaka Rajan. Measuring and Controlling Solution Degeneracy across Task-Trained Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3517. URL https://papers.nips.cc/paper_files/paper/2025/hash/979d5a0ecc2cc3653928c5b793fc416b-Abstract-Conference.html. NeurIPS 2025; preprint: arXiv:2410.03972.

Jer-Nan Juang and Richard S. Pappa. An eigensystem realization algorithm for modal parameter identification and model reduction. *Journal of Guidance, Control, and Dynamics*, 8(5):620–627, 1985. doi: 10.2514/3.20031. URL https://doi.org/10.2514/3.20031.

Thomas Kailath. *Linear Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1980. URL https://books.google.com/books?id=ggYqAQAAMAAJ.

Thomas Kailath, Ali H. Sayed, and Babak Hassibi. *Linear Estimation*. Prentice Hall, Upper Saddle River, NJ, 2000. URL https://books.google.com/books?id=zNJFAQAAIAAJ.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Sina Khajehabdollahi, Roxana Zeraati, Emmanouil Giannakakis, Tim Jakob Schäfer, Georg Martius, and Anna Levina. Emergent mechanisms for long timescales depend on training curriculum and affect performance in memory tasks. In *The Twelfth International Conference on Learning Representations*, May 2024. doi: 10.48550/arXiv.2309.12927. URL https://openreview.net/forum?id=xwKt6bUkXj. ICLR 2024; arXiv:2309.12927.

Timothy Doyeon Kim, Tankut Can, and Kamesh Krishnamurthy. Trainability, Expressivity and Interpretability in Gated Neural ODEs, July 2023. URL https://arxiv.org/abs/2307.06398. arXiv:2307.06398 [cs].

Niru Maheswaranathan, Alex H. Williams, Matthew D. Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks, December 2019. URL https://arxiv.org/abs/1907.08549. arXiv:1907.08549 [cs, q-bio].

Giovanni Luca Marchetti, Daniel Kunin, Adele Myers, Francisco Acosta, and Nina Miolane. Sequential group composition: A window into the mechanics of deep learning, 2026. URL https://arxiv.org/abs/2602.03655. Accepted at ICML 2026.

Francesca Mastrogiuseppe and Srdjan Ostojic. Linking Connectivity, Dynamics, and Computations in Low-Rank Recurrent Neural Networks. *Neuron*, 99(3):609–623.e29, August 2018. ISSN 0896-6273. doi: 10.1016/j.neuron.2018.07.003. URL https://www.cell.com/neuron/abstract/S0896-6273(18)30543-9.

Alexandra M. Proca, Clémentine C. J. Dominé, Murray Shanahan, and Pedro A. M. Mediano. Learning dynamics in linear recurrent neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 49860–49902, 2025. URL https://proceedings.mlr.press/v267/proca25a.html.

Silvan C. Quax, Michele D'Asaro, and Marcel A. J. van Gerven. Adaptive time scales in recurrent neural networks. *Scientific Reports*, 10(1):11360, July 2020. ISSN 2045-2322. doi: 10.1038/s41598-020-68169-x. URL https://www.nature.com/articles/s41598-020-68169-x.

Rajesh P. N. Rao and Dana H. Ballard. Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2(1):79–87, 1999. doi: 10.1038/4580. URL https://doi.org/10.1038/4580.

Ryan V. Raut, Abraham Z. Snyder, and Marcus E. Raichle. Hierarchical dynamics as a macroscopic organizing principle of the human brain. *Proceedings of the National Academy of Sciences*, 117 (34):20890–20897, August 2020. doi: 10.1073/pnas.2003383117. URL https://www.pnas.org/doi/10.1073/pnas.2003383117.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations*, 2014. URL https://arxiv.org/abs/1312.6120.

Friedrich Schuessler, Francesca Mastrogiuseppe, Alexis Dubreuil, Srdjan Ostojic, and Omri Barak. The interplay between randomness and structure during learning in RNNs. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/9ac1382fd8fc4b631594aa135d16ad75-Abstract.html.

Yan-Liang Shi, Roxana Zeraati, International Brain Laboratory, Anna Levina, and Tatiana A. Engel. Brain-wide organization of intrinsic timescales at single-neuron resolution, August 2025. URL https://doi.org/10.1101/2025.08.30.673281. ISSN: 2692-8205 Pages: 2025.08.30.673281 Section: New Results.

H. Sompolinsky, A. Crisanti, and H. J. Sommers. Chaos in Random Neural Networks. *Physical Review Letters*, 61(3):259–262, July 1988. doi: 10.1103/PhysRevLett.61.259. URL https://link.aps.org/doi/10.1103/PhysRevLett.61.259.

Mehran Spitmaan, Hyojung Seo, Daeyeol Lee, and Alireza Soltani. Multiple timescales of neural dynamics and integration of task-relevant signals across cortex. *Proceedings of the National Academy of Sciences*, 117(36):22522–22531, September 2020. doi: 10.1073/pnas.2005993117. URL https://www.pnas.org/doi/10.1073/pnas.2005993117.

David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 25(3):626–649, March 2013. ISSN 0899-7667. doi: 10.1162/NECO_a_00409. URL https://doi.org/10.1162/NECO_a_00409.

Onat Üre, Samet Demir, and Zafer Doğan. Learning beyond the gaussian data: Learning dynamics of neural networks on an expressive and cumulant-controllable data model, 2026. URL https://arxiv.org/abs/2602.02153.

Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. MIT Press, 1949. doi: 10.7551/mitpress/2946.001.0001. URL https://doi.org/10.7551/mitpress/2946.001.0001.

Norbert Wiener. *Nonlinear Problems in Random Theory*. MIT Press, Cambridge, MA, 1958. URL https://mitpress.mit.edu/9780262730129/nonlinear-problems-in-random-theory/.

Roxana Zeraati, Anna Levina, Jakob H. Macke, and Richard Gao. Neural timescales from a computational perspective. *Nature Neuroscience*, 29(7):1534–1547, July 2026. doi: 10.1038/s41593-026-02343-8. URL https://doi.org/10.1038/s41593-026-02343-8. Preprint: arXiv:2409.02684.

## A Proof of Theorem 1

We prove that the BCL predictor admits a minimal linear RNN realization of width $r$, as stated in Theorem 1.

*Proof.* This is the classical Ho–Kalman realization argument (Ho & Kalman, 1966; Juang & Pappa, 1985).

For each $j \geq 0$ and $u \in \mathbb{R}^{d_x}$, define the future-response sequence

$$g_j u \coloneqq \begin{bmatrix} H_j^\star u \\ H_{j+1}^\star u \\ H_{j+2}^\star u \\ \vdots \end{bmatrix},$$

which is an infinite stack of vectors in $\mathbb{R}^{d_y}$. Let

$$\mathcal{X} \coloneqq \operatorname{span}\{g_j u : \ j \geq 0, \ u \in \mathbb{R}^{d_x}\}.$$

The dimension of $\mathcal{X}$ is exactly the rank $r$ of the block-Hankel matrix:

$$\begin{bmatrix} H_0^\star & H_1^\star & H_2^\star & \cdots \\ H_1^\star & H_2^\star & H_3^\star & \cdots \\ H_2^\star & H_3^\star & H_4^\star & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Define the left-shift operator on output sequences by

$$S \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \end{bmatrix}.$$

Because $S g_j u = g_{j+1} u$, the space $\mathcal{X}$ is invariant under $S$. After choosing a basis for the $r$-dimensional space $\mathcal{X}$, define

$$J \coloneqq S|_{\mathcal{X}} \in \mathbb{R}^{r \times r}, \qquad W_{\text{in}} u \coloneqq g_0 u, \ W_{\text{in}} \in \mathbb{R}^{r \times d_x}, \qquad W_{\text{out}} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \end{bmatrix} \coloneqq z_0, \ W_{\text{out}} \in \mathbb{R}^{d_y \times r}$$

Then, for every $u \in \mathbb{R}^{d_x}$ and $k \geq 0$,

$$W_{\text{out}} J^k W_{\text{in}} u = W_{\text{out}} g_k u = H_k^\star u.$$

Hence

$$H_k^\star = W_{\text{out}} J^k W_{\text{in}}, \qquad k \geq 0.$$

Because this realization is both reachable and observable, and $H^\star$ is stable, its recurrent matrix satisfies $\rho(J) < 1$. The resulting RNN therefore implements the same causal linear filter as $H^\star$ and attains the same expected MSE.

Conversely, any width-$N$ RNN realization (i.e., with recurrent matrix $J \in \mathbb{R}^{N \times N}$) satisfies

$$\begin{bmatrix} H_0^\star & H_1^\star & \cdots \\ H_1^\star & H_2^\star & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} W_{\text{out}} \\ W_{\text{out}} J \\ W_{\text{out}} J^2 \\ \vdots \end{bmatrix} \begin{bmatrix} W_{\text{in}} & J W_{\text{in}} & J^2 W_{\text{in}} & \cdots \end{bmatrix}.$$

The right-hand side factors through $\mathbb{R}^N$, so the block-Hankel rank $r$ is at most $N$. Therefore $N \geq r$, proving minimality.

