ZNO: Stable Rational Neural Operators in the Z-Domain for Discrete-Time Dynamics

Xianli Zhu
School of Mathematical Sciences
Fudan University
Shanghai, China
25210180115@m.fudan.edu.cn

Jia Yin*
School of Mathematical Sciences
Fudan University
Shanghai, China
jiayin@fudan.edu.cn

Abstract

We introduce the *Z-Domain Neural Operator* (ZNO), a causal neural operator for discrete-time dynamics derived from the z-transform. Each ZNO layer realizes a low-rank rational transfer matrix in the one-step delay z^{-1}: hidden channels are projected onto a few latent channels, filtered by banks of pole-residue modes with an optional finite impulse response (FIR) branch, and mixed back alongside a pointwise skip. Poles are parameterized inside the unit disk, so every layer is stable by construction and is evaluated exactly by a causal recurrence whose cost is linear in sequence length. This z-domain rational recurrent layer brings classical rational filtering into the neural-operator template and connects it with diagonal state-space models. On synthetic discrete-time identification tasks with about 8.5k parameters, ZNO is more accurate than Fourier and Laplace neural operators, has the lowest error on resonant and nonlinear autoregressive dynamics, and remains competitive with the S5 state-space model, which leads on a high-order IIR cascade; this pattern largely holds from 4k to 32k parameters. As the poles approach the unit circle, ZNO is more accurate than FNO and S4D in every memory regime, and at four times the training length its error stays nearly unchanged and remains the lowest, whereas FNO's grows by up to $8.1\times$. On five public nonlinear system-identification benchmarks, ZNO is the most accurate neural model on three. ZNO is thus well suited to stable, lightly damped discrete-time dynamics and complements state-space and classical identification methods.

1 Introduction

Neural operators learn mappings between infinite-dimensional function spaces from data [5, 14, 16, 19]. Two transform-based families are prominent: the *Fourier Neural Operator (FNO)* [16], which parameterizes kernel integrals by learnable Fourier coefficients on the unit circle, and the *Laplace Neural Operator (LNO)* [4], which parameterizes each layer by trainable continuous-time (s-plane) poles and residues. These parameterizations are most natural when the target operator is defined on a continuous time or space domain, as in ODE/PDE settings, and the discrete samples arise from numerical discretization. Many systems of interest, however, are intrinsically discrete-time or observed at a fixed sampling interval: autoregressive moving-average (ARMA) models, digital infinite and finite impulse response (IIR/FIR) filters [20], sampled control plants, and nonlinear autoregressive models with exogenous inputs (NARX) [3, 17]. Their classical model class is the stable discrete-time rational transfer function, a rational function of z^{-1} with all poles strictly inside the unit circle. This paper builds a neural operator around that model class.

*Corresponding author: jiayin@fudan.edu.cn.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

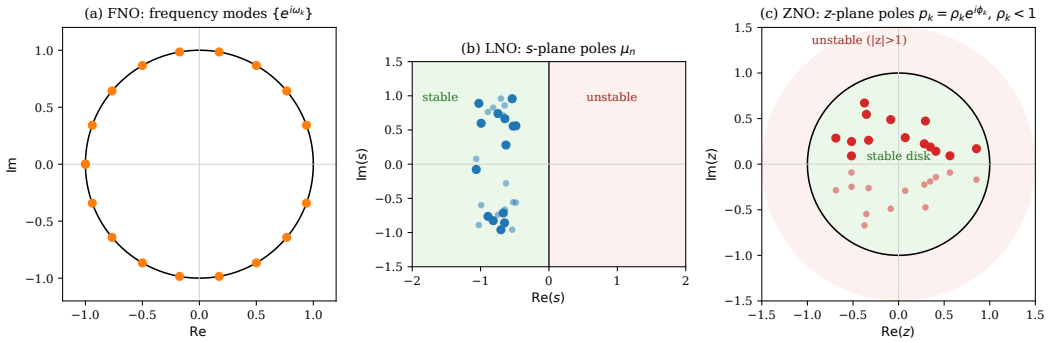


Figure 1: Spectral geometries of FNO, LNO and ZNO. (a) FNO parameterizes the transfer function at equally spaced points on the unit circle. (b) LNO parameterizes continuous-time poles μ_n in the s-plane, whose left half is the stable region. (c) ZNO parameterizes discrete-time poles p_k inside the unit disk of the z-plane; the unit circle is the natural stability boundary for sampled sequences.