Finally, every width-$r$ realization is reachable and observable because its block-Hankel matrix has rank $r$. The classical uniqueness theorem for minimal realizations therefore implies that any two such realizations are related by an invertible similarity transformation. □

**Example: the heterogeneous $n$-bit flip-flop.** For each channel $m \in \{1, \dots, n\}$, independently across channels and time, let

$$b_{t,m} \sim \text{Bernoulli}(p_m), \qquad \Pr(\xi_{t,m} = +1) = \Pr(\xi_{t,m} = -1) = \tfrac{1}{2}, \qquad x_{t,m} = a\, b_{t,m} \xi_{t,m}. \tag{17}$$

The target stores the sign of the most recent pulse:

$$y_{t,m} = (1 - b_{t,m}) y_{t-1,m} + b_{t,m} \xi_{t,m} = \left(1 - \frac{x_{t,m}^2}{a^2}\right) y_{t-1,m} + \frac{x_{t,m}}{a}. \tag{18}$$

In the stationary regime, the channels are centered and independent, and their second-order statistics are

$$\begin{aligned} C_{xx}(\ell) &= a^2 \operatorname{diag}(p_1, \dots, p_n) \mathbf{1}_{\{\ell = 0\}}, \\ C_{yx}(\ell) &= a \operatorname{diag}\big(p_1(1-p_1)^\ell, \dots, p_n(1-p_n)^\ell\big), \qquad \ell \geq 0. \end{aligned} \tag{19}$$

Because the input is temporally white, the Wiener–Hopf equations can be solved independently at each lag, giving

$$H_k^\star = \frac{1}{a} \operatorname{diag}\big((1-p_1)^k, \dots, (1-p_n)^k\big), \qquad k \geq 0. \tag{20}$$

The associated block-Hankel matrix has rank $n$, and one minimal recurrent realization is

$$J = \operatorname{diag}(1 - p_1, \dots, 1 - p_n), \qquad W_{\text{in}} = \frac{1}{a} I_n, \qquad W_{\text{out}} = I_n. \tag{21}$$

Thus, each channel obeys

$$\widehat{y}_{t,m}^{\text{lin}} = (1 - p_m) \widehat{y}_{t-1,m}^{\text{lin}} + \frac{x_{t,m}}{a}, \tag{22}$$

with one predictive mode of timescale

$$T_m^{\text{lin}} = -\frac{1}{\log(1 - p_m)} \approx \frac{1}{p_m} \tag{23}$$

for small $p_m$. The minimum linear MSE in channel $m$ is

$$\mathcal{L}_{\text{lin},m}^\star = \frac{1 - p_m}{2 - p_m}. \tag{24}$$

The exact flip-flop update is nonlinear because each new pulse must erase the effect of all earlier pulses. The optimal linear predictor cannot implement this input-dependent gating; instead, it replaces perfect persistence by an exponentially decaying memory. This gives both the BCL poles and the exact linear-reference loss used in our later analyses.

Importantly, the target autocovariance alone does not determine this predictor: the relevant temporal structure is the structure in $y_t$ that can be predicted from the input history. Our subsequent analyses ask how training embeds this task-defined filter within an overparameterized RNN and how nonlinear learning eventually departs from it.

## B Predictive-Operator and Modal Diagnostics

This appendix collects the operator-level quantities used to compare a trained recurrent realization with the population-optimal causal linear predictor. These details are not required to derive the Wiener–Hopf solution in the main text.

### B.1 Transfer matrices and prediction-weighted distance

For a stable causal impulse-response sequence $H = \{H_k\}_{k \geq 0}$, define the one-sided transfer matrix

$$H(z) = \sum_{k=0}^{\infty} H_k z^{-k}. \tag{25}$$

The Wiener solution is the orthogonal projection of $y_t$ onto the closed linear span of the input history. Hence, for any other stable causal filter $H$, with $\Delta H = H - H^\star$,

$$\mathcal{L}_{\mathrm{lin}}(H) - \mathcal{L}_{\mathrm{lin}}(H^\star) = \mathbb{E}\left[\left\|\sum_{k=0}^{\infty} \Delta H_k x_{t-k}\right\|_2^2\right]. \tag{26}$$

If the input has spectral density $S_{xx}(\omega)$, the same quantity is

$$D^2_{\mathrm{pred}}(H, H^\star) := \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{tr}\big[\Delta H(e^{\mathrm{i}\omega}) S_{xx}(\omega) \Delta H(e^{\mathrm{i}\omega})^*\big]\, d\omega, \tag{27}$$

so that

$$\mathcal{L}_{\mathrm{lin}}(H) - \mathcal{L}_{\mathrm{lin}}(H^\star) = D^2_{\mathrm{pred}}(H, H^\star). \tag{28}$$

This is the behaviorally relevant distance between two linear predictors: errors are weighted by how strongly the corresponding input directions and frequencies are excited. For temporally white, isotropic inputs it reduces to the squared $\mathcal{H}_2$ distance between the two stable filters.

### B.2 Transfer and residue diagnostics for an RNN

Under the post-update convention in eq. (9), the RNN transfer matrix is

$$H_{\mathrm{RNN}}(z) = W_{\mathrm{out}}(I - z^{-1}J)^{-1} W_{\mathrm{in}}. \tag{29}$$

For a diagonalizable $J$, this becomes

$$H_{\mathrm{RNN}}(z) = \sum_{j=1}^{N} \frac{R_j}{1 - \lambda_j z^{-1}}, \qquad R_j = (W_{\mathrm{out}} v_j)(w_j^\dagger W_{\mathrm{in}}), \tag{30}$$

using the biorthogonal normalization from eq. (12). Thus an eigenvalue of $J$ is a pole of the input–output transfer matrix only if its aggregate residue remains nonzero after cancellations.

To rank modes relative to the task distribution, let

$$G_j(z) := \frac{R_j}{1 - \lambda_j z^{-1}} \tag{31}$$

denote the transfer contribution of a simple mode. Its prediction-weighted energy is

$$s_j^2 := \frac{1}{2\pi}\int_{-\pi}^{\pi} \mathrm{tr}\big[G_j(e^{\mathrm{i}\omega}) S_{xx}(\omega) G_j(e^{\mathrm{i}\omega})^*\big]\, d\omega. \tag{32}$$

For temporally white inputs with covariance $\Sigma_x$ and $|\lambda_j| < 1$,

$$s_j^2 = \frac{\mathrm{tr}(R_j \Sigma_x R_j^*)}{1 - |\lambda_j|^2}. \tag{33}$$

This quantity combines input excitation, output visibility, and persistence. It is therefore more informative than $\|R_j\|_F$ alone. Modal energies need not sum to total output energy because distinct transfer terms can reinforce or cancel; $D_{\mathrm{pred}}$ remains the exact whole-operator comparison. For real systems, complex-conjugate modes are scored jointly.

#### B.2.1 Channel-specific mode selection and continuation in Figure 3

For the signed flip-flop task, input and output channel $i$ correspond to the same bit. Let

$$r_{ij} := e_i^\top R_j e_i = (W_{\mathrm{out}} v_j)_i (w_j^\dagger W_{\mathrm{in}})_i \tag{34}$$

be mode $j$'s matched-channel residue. We partition the eigenspectrum into singleton real modes and complex-conjugate groups $G$. Because channel $i$ is temporally white with variance $p_i$, the group score used in fig. 3B is

$$s_{i,G}^2 = p_i \sum_{j,k \in G} \frac{r_{ij}\overline{r_{ik}}}{1 - \lambda_j \overline{\lambda_k}}, \qquad G_i(t) \in \arg\max_G s_{i,G}^2. \tag{35}$$

This is the matched-channel, conjugate-group specialization of eq. (32); it is not the raw residue magnitude. The score is evaluated only while the origin linearization is stable.

Each channel has its own behavioral anchor

$$a_i = \min\{t \in \mathcal{T}_{\text{saved}} : \mathcal{L}_i(\theta_t) \le \mathcal{L}_{\text{BCL},i}\}. \tag{36}$$

At every stable checkpoint, including $t < a_i$, we select $G_i(t)$ afresh by eq. (35). Thus the pre-anchor curve reports which group is most predictive at each checkpoint and does not impose a persistent eigenvector identity. Eigensolver indices are arbitrary across checkpoints and cannot by themselves diagnose a switch. Starting from the last stable checkpoint $t_s$, where the selected groups are real singletons, we continue the representative vector by

$$j_i(t_{k+1}) = \arg\max_j \left| v_{j_i(t_k)}(t_k)^\dagger v_j(t_{k+1}) \right|, \tag{37}$$

and stop the trace at the first checkpoint with $|\lambda_{j_i}| \ge 1$.

For a stable pole $\lambda_j = r_j e^{\mathrm{i}\theta_j}$, we report the associated decay and oscillation times as

$$T_j^{\text{decay}} = -\frac{\Delta t}{\log r_j}, \qquad T_j^{\text{osc}} = \frac{2\pi\Delta t}{|\theta_j|}, \tag{38}$$

with $T_j^{\text{osc}} = \infty$ for a real positive pole. These are predictively meaningful only when the corresponding residue is non-negligible.

### B.3 A DRIVEN-RESONATOR TASK WITH CONTROLLED COMPLEX POLES

As an input-driven analogue of autonomous sine-wave generation, consider a bank of $M$ damped resonators driven by scalar white noise

$$x_t = u_t, \qquad u_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_u^2). \tag{39}$$

For output channel $m$, define

$$A_m = r_m \begin{bmatrix} \cos\omega_m & -\sin\omega_m \\ \sin\omega_m & \cos\omega_m \end{bmatrix}, \qquad s_{t+1}^{(m)} = A_m s_t^{(m)} + b_m x_t, \tag{40}$$

where $0 < r_m < 1$ and $b_m \in \mathbb{R}^2$. With target

$$y_{t,m} = c_m^\top s_t^{(m)} + \epsilon_{t,m}, \qquad \epsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2 I), \tag{41}$$

the Bayes predictor is linear and has impulse response

$$H^\star_{m,0} = 0, \qquad H^\star_{m,k} = c_m^\top A_m^{k-1} b_m, \quad k \ge 1. \tag{42}$$

For generic $b_m$ and $c_m$, channel $m$ therefore has a known conjugate pole pair $r_m e^{\pm \mathrm{i}\omega_m}$ and known residues.

## C ADDITIONAL NONLINEAR-LEARNING EXPERIMENTS AND DIAGNOSTICS

### C.1 A CONTROLLED FILTERED-CUBIC TASK

The flip-flop couples functional learning to the formation of discrete memory states. To isolate functional order, we introduce a task authored directly in the monomial basis. Let $x_t \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$ and define the normalized probabilists' Hermite polynomials

$$\phi_q(x) = \frac{\mathrm{He}_q(x)}{\sqrt{q!}}. \tag{43}$$

The normalized cubic decomposes as

$$\frac{x^3}{\sqrt{15}} = \sqrt{\frac{3}{5}}\,\phi_1(x) + \sqrt{\frac{2}{5}}\,\phi_3(x). \tag{44}$$

We pass both orthogonal components through the same temporal filter,

$$z_t^{(q)} = \rho z_{t-1}^{(q)} + \sqrt{1-\rho^2}\,\phi_q(x_t), \qquad y_t = \sqrt{\frac{3}{5}}\,z_t^{(1)} + \sqrt{\frac{2}{5}}\,z_t^{(3)}, \qquad \rho = 0.9. \tag{45}$$

Hermite orthogonality makes the first term the optimal linear predictor and the second an orthogonal nonlinear residual. The target has unit variance, so the optimal-linear expected MSE is $2/5 = 0.4$, while the Bayes minimum is zero.

At each checkpoint, we project the network output onto the known filtered $\phi_1$ and $\phi_3$ components. On a fixed validation distribution, we measure recovery of component $q \in \{1, 3\}$ by the projection coefficient

$$a_q(\theta) = \frac{\mathbb{E}[\widehat{y}_t(\theta) z_t^{(q)}]}{\mathbb{E}[(z_t^{(q)})^2]}. \tag{46}$$

The coefficients are estimated after the burn-in interval. Because the two components are orthogonal in population, this de-mixes first- and third-order acquisition even though the network produces a single scalar output. We report these coefficients alongside validation loss.

We vary a global initialization multiplier $s$, applied to the input, recurrent, and output weights, and the strength of the normalized tanh nonlinearity

$$\phi_\beta(u) = \frac{\tanh(\beta u)}{\beta}, \qquad \phi'_\beta(0) = 1, \tag{47}$$

with the $\beta \to 0$ limit equal to the identity. Thus $\beta$ changes the activation's departure from linearity without changing its local gain at the origin.

For $s = 0.5$ and $\beta = 0.5$, the first-order component is learned well before the third-order component, producing a resolved plateau at the optimal-linear loss (fig. 4A). Holding the initialization scale fixed at $s = 1$ reveals a systematic dependence on nonlinearity strength: increasing $\beta$ advances cubic acquisition and shortens the time spent near the optimal-linear solution (fig. 4B). The $\beta = 0$ limit remains at that solution, confirming that the activation nonlinearity controls the later nonlinear phase. Thus, the linear-first trajectory observed in fig. 3 is a regime rather than a universal ordering.

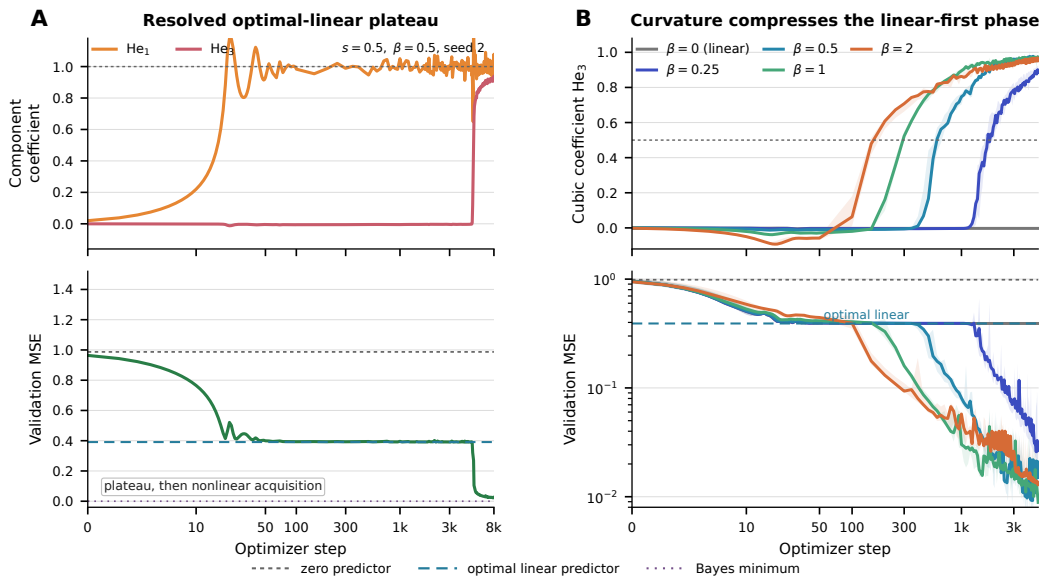



Figure 4: **Increasing activation nonlinearity compresses the linear-first phase.** The filtered-cubic task has an optimal-linear MSE of 0.4. **A.** At $s = 0.5$ and $\beta = 0.5$ (seed 2), the network acquires the first-order component, dwells near the linear optimum, and later acquires the cubic component. **B.** At fixed $s = 1$, increasing $\beta$ advances cubic acquisition and escape below the linear optimum; lines are three-seed medians and bands span the seeds. The $\beta = 0$ limit is linear. Axes use $\log_{10}(t + 1)$ spacing.

### C.2 Fixed points and local stability

At a training checkpoint with parameters $\theta$, zero-input fixed points satisfy

$$h^\star = F_\theta(h^\star, 0). \tag{48}$$

Their local stability is determined by the multipliers of

$$J^\star = \left.\frac{\partial F_\theta(h, 0)}{\partial h}\right|_{h=h^\star}. \tag{49}$$

A multiplier crossing the unit circle marks a loss of local stability. We use the term *bifurcation* only when continuation of the fixed-point branch, together with the appearance of the corresponding attractor or invariant set, establishes a qualitative change in the autonomous dynamics. Otherwise, we refer conservatively to a stability transition.

### C.3 Two-dimensional state-space construction for Figure 3

We drove the minimal BCL RNN and the nonlinear RNN checkpoints with the same 640 deterministic validation sequences. For each network checkpoint, we pooled hidden activity from step 500 and step 4000, fit one PCA basis, and used that basis for both C2 and C3. PC1–PC2 explains 66.4% of the pooled variance. The point clouds in the figure are a common subsample of 80,000 states before conditioning on the displayed memory states; plot limits are chosen independently for legibility.