From s to z. A continuous-time pole $\mu \in \mathbb{C}$ with $\mathrm{Re}(\mu) < 0$ has the decaying impulse response $\mathrm{e}^{\mu t}$; sampled with step Δt, it becomes the discrete-time pole $p = \mathrm{e}^{\mu \Delta t}$ inside the unit disk. The z-transform describes sequences directly: z^{-1} is a delay of one step, a mode is a geometric sequence p^n, and it decays when $|p| < 1$ [20]. When the data are discrete-time from the start, there is no natural Δt to invert, and a continuous-time parameterization must adopt an arbitrary step and an exponential map to turn the sign constraint on $\mathrm{Re}(\mu)$ into the bound $|p| < 1$. Parameterizing poles directly in the z-plane expresses stability by the unit disk and makes the learned poles readable in discrete-time coordinates (Figure 1). The radius of a pole has a direct meaning: a mode with $|p| = \rho$ decays with time constant $\tau \approx -1/\log \rho$ steps, about 10 steps for $\rho = 0.9$ and 100 for $\rho = 0.99$. Poles near the unit circle thus describe lightly damped oscillations and long memory, as in resonant mechanical and electrical systems, digital filters, and sampled control loops.

The ZNO layer. ZNO is built from the transfer-function side. A ZNO layer projects a hidden sequence with w channels onto $r \ll w$ latent channels, filters each latent channel with its own bank of stable discrete-time poles and an optional short FIR branch, and mixes the filtered streams back to w channels, in parallel with a pointwise skip. In transfer-function form, $\boldsymbol{H}(z) = \boldsymbol{A}\,\mathrm{diag}(G_1(z), \ldots, G_r(z))\,\boldsymbol{B}$ with rational scalar filters G_a, and the rest of the layer follows from this form: the pole-residue expansion supplies the long-memory modes, the FIR part supplies direct feedthrough and short memory, the unit disk supplies stability, and the inverse z-transform supplies an exact causal recurrence whose parameters do not depend on the sequence length. Pole-residue models, unit-disk stability, and diagonal linear recurrences are classical in signal processing and central to modern state-space models (Section 2); ZNO assembles them into one operator layer, which we compare with FNO, LNO, S4D [8], and the closest multiple-input multiple-output (MIMO) diagonal state-space model, S5 [27].

Contributions.

1. **A z-domain rational recurrent layer.** A stable low-rank MIMO rational layer with r latent pole-residue filters, conjugate-pair poles constrained to the unit disk, an optional FIR branch, and real input/output mixing, evaluated exactly by a causal recurrence whose cost is linear in sequence length (Section 4). A fused CUDA/Triton scan [28] accelerates the recurrence (Appendix C).
2. **A synthetic benchmark at matched budgets.** On three discrete-time identification tasks with about 8.5k parameters per model, ZNO is more accurate than FNO and a stabilized LNO on all tasks, has the lowest error on a resonant ARMA task and a nonlinear NARX task, and trails S5 on a sixth-order IIR cascade. The pattern largely persists from 4k to 32k parameters (Section 5.1).
3. **Where the advantage lives.** In a five-bin pole-radius sweep, ZNO has lower error than FNO and S4D in every memory regime, with the gap narrowing for the longest memories, and it remains stable at $4\times$ the training length (Sections 5.2–5.3).

4. **Public benchmarks.** On five public nonlinear system-identification benchmarks [1], ZNO has the lowest neural error on Silverbox, Wiener–Hammerstein, and coupled electric drives (CED); S4D is best on Cascaded Tanks and a classical model on EMPS. Diagnostics of the two negative cases point to the bounded radius map and to pole initialization (Section 5.4).

Organization. Section 2 reviews related work and Section 3 the z-transform background. Section 4 defines the ZNO layer, Section 5 presents the experiments, and Section 6 concludes. The appendices give implementation details, training protocols and configurations, and additional results.

2 Related work

Neural operators. Operator learning builds on universal approximation results for nonlinear operators [5] and includes DeepONet [19] and the integral-kernel neural operators [14]. FNO [16] learns multipliers on a truncated set of Fourier modes; its global spectral convolution acts on the whole input window and is therefore not causal in time. LNO [4] uses a pole-residue expansion with continuous-time poles, which on sampled data implies a choice of sampling interval. ZNO keeps the lift–layer–projection template of these models but replaces the spectral layer by a stable discrete-time rational filter evaluated causally.

Rational models and system identification. Digital IIR/FIR filters, partial-fraction (pole-residue) expansions, and the unit-circle stability criterion are classical [20], and rational approximation of measured frequency responses by pole-residue models, as in vector fitting [11], is standard engineering practice. System identification fits ARX/ARMAX, Wiener–Hammerstein, and NARMAX structures to input-output records [3, 17, 24]; we use SysIdentPy [15] NARMAX estimators as classical baselines. Ljung et al. [18] discuss the relation between deep learning and system identification, and deep subspace encoders [2] learn neural state-space models whose initial state is estimated from past input-output data. ZNO brings the pole-residue form, with an explicit unit-disk map, into a residual operator layer with low-rank latent routing.

Diagonal state-space models. S4 [9] parameterizes structured linear state-space models in continuous time, and DSS [10] and S4D [8] simplify the state matrix to a diagonal one; all three discretize with a learned step. S4D applies an independent single-input single-output (SISO) system to each channel and mixes channels pointwise. S5 [27] replaces the bank of SISO systems by one MIMO diagonal system per layer with dense complex input and output matrices, evaluated by a parallel scan. The linear recurrent unit (LRU) [21] parameterizes the eigenvalues of a diagonal complex recurrence directly in discrete time, keeping their magnitudes below one through an exponential map and initializing them near the unit circle. Rational transfer function (RTF) parameterizations [22] represent a state-space layer by the coefficients of its transfer function and evaluate it without materializing the state. ZNO belongs to this family. It combines a pole-residue transfer function, a real rank-r input/output bottleneck shared by banks of conjugate poles, an optional FIR branch, and a dense pointwise skip in one operator layer (Section 4).

3 Background

Discrete-time operator learning. Let T be the sequence length, $n \in \Omega = \{0, \dots, T-1\}$ the time index, d_u and d_y the input and output channel dimensions, and θ the model parameters. We use $\boldsymbol{u} \in \mathbb{R}^{T \times d_u}$ to denote a forcing and $\boldsymbol{y} \in \mathbb{R}^{T \times d_y}$ to denote a response. Given N i.i.d. training pairs $\{(\boldsymbol{u}^{(j)}, \boldsymbol{y}^{(j)})\}_{j=1}^{N}$, we learn a causal operator $\mathcal{G}_\theta \colon \boldsymbol{u} \mapsto \boldsymbol{y}$, with $y_n = (\mathcal{G}_\theta \boldsymbol{u})_n$ depending only on $u_{0:n}$, by minimizing the example-averaged unsquared relative Frobenius norm

$$\widehat{R}(\theta) \;=\; \frac{1}{N} \sum_{j=1}^{N} \frac{\|\mathcal{G}_\theta \boldsymbol{u}^{(j)} - \boldsymbol{y}^{(j)}\|_{\mathrm{F}}}{\|\boldsymbol{y}^{(j)}\|_{\mathrm{F}}}, \tag{3.1}$$

where $\|A\|_{\mathrm{F}} = (\sum_{n,c} A_{n,c}^2)^{1/2}$ is the Frobenius norm over the time–channel array. Each example is normalized by its own norm; this equals the relative ℓ_2 norm of the flattened sequence, and we report it as the relative ℓ_2 error ($\mathrm{rel}\,\ell_2$). It corresponds to the relative error reported in [4, 16], and the same metric is used at test time.

z-transform and pole-residue form. For a causal sequence $\{f_n\}_{n\geq 0}$, the z-transform is defined as $F(z) = \sum_{n\geq 0} f_n z^{-n}$ [20], so multiplication by z^{-1} corresponds to a delay of one step. A causal linear time-invariant (LTI) transfer function with simple poles admits the pole-residue decomposition

$$G(z) \;=\; \sum_{k=1}^{K} \frac{c_k}{1 - p_k z^{-1}} \;+\; \mathrm{FIR}(z^{-1}), \tag{3.2}$$

where the pole-residue terms provide the long-memory IIR response and $\mathrm{FIR}(z^{-1}) = \sum_{j=0}^{F} g_j z^{-j}$ is a finite impulse response branch capturing direct feedthrough and short-memory effects; for $F = 0$ the FIR branch is omitted. Bounded-input bounded-output (BIBO) stability is equivalent to the condition $|p_k| < 1$ [20] for all k. For the kth summand in (3.2), the impulse response is $c_k p_k^n$ for $n \geq 0$. Hence, this mode decays exactly when $|p_k| < 1$, with time constant $\tau_k \approx -1/\log|p_k|$ steps.

Causal recurrence. Introducing a complex state $s_n^{(k)}$ for each pole $k = 1, \ldots, K$, driven by an input sequence b_n,

$$s_n^{(k)} \;=\; p_k\, s_{n-1}^{(k)} \;+\; b_n, \qquad y_n \;=\; \mathrm{Re}\sum_{k=1}^{K} c_k\, s_n^{(k)} \;+\; (\mathrm{FIR} * b)_n, \tag{3.3}$$

realizes (3.2). Indeed, with zero initial state $s_{-1}^{(k)} = 0$, the z-transform of the state update gives $S^{(k)}(z) = p_k z^{-1} S^{(k)}(z) + B(z)$, that is, $S^{(k)}(z) = B(z)/(1 - p_k z^{-1})$, so that $Y(z) = \sum_k c_k S^{(k)}(z) + \mathrm{FIR}(z^{-1})B(z) = G(z)B(z)$. Poles and residues occur in conjugate pairs (and at most one real pole), so the sum of the pole-residue responses $c_k s_n^{(k)}$ is real and the real part in (3.3) only makes this explicit. The recurrence holds pointwise in n, so the same parameters (p_k, c_k) define a valid operator at every sequence length $T' \geq 1$.