For the chosen diagonal minimal BCL RNN, the hidden coordinates obey

$$h^{\mathrm{BCL}}_{m,t} = (1 - p_m)h^{\mathrm{BCL}}_{m,t-1} + x_{m,t}, \qquad m \in \{1, 2, 3\}, \tag{50}$$

with units $m = 1, 2, 3$ corresponding to $\tau = 100, 200, 400$. C1 displays the invariant slow plane $\mathrm{span}\{h^{\mathrm{BCL}}_2, h^{\mathrm{BCL}}_3\}$; the horizontal coordinate is plotted as $-h^{\mathrm{BCL}}_3$ to align its visual orientation

with C2–C3. Because this realization is expressed in its eigen-coordinate basis, the $v_{200}$ and $v_{400}$ directions are collinear with the corresponding coordinate axes; $v_{400}$ points opposite to the displayed $-h_3^{\mathrm{BCL}}$ axis. To choose the corresponding two-dimensional view of the trained networks, let $\mu_{b,+}^{(s)}$ and $\mu_{b,-}^{(s)}$ be the state centroids at checkpoint $s$ after averaging over the other two bits, expressed in the first three PCs. For each bit, we compute the retained fraction

$$q_b^{(s)} = \frac{\left\| P_{12}(\mu_{b,+}^{(s)} - \mu_{b,-}^{(s)}) \right\|_2}{\left\| \mu_{b,+}^{(s)} - \mu_{b,-}^{(s)} \right\|_2}, \tag{51}$$

where $P_{12}$ projects onto PC1–PC2. The $\tau = 100$ bit has the smallest retained fraction at both checkpoints, so we omit that direction, fix $y_{100} = -1$, and show the four remaining $(y_{200}, y_{400})$ states in all three views.

Circles in C1–C2 are target-state-conditioned trajectory centroids. Zero-input fixed points are obtained by minimizing $\frac{1}{2}\|h - F_\theta(h, 0)\|_2^2$ and accepting solutions with small residual; their stability is checked from eq. (49). The final network has eight stable fixed points related by the odd symmetry of the bias-free dynamics, of which four are displayed after conditioning on $y_{100} = -1$. Crosses denote stable fixed points: each black cross is the stable origin in C1 or C2, and the colored crosses are the four displayed final memory states. The red diamond denotes the unstable final origin. Arrows show the $v_{200}$ and $v_{400}$ directions: in C1 they are collinear with the analytic BCL coordinate axes, whereas in C2 they are projections of the corresponding channel-selected origin-Jacobian eigenvectors into PC1–PC2. At step 500 the raw-residue and prediction-weighted selectors choose the same three real modes.

## D Overview of RNN Models

A rich model of the dynamics of $N$ recurrent neurons (Quax et al., 2020) is given by the coupled system

$$\begin{aligned} \tau_x \dot{x}(t) &= -x(t) + W r(t) + W^{\text{in}} u(t), \\ \tau_r \dot{r}(t) &= -r(t) + \phi\left(x(t)\right). \end{aligned} \tag{52}$$

where $x_i(t)$ is current into the $i^{\text{th}}$ postsynaptic neuron, $r_i(t)$ is the neuron's firing rate, $W \in \mathbb{R}^{N \times N}$ is the recurrent connectivity matrix, $u(t) \in \mathbb{R}^d$ are the external inputs, $W^{\text{in}} \in \mathbb{R}^{N \times d_{\text{in}}}$ is the input connectivity matrix, and $\phi(\cdot)$ is a fixed activation function. $\tau_x, \tau_r > 0$ are scalar neuronal time constants associated with the current and firing rate activity dynamics, respectively.

The $d_{\text{out}}$-dimensional output of the network is a linear readout of the firing rates, given by

$$y(t) = W^{\text{out}} r(t) \in \mathbb{R}^{d_{\text{out}}}. \tag{53}$$

Two common limiting cases of eq. (52) are widely used. In the first case, we assume that $r$ updates instantaneously compared to $x$, i.e. $\tau_r \to 0$ while $\tau_x$ remains finite, which reduces Eqs. 52 to the *current-based model*:

$$\tau_x \dot{x}(t) = -x(t) + W\phi\left(x(t)\right) + W^{\text{in}} u(t), \tag{54}$$

which is widespread in many mean-field style analyses in theoretical neuroscience (Sompolinsky et al., 1988; Mastrogiuseppe & Ostojic, 2018; Shi et al., 2025; Clark et al., 2026).

In the second case we assume that $x$ updates instantaneously compared to $r$, i.e. $\tau_x \to 0$ while $\tau_r$ remains finite, which reduces Eqs. 52 to the *activity-based model*:

$$\tau_r \dot{r}(t) = -r(t) + \phi\left(x(t)\right) = -r + \phi\left(W r(t) + W^{\text{in}} u(t)\right). \tag{55}$$

This activity-based model is common in task-trained RNNs in computational neuroscience (Dinc et al., 2025; Huang et al., 2025).

Meanwhile, in the AI literature it is standard to take both $\tau_x \to 0$ and $\tau_r \to 0$, giving the discrete-time update rule

$$r_t = \phi\left(W r_t + W^{\text{in}} u_t\right), \tag{56}$$

known as *Elman networks* (Elman, 1990; Quax et al., 2020).

Gated architectures extend this framework with timescales that are modulated by the input at each step Kim et al. (2023). Input-dependent gate activations adjust how much of the previous state is kept vs. overwritten, resulting in context-dependent memory. These architectures have more expressivity but at the cost of higher parameter complexity, and their gating mechanism makes per-neuron timescales less identifiable.

# E MECHANISMS OF LEARNING IN LINEAR RNNS

Some key factors affect spectral timescales at initialization, which can be seen from their effect on the Jacobian spectrum. Fig. 6 shows that increasing the recurrent gain $g$ has the effect of increasing the spectrum disk radius and shifts it closer to the stability boundary, leading to the emergence of long timescales and a power-law timescale distribution Shi et al. (2025). Fig. 7 shows that changing the distribution of neuronal time constants $\tau_i$, in particular introducing heterogeneity in the form of a power-law $\tau_i \sim \tau_i^{-\beta}$ with exponent $\beta$ has the effect of "pinching" the Jacobian spectrum toward the real axis and the stability boundary; as we show in Sec. **??**, this introduces a spectral inductive bias that can accelerate learning for certain tasks characterized by heterogeneous decay timescales. Fig. 8 shows that heavy-tailed recurrent weight distributions scatter the Jacobian spectrum outside of the stability boundary, leading to instability in linear networks and chaos in nonlinear networks, which is likely the primary mechanism behind the emergence of long network timescales in nonlinear networks as observed by Shi et al. (2025).

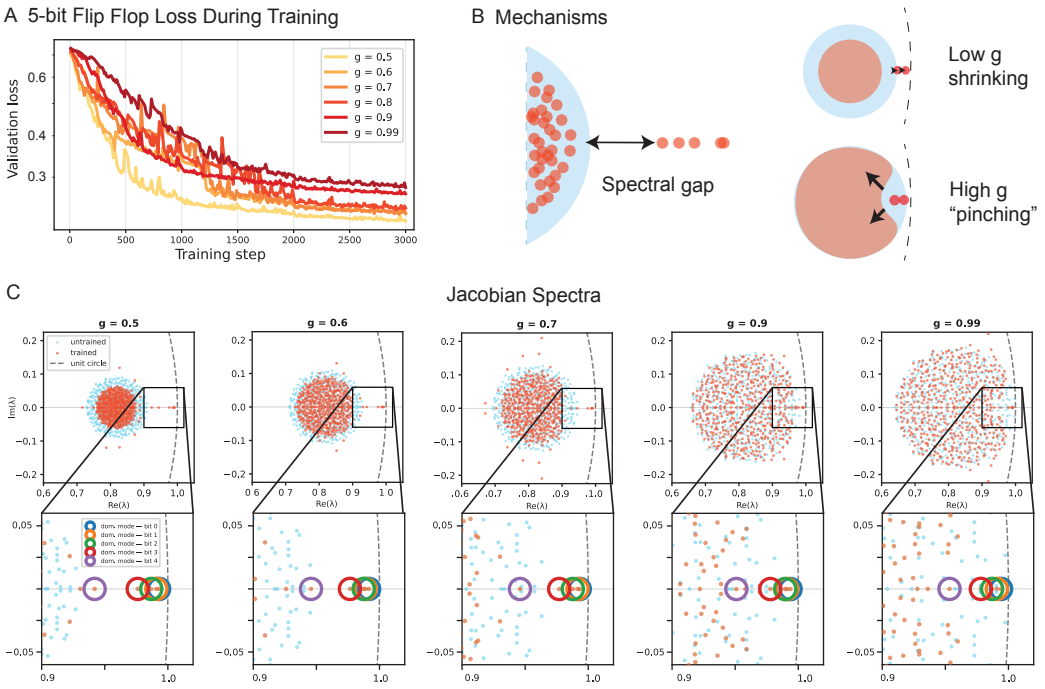



Figure 5: **Mechanisms of spectral reorganization in linear RNNs.** We trained linear RNNs on the 5-bit heterogeneous flip-flop task with pulse probabilities 0.005, 0.007, 0.01, 0.02, and 0.05. The trainable parameters $\Theta = \{W, W_{in}, W_{out}\}$ were jointly optimized and the $\tau_i$ are fixed to be 1. **A.** Validation loss curves for networks with gains $g \in \{0.6, 0.7, 0.8, 0.9, 0.99\}$. Final loss decreases with $g$ indicating that higher gain networks are harder to optimize. **B.** In both regimes, training pulls $n = 5$ slow modes out of the uniform disk-like spectral mass along the real axis, creating a spectral gap. When gain is low, the mass of eigenvalues start in a disk well away from the unit circle stability boundary, so creating the gap occurs from the mass shrinking as well as the necessary eigenvalues traveling outward. When gain is high, the mass of eigenvalues start near the unit circle stability boundary. This means the process of optimization must both shape the mass of eigenvalues and draw the task-relevant ones out. This results in "pinching", which carves out a space for the slow-mode spectra to emerge. **C.** Jacobian spectra before (blue) and after (orange) training for the five gain values. The dashed black line is the unit circle stability boundary. The top row shows the full region of the complex plain with the eigenvalues. The bottom row shows the same plots zoomed into $\mathrm{Re}(\lambda) \in [0.9, 1]$ and $\mathrm{Im}(\lambda) \in [-0.05, 0.05]$. Colored rings indicate the dominant mode for each of the 5 output bits after training. Across all gains exactly five real eigenvalues are pulled to high magnitudes, one per bit, and a gap separates them from the remaining mass disk. At high gains, the gap is narrower and the edge of the mass disk is visibly indented, consistent with the "pinching" phenomenon.

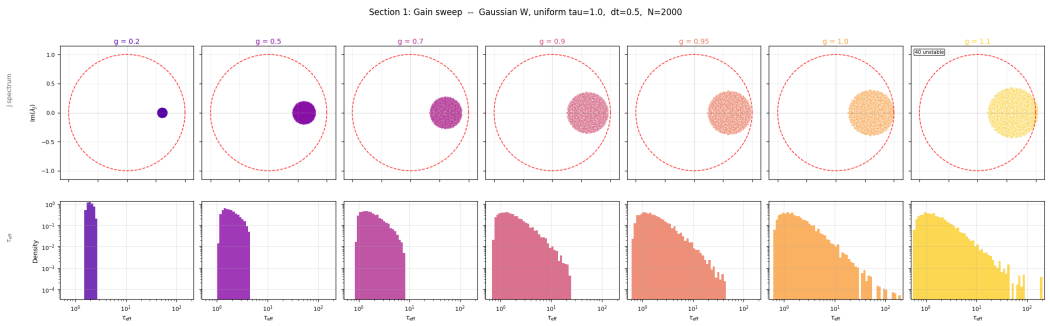



Figure 6: **Effect of varying recurrent gain $g$ on the dynamics of linear RNNs.** All neuronal time constants $\tau_i = 1$ and the recurrent weights are sampled from a Gaussian $W_{ij} \sim \mathcal{N}(0, \frac{1}{N})$ with variance $\frac{1}{N}$. The discretization step is $dt = 0.5$ and there are $N = 2000$ neurons. **Top row:** Jacobian spectrum; red unit circle indicates the stability boundary. Given an eigenvalue of $W$ $\lambda_i^W$ in the unit disc, $J$ has a corresponding eigenvalue $\lambda_i^J = (1-\alpha) + \alpha g \lambda_i^W$. Thus, increasing $g$ has the effect of expanding the spectrum of $J$, centered at $((1-\alpha), 0)$ in the complex plane. As the spectrum expands, more eigenvalues approach the stability boundary, defined by the unit circle. For $g > 1$, some eigenvalues cross the boundary and the linear dynamics become unstable. **Bottom row:** The distribution of decay timescales, defined as $T_i = -1/\log|\lambda_i|$. Increasing $g$ leads to the appearance of long timescales, and a power-law timescale distribution. (Shi et al., 2025)

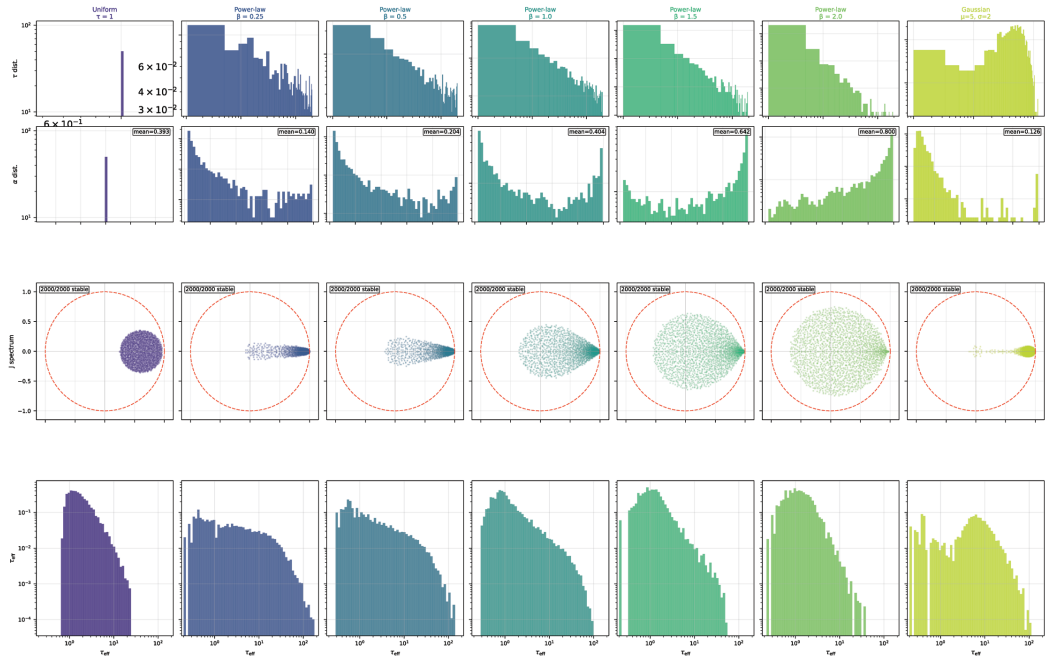



Figure 7: **Effect of varying the distribution of neuronal time constants $\tau_i$.** Fixed recurrent gain $g = 0.9$, $dt = 0.5$, $N = 2000$, and $W_{ij} \sim \mathcal{N}(0, \frac{1}{N})$. **Top rows:** distribution of neuronal time constants $\tau_i$ and corresponding neuron update rates $\alpha_i$. We show a uniform distribution with $\tau_i = 1 \quad \forall i$, 4 power-law distributions $p(\tau_i) \propto \tau_i^{-\beta}$ with varying exponents $\beta \in \{0.25, 0.5, 1.0, 1.5, 2\}$, $\tau_i \in [0.1, 20]$, and a Gaussian distribution $\tau_i \sim \mathcal{N}(5, 4)$ clipped to $[0.1, 20]$. **Middle row:** Jacobian spectra. Heterogeneity in the neuronal time constants has a large impact on the spectrum of the Jacobian; when $\tau_i$ is heterogeneous, the Jacobian $J$ and the recurrent connectivity matrix $W$ do not share an eigenbasis, and there is no simple mapping between eigenvalues of $W$ and eigenvalues of $J$. The power-law distributed $\tau$ enlarge the spectrum of $J$ and shift its center toward the origin; for $\beta = 1$ and $\beta = 2$, it 'pinches' the distribution near the stability boundary into the real axis. The normally-distributed $\tau_i$ collapse the spectrum bulk on the real axis. **Bottom row:** Corresponding decay timescale distribution $T_i$ for each network.

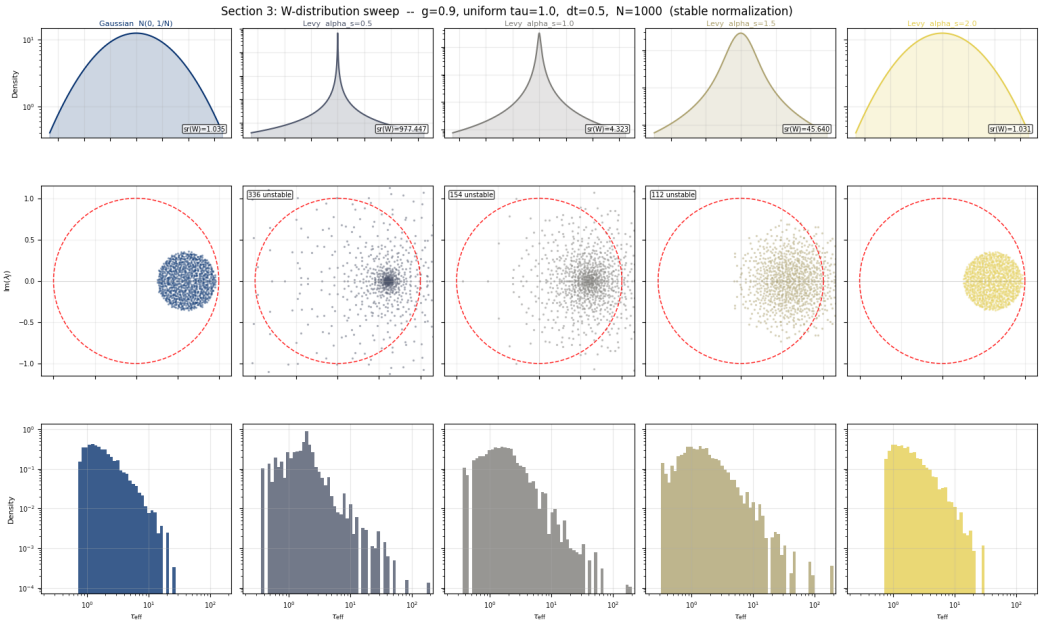



Figure 8: **Effect of varying the distribution of recurrent weights $W_{ij}$.** Fixed recurrent gain $g$, uniform time constants $\tau_i$, $dt = 0.5$, and $N = 2000$. **Top Row:** Distribution of $W_{ij}$. We show a normal distribution with $W_{ij} \sim \mathcal{N}(0, \frac{1}{N})$, and 4 Levy-stable distributions, implemented with `scipy.stats.levy_stable`, with stability parameters $\alpha_s$ (not to be confused with neuron update rates $\alpha_i$) in $\{0.5, 1.0, 1.5, 2.0\}$. Levy-stable distributions have heavy tails and undefined variance for $\alpha_s < 2$; when $\alpha_s = 2$, it corresponds exactly to the normal distribution. **Middle row:** Sampling $W_{ij}$ from heavy-tailed distributions means that the circular law from random matrix theory no longer applies to the spectrum of $W$, and results in a highly dispersed Jacobian spectrum, with many unstable modes. Shi et al. (2025) popose that this is the mechanism behind the emergence of long timescales and power-law timescale distributions in nonlinear RNNs.

# F TASK TIMESCALES

## F.1 THE $n$-BIT FLIP-FLOP TASK

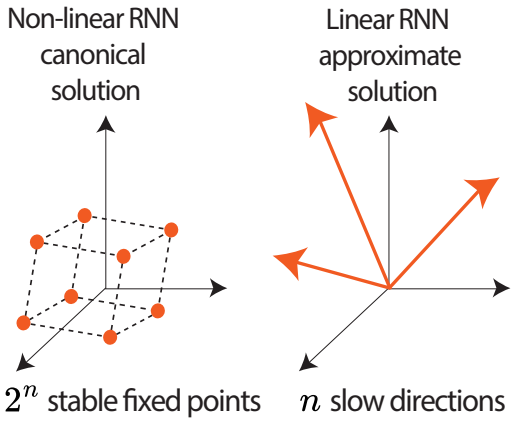



Figure 9: **Different representational geometries solve the same task.** Left: The canonical representation used by nonlinear RNNs to solve the $n$-bit flip flop task is $2^n$ stable fixed points in state space. Right: Linear RNNs learn to approximately solve the $n$-bit flip flop task by learning $n$ slow directions in state space.

We characterize the temporal structure of the heterogeneous $n$-bit flip-flop task by computing the autocorrelations $R_u(\Delta)$ and $R_y(\Delta)$ of its input and target processes. We find that the input is temporally white, while each target bit possesses a pure exponential autocorrelation, yielding a finite set of task timescales $\{T_i^{\text{task}}\}$ given by closed-form expressions in the pulse probabilities $\{p_i\}$.

### F.1.1 DEFINITION OF THE TASK

Consider an $n$-bit flip-flop task in discrete time $t \in \mathbb{Z}_{\geq 0}$. The input $u_t = (u_{t,1}, \ldots, u_{t,n})^\top \in \{-a, 0, +a\}^n$ and task state $x_t = (x_{t,1}, \ldots, x_{t,n})^\top \in \{0,1\}^n$ evolve as follows. For each bit $i \in \{1, \ldots, n\}$, independently across bits and time, the input takes values $\pm a$ each with probability $p_i/2$, and 0 with probability $1 - p_i$, where $p_i \in [0,1]$ is the pulse probability for bit $i$. The state update rule is

$$x_{t,i} = \begin{cases} 1 & \text{if } u_{t,i} = +a, \\ 0 & \text{if } u_{t,i} = -a, \\ x_{t-1,i} & \text{if } u_{t,i} = 0, \end{cases} \tag{57}$$

and the target output is the current state, $y_t^\star = x_t$. A nonzero pulse overwrites bit $i$; a zero input leaves it unchanged. Since the bits evolve independently across channels, it suffices to analyze a single bit.

### F.1.2 INPUT AUTOCORRELATION

Fix a bit $i$. From the input distribution, $\mathbb{E}[u_{t,i}] = 0$ and $\mathbb{E}[u_{t,i}^2] = p_i a^2$. Since the inputs are independent across both time and channels,

$$R_u(\Delta)_{ij} = \mathbb{E}[u_{t,i}\, u_{t+\Delta,j}] = p_i\, a^2\, \delta_{ij}\, \delta_{\Delta 0}. \tag{58}$$

The input is therefore temporally white and channel-diagonal: it has no memory structure beyond lag zero. Any nontrivial timescale in the task must arise from the target state process.

### F.1.3 TARGET AUTOCORRELATION

We now compute the target autocorrelation $R_y(\Delta)$. Since the bits are independent across channels, $R_y(\Delta)$ is diagonal, and it suffices to compute the autocorrelation of a single bit. To exploit the

symmetry of the state distribution, we recode $x_{t,i} \in \{0, 1\}$ as the spin variable

$$z_{t,i} := 2x_{t,i} - 1 \in \{-1, +1\}. \tag{59}$$

The state-update rule (57) translates to

$$z_{t,i} = \begin{cases} +1 & \text{w.p. } p_i/2, \\ -1 & \text{w.p. } p_i/2, \\ z_{t-1,i} & \text{w.p. } 1 - p_i. \end{cases} \tag{60}$$

At each step, with probability $p_i$ the bit is overwritten by a fresh sign with mean zero, and with probability $1 - p_i$ it is preserved. Conditioning on $z_{t,i}$,

$$\mathbb{E}[z_{t+1,i} \mid z_{t,i}] = (1 - p_i)\, z_{t,i}, \tag{61}$$

and iterating yields, for $\Delta \geq 0$,

$$\mathbb{E}[z_{t+\Delta,i} \mid z_{t,i}] = (1 - p_i)^{\Delta}\, z_{t,i}. \tag{62}$$

Under the stationary distribution, $\mathbb{E}[z_{t,i}] = 0$ and $\mathbb{E}[z_{t,i}^2] = 1$, so

$$\mathbb{E}[z_{t,i}\, z_{t+\Delta,i}] = \mathbb{E}[z_{t,i}\, \mathbb{E}(z_{t+\Delta,i} \mid z_{t,i})] = (1 - p_i)^{|\Delta|}. \tag{63}$$

Returning to the original encoding $y^\star_{t,i} = x_{t,i}$, the relation $x_{t,i} - \frac{1}{2} = \frac{1}{2} z_{t,i}$ implies that the normalized autocorrelation is unchanged. Thus

$$\rho_i(\Delta) := \frac{R_y(\Delta)_{ii}}{R_y(0)_{ii}} = (1 - p_i)^{|\Delta|}. \tag{64}$$

#### F.1.4 Task timescales

Writing $\rho_i(\Delta) = \exp(-|\Delta|/T_i^{\text{task}})$ defines the task timescale

$$T_i^{\text{task}} = -\frac{1}{\ln(1 - p_i)}. \tag{65}$$

Since $0 < 1 - p_i < 1$, we have $\ln(1 - p_i) < 0$ and $T_i^{\text{task}} > 0$. For small $p_i$, the expansion $\ln(1 - p_i) \approx -p_i$ yields

$$T_i^{\text{task}} \approx \frac{1}{p_i}. \tag{66}$$

Bits updated less frequently exhibit longer task-induced memory timescales.

In summary, the heterogeneous $n$-bit flip-flop task decomposes into $n$ independent memory channels. The input process is temporally unstructured, and each target bit exhibits a pure exponential autocorrelation, yielding a finite set of *decay* task timescales $\{T_i^{\text{task}}\}_{i=1}^n$. The task contains no oscillatory structure: $T^{\text{osc}} = \infty$.

### F.2 Sine-Wave Generation

For the autonomous sine-wave generation task, the target is a deterministic periodic signal $y_t^\star = \cos(\omega t)$, with no input ($R_u \equiv 0$). Since the target is deterministic, we use the time-averaged autocorrelation,

$$R_y(\Delta) = \lim_{T \to \infty} \frac{1}{T} \int_0^T \cos(\omega t)\, \cos(\omega(t + \Delta))\, dt = \tfrac{1}{2} \cos(\omega \Delta). \tag{67}$$

The normalized autocorrelation,

$$\rho(\Delta) := \frac{R_y(\Delta)}{R_y(0)} = \cos(\omega \Delta), \tag{68}$$

is a pure cosine: persistent and oscillatory, with no decay. The associated *oscillation* task timescale is the period

$$T^{\text{task}} = \frac{2\pi}{\omega}, \tag{69}$$

and the decay timescale is infinite, $T^{\text{decay}} = \infty$.

The sine-wave task is therefore the dual of heterogeneous flip-flop: where flip-flop has decay timescales without oscillation, sine-wave generation has an oscillation timescale without decay. Together, the two tasks span the two extremes of the autocorrelation space considered in this paper.

# G IDENTIFIABILITY OF PER-NEURON TIME CONSTANTS

We expand on the claim, made in the main text, that the per-neuron leak parameter $\alpha_i$ is generically non-identifiable in leaky RNNs: a change in $\alpha_i$ can be absorbed by compensating changes in other parameters that leave the input–output map unchanged. We give two complementary reparameterization symmetries: one for the linear regime, one for ReLU, and connect both to the activation-dependent recovery results reported by Quax et al. Quax et al. (2020).

**Setup.** We consider the activity-based RNN

$$r_{t+1} \;=\; (I - A)\, r_t \;+\; A\, \phi\left(W r_t + W^{\text{in}} u_t + b\right), \tag{70}$$

with $A = \operatorname{diag}(\alpha_1, \ldots, \alpha_N)$ and $\alpha_i \in (0, 1)$.

## G.1 LINEAR REGIME: $\alpha_i$ TRADES OFF WITH DIAGONAL SELF-FEEDBACK

When $\phi$ is approximately linear in the operating regime (e.g., $\tanh(x) \approx x$ near zero), the $i$th update reduces to

$$r_i(t+1) \;\approx\; \left(1 - \alpha_i + \alpha_i w_i\right) r_i(t) \;+\; \alpha_i\, s_i(t), \tag{71}$$

where $w_i \coloneqq W_{ii}$ and $s_i(t)$ collects all off-diagonal, input, and bias contributions. The leak $\alpha_i$ and self-coupling $w_i$ enter only through the combination $1 - \alpha_i + \alpha_i w_i = 1 - \alpha_i(1 - w_i)$.

**Lemma 1** (Linear-regime reparameterization)**.** *For any $w_i \neq 1$, the update is invariant under*

$$\tilde{\alpha}_i \;=\; \alpha_i(1 - w_i), \qquad \tilde{w}_i \;=\; 0, \qquad \tilde{s}_i(t) \;=\; \frac{s_i(t)}{1 - w_i}. \tag{72}$$

*That is, one may eliminate the diagonal self-coupling, replace $\alpha_i$ by $\tilde{\alpha}_i$, and rescale all incoming connections to neuron $i$ (the off-diagonal $i$th row of $W$, the $i$th row of $W^{\text{in}}$, and $b_i$) by $(1 - w_i)^{-1}$, without changing the dynamics.*

The proof is direct substitution. The lemma exhibits a continuous family of $(\alpha_i, w_i, \text{incoming weights})$ triples that produce identical linear dynamics, so $\alpha_i$ is not identifiable from input–output behavior alone. To remove this redundancy in the analyses of the main text, we enforce $\operatorname{diag}(W) = 0$, isolating $\alpha_i$ as the unique self-memory mechanism in the linearized regime.

## G.2 RELU: POSITIVE HOMOGENEITY CREATES A GLOBAL SCALING SYMMETRY

For ReLU, a stronger symmetry holds. ReLU is positively homogeneous: $\operatorname{ReLU}(cx) = c\operatorname{ReLU}(x)$ for $c \geq 0$. Under the joint rescaling

$$W \to cW, \qquad W^{\text{in}} \to cW^{\text{in}}, \qquad b \to cb, \tag{73}$$

the drive term satisfies $\operatorname{ReLU}(c(W r_t + W^{\text{in}} u_t + b)) = c\operatorname{ReLU}(W r_t + W^{\text{in}} u_t + b)$, so the product $\alpha \cdot \operatorname{ReLU}(\cdot)$ admits a one-parameter family of trade-offs: increasing $\alpha$ can be compensated by shrinking $(W, W^{\text{in}}, b)$, and vice versa.

More strongly, on any region of state space where the ReLU gating pattern $D = \operatorname{diag}(\mathbf{1}[z > 0])$ is fixed, the dynamics reduce to an affine system $r_{t+1} = ((1 - \alpha)I + \alpha DW)\, r_t + \alpha D W^{\text{in}} u_t + \alpha D b$, and a change $\alpha \to \alpha'$ can be exactly absorbed by an explicit map on $(W, W^{\text{in}}, b)$. Thus, within a fixed gating regime, $\alpha$ is exactly non-identifiable; across regimes, training can still exploit this freedom by keeping units in stable gating regions or by learning weights that compensate on-average.

## G.3 WHY BOUNDED NONLINEARITIES PARTIALLY RESTORE IDENTIFIABILITY

Saturating nonlinearities such as $\tanh$ and sigmoid are not positively homogeneous and bound the drive term. The local Jacobian along a trajectory is

$$\frac{\partial r_{t+1}^{(i)}}{\partial r_t^{(i)}} \;=\; (1 - \alpha_i) + \alpha_i\, \phi'(z_t^{(i)})\, w_i, \tag{74}$$

and the gain $\phi'(z_t^{(i)})$ varies with the state. Trajectories that explore both the linear (high-gain) and saturating (low-gain) regions of $\phi$ therefore cannot be matched by any single constant rescaling of the drive: the symmetries of Sections G.1 and G.2 hold only in restricted regimes and break globally. ReLU, by contrast, has $\phi'(z) \in \{0, 1\}$ and reduces to a linear system on each gating region, so the symmetry persists.

This predicts the activation-dependent recovery reported by Quax et al. Quax et al. (2020): their sigmoid networks recover $(\alpha_s, \alpha_r)$ from the teacher, while their ReLU networks yield a broad, approximately symmetric loss landscape and fail to recover the target rate constants. The asymmetry of their loss landscape under $\hat{\alpha}_s \leftrightarrow \hat{\alpha}_r$ in the ReLU case is a direct consequence of the linear-regime exchange symmetry between the two leaks (derived for the two-state ARU in their Appendix S2), which our Lemma 1 generalizes to any leak/self-coupling pair in a linearized leaky RNN.

# H Calculation of coupling between task inputs, outputs and computational units

## H.1 Residue-based input–output coupling used in Figure 2

We highlight modes according to their contribution to the network's input–output map, rather than according to eigenvalue magnitude or proximity to a BCL pole. Write the effective discrete-time linear RNN as

$$h_t = Jh_{t-1} + Bx_t, \qquad \widehat{y}_t = Ch_t, \tag{75}$$

where $J$ includes the leak and recurrent gain, $B$ is the effective input map, and $C$ is the readout. Let

$$Jv_j = \lambda_j v_j, \qquad w_j^\dagger J = \lambda_j w_j^\dagger, \tag{76}$$

where $v_j$ and $w_j$ are right and left eigenvectors and $\dagger$ denotes conjugate transpose. We do not require the numerical eigensolver to return a particular normalization. The residue matrix associated with mode $j$ is therefore computed as

$$R_j = \frac{(Cv_j)(w_j^\dagger B)}{w_j^\dagger v_j} \in \mathbb{C}^{d_y \times d_x}. \tag{77}$$

For a diagonalizable $J$, these matrices give

$$CJ^k B = \sum_j R_j \lambda_j^k, \qquad C(I - z^{-1}J)^{-1}B = \sum_j \frac{R_j}{1 - \lambda_j z^{-1}}. \tag{78}$$

Thus $Cv_j$ measures observability through the readout, $w_j^\dagger B$ measures controllability from the inputs, and their outer product measures their joint input–output effect. The expression is invariant to reciprocal rescalings of the left and right eigenvectors.

For the signed heterogeneous flip-flop, input channel $o$ is paired with target and output channel $o$. We therefore quantify the matched-channel coupling of mode $j$ by the diagonal residue energy

$$c_{oj} = \left|[R_j]_{oo}\right|^2. \tag{79}$$

Because a real matrix has complex modes in conjugate pairs, we treat each real mode or conjugate pair as a single group $G$. Its coupling to output $o$ is

$$S_{oG} = \sum_{j \in G} \left|[R_j]_{oo}\right|^2. \tag{80}$$

To display one distinct mode group per output, we solve the maximum-weight bipartite assignment

$$\pi^\star = \underset{\pi \text{ one-to-one}}{\arg\max} \sum_{o=1}^{d_y} S_{o,\pi(o)}. \tag{81}$$

The colored trajectory for output $o$ in fig. 2B–C marks the eigenvalue or conjugate pair in $\pi^\star(o)$; the same-colored cross marks the analytic BCL pole $\lambda_o^\star = 1 - p_o$. Pole proximity is not included in

eq. (81). Consequently, agreement between a mode and its BCL pole is an empirical property of the learned realization, not an outcome imposed by the visualization.

This raw residue score is used only for the output-specific highlighting in fig. 2B–C. It is not the prediction-weighted modal strength of eq. (32): $S_{oG}$ does not weight modes by their persistence or by the input covariance. A normalized specialization score,

$$\widetilde{S}_{oG} = \frac{S_{oG}}{\sum_{G'} S_{oG'}}, \tag{82}$$

may be reported as a within-output residue share, but modal contributions can interfere, so it should not be interpreted as a fraction of output variance.

### H.2 Pole matching and spectral displacement used in Figure 2

Panels 2E–F use a geometric selection distinct from residue coupling. At the final checkpoint, we assign one eigenvalue to each BCL pole by minimizing the summed absolute distance over one-to-one assignments. We then track those pole-matched modes backward through the saved checkpoints by the analogous minimum-displacement assignment. After removing the three selected modes, we match the remaining initial and final eigenvalues by minimizing their total displacement in the complex plane. The "nearest $5\%$" and "nearest $1\%$" subsets are defined by each unused mode's initial distance to the closest BCL pole. Because the pole-matched modes are selected geometrically, their proximity to the BCL poles is not an independent measure of functional coupling; functional identification in panels B–C instead uses the residue assignment above.

### H.3 Auxiliary basis-dependent coupling measures

The linear network state $x_t$ can be written in Schur coordinates as $x_t = Qq_t$. In these coordinates, the autonomous evolution is given by $q_{t+\Delta t} = Hq_t$. We can therefore compute the coupling between these Schur modes and the input and output dimensions: the output $y_t = W^{\text{out}}x_t = W^{\text{out}}Qq_t$. Therefore, we define the coupling strength between the $i^{\text{th}}$ output dimension $y_i$ and the $j^{\text{th}}$ Schur mode to be $|(W^{\text{out}}Q)_{ij}|$. Similarly, the coupling between the $i^{\text{th}}$ input dimension and the $j^{\text{th}}$ Schur mode is $|(Q^\top AW^{\text{in}})_{ij}|$. Note there is a slight asymmetry between the two expressions; this is because the output is an instantaneous readout of the current state, while the input affects the state update, and therefore the input coupling term involves the update rates $A$.

We compute coupling along three independent axes: the *basis* in which modes are defined, the *method* by which coupling is quantified, and the *direction* (input $\to$ mode vs. mode $\to$ output).

### H.4 Basis

Beyond the Schur basis $Q$ defined above, we consider two additional bases. The *eigenbasis* $V$ diagonalises the Jacobian, $J = V\Lambda V^{-1}$, with mode coordinates $e_t = V^{-1}x_t$; its columns are generally complex and appear in conjugate pairs, but each pair carries the same coupling magnitude and is treated as a single mode. The *neuron basis* is the raw hidden state itself (no change of basis), with mode index $k$ identified with unit $k$. The Schur basis is our default because its columns are real and orthonormal, and $2\times2$ diagonal blocks of $H$ transparently encode complex-conjugate eigenvalue pairs.

### H.5 Connectivity coupling

Purely structural: derived from the weight matrices alone, independent of any trajectories. With $\mathcal{B} \in \{Q, V, I\}$ selecting Schur, eigen, or neuron basis, let $L_{\mathcal{B}}$ denote the associated coordinate map: $L_Q = Q^\top$, $L_V = V^{-1}$, and $L_I = I$. The mode-to-output and input-to-mode connectivity couplings are

$$C^{\text{out}}_{\mathcal{B}} = \left|W^{\text{out}}\mathcal{B}\right|, \qquad C^{\text{in}}_{\mathcal{B}} = \left|L_{\mathcal{B}}AW^{\text{in}}\right|. \tag{83}$$

In the neuron basis ($\mathcal{B} = I$) these reduce to $|W^{\text{out}}|$ and $|AW^{\text{in}}|$.

### H.6 CORRELATION COUPLING

Observational: measures whether a mode's activity actually co-varies with the input or target signals over sampled trajectories. Given hidden-state trajectories $\{x_t^{(n)}\}$ across $N_{\text{traj}}$ trials, we project to the desired basis, $z_t^{\mathcal{B}} = L_{\mathcal{B}} r_t$, and compute the Pearson correlation with the target $y_t^*$ or the input $u_t$ over pooled $(n, t)$ samples,

$$R^{\text{out}}_{\mathcal{B},ij} = \big|\text{corr}\big(z^{\mathcal{B}}_{j,t},\, y^*_{i,t}\big)\big|, \qquad R^{\text{in}}_{\mathcal{B},ij} = \big|\text{corr}\big(z^{\mathcal{B}}_{j,t},\, u_{i,t}\big)\big|. \tag{84}$$

Unlike connectivity, correlation depends on the actual driven dynamics and can reveal that structurally-coupled modes carry no task-relevant variance (or vice versa).

### H.7 ABLATION COUPLING

Interventional: the counterfactual drop in task performance when a mode's contribution to the readout is silenced post-hoc. Because $y_t = W^{\text{out}} x_t$ is linear, silencing mode $j$ in basis $\mathcal{B}$ is equivalent to zeroing the readout column $c^{\mathcal{B}}_{:,j} = W^{\text{out}} \mathcal{B}_{:,j}$:

$$\tilde{y}_t^{(-j)} \;=\; y_t \;-\; z^{\mathcal{B}}_{j,t}\, c^{\mathcal{B}}_{:,j}, \tag{85}$$

and the causal coupling to output $i$ is the induced change in per-bit accuracy,

$$\Delta^{\text{out}}_{\mathcal{B},ij} = \text{acc}_i(y) - \text{acc}_i\big(\tilde{y}^{(-j)}\big). \tag{86}$$

For conjugate pairs (eigenbasis) or $2\times2$ Schur blocks, we silence the two partner columns jointly. Ablation is the closest analogue to a do-calculus intervention available without retraining: it tests "what would the output be if mode $j$'s readout contribution were zero?"

**Remarks.** The three methods answer complementary questions — *structural* ("is the mode wired to the output?"), *observational* ("does it co-vary with the output?"), and *causal* ("does it actually drive the output?") — and need not agree. We quantify concentration of each coupling across modes via the participation ratio $\text{PR}(c) = (\sum_j c_j^2)^2 / \sum_j c_j^4$, where $\text{PR} = 1$ indicates perfect localization onto a single mode and $\text{PR} = N$ a uniform spread.

## I IMPLEMENTATION AND TRAINING DETAILS

**Architecture.** The main-text experiments use the bias-free voltage-form RNN in eq. (14) with hidden width $N = 512$. The linear models use the identity activation and the nonlinear models use $\tanh$. All units have fixed neuronal time constant $\tau_i = 1$, with $dt = 0.1$ and $\alpha_i = 1 - e^{-dt/\tau_i}$. Layer normalization is disabled, recurrent noise is zero, and no zero-diagonal constraint is imposed on the recurrent matrix.

**Tasks.** Figures 2 and 3 use the three-channel signed flip-flop task with pulse probabilities $p_m \in \{0.01, 0.005, 0.0025\}$, corresponding to nominal timescales $1/p_m \in \{100, 200, 400\}$ steps. Pulses have amplitude $\pm1$, trajectories have length $T = 2000$, and the initial hidden state is zero. Models are trained with component-averaged MSE. Each training batch contains 200 newly generated trajectories, and validation uses 2000 trajectories generated from a fixed validation seed.

**Initialization.** The representative network in fig. 2A–C and the networks in fig. 3 use recurrent gain $g = 0.5$, recurrent initialization scale $s = 0.08$, and readout scale $\gamma_{\text{out}} = 0.03$. The controlled gain sweep in fig. 2D–F uses $s = 1$, $\gamma_{\text{out}} = 0.03$, and $g \in \{0.3, 0.5, 0.7, 0.9, 0.99\}$. Input, recurrent, and readout parameters are trainable; neuronal time constants are fixed.

**Optimization.** All main-text conditions use Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$) with learning rate $10^{-3}$ for 4000 optimization steps. No learning-rate schedule or weight decay is used. Training uses full-sequence BPTT in fp32 without gradient clipping, and validation is evaluated every 10 optimization steps.

**Replicates and checkpoints.** The gain-sweep curves and high-gain endpoint analyses in fig. 2D–F use three seeds per gain. Checkpoints are saved every 50 optimization steps for the gain sweep and every 10 steps for the matched linear–nonlinear experiment in fig. 3. Panels that display a learning trajectory from a representative network are identified as such in their captions.

**Compute resources.** The experiments were run on NVIDIA GPUs.