A common state-space view. S4D, S5, and ZNO are all diagonal linear state-space layers,

$$\boldsymbol{x}_n = \bar{\boldsymbol{\Lambda}}\boldsymbol{x}_{n-1} + \bar{\boldsymbol{B}}\boldsymbol{u}_n, \qquad \boldsymbol{y}_n = \mathrm{Re}(\tilde{\boldsymbol{C}}\boldsymbol{x}_n) + \boldsymbol{D}\boldsymbol{u}_n, \tag{3.4}$$

with diagonal $\bar{\boldsymbol{\Lambda}}$, and differ in how the matrices are parameterized and shared across channels. S4D and S5 start from continuous-time eigenvalues $\boldsymbol{\Lambda}$ discretized with learned steps, $\bar{\boldsymbol{\Lambda}} = \exp(\boldsymbol{\Lambda}\boldsymbol{\Delta})$; S4D uses one SISO system per channel, and S5 one MIMO system per layer with dense complex $\bar{\boldsymbol{B}}$ and $\tilde{\boldsymbol{C}}$ and a diagonal feedthrough $\boldsymbol{D}$ [27]. ZNO starts from the transfer function (3.2): its modal realization $\bar{\boldsymbol{\Lambda}} = \mathrm{diag}(p_1, \ldots, p_K)$, $\bar{\boldsymbol{B}} = \mathbf{1}$, $\tilde{\boldsymbol{C}} = (c_1, \ldots, c_K)$ is exactly (3.3), so its poles are discrete-time from the outset, and the FIR branch adds a finite-memory direct path. Section 4 organizes these modes into latent banks, and Appendix A gives the matrix form of a full ZNO layer.

4 The Z-Domain Neural Operator

Network. Following the lift/transform/project template of FNO and LNO, a depth-L ZNO network is

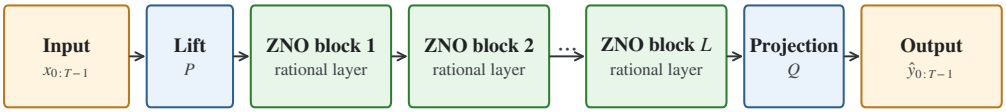
$$\boldsymbol{y} \;=\; Q \circ \sigma \circ \mathcal{T}^{(L)} \circ \cdots \circ \sigma \circ \mathcal{T}^{(1)} \circ P(\boldsymbol{u}), \tag{4.1}$$

with pointwise lift P, projection Q, smooth pointwise nonlinearity σ (GELU for discrete-native experiments), and residual layers $\mathcal{T}^{(\ell)} = \mathcal{R}^{(\ell)} + \mathcal{W}^{(\ell)}$. Here $\mathcal{R}^{(\ell)}$ is the low-rank MIMO rational branch defined below, while $\mathcal{W}^{(\ell)}$ is a dense pointwise skip with matrix $\boldsymbol{W}^{(\ell)}$, that is, $(\mathcal{W}^{(\ell)}\boldsymbol{h})_n = \boldsymbol{W}^{(\ell)}\boldsymbol{h}_n$. Throughout this paper, w denotes the hidden width and L the network depth. The synthetic experiments use $w = 20$ and $L = 4$ unless stated otherwise; the capacity study rescales w, and every configuration is listed in Appendix D. Figure 2 illustrates the architecture.

Low-rank MIMO rational branch. The dynamic branch acts on $\boldsymbol{h}^{(\ell)} \in \mathbb{R}^{T\times w}$ via a low-rank factorization:

$$\mathcal{R}^{(\ell)}(\boldsymbol{h}^{(\ell)})_n \;=\; \boldsymbol{A}^{(\ell)}\,\boldsymbol{q}_n^{(\ell)}, \quad \boldsymbol{q}_n^{(\ell)} = \mathrm{Re}\sum_{k=1}^{K} \boldsymbol{c}_k^{(\ell)} \odot \boldsymbol{s}_n^{(\ell,k)} + \mathbf{1}_{F>0} \sum_{j=0}^{F} \boldsymbol{g}_j^{(\ell)} \odot \boldsymbol{b}_{n-j}^{(\ell)}, \tag{4.2}$$

(a) Causal network with per-time-step residual ZNO blocks



(b) One residual ZNO layer: low-rank rational branch plus skip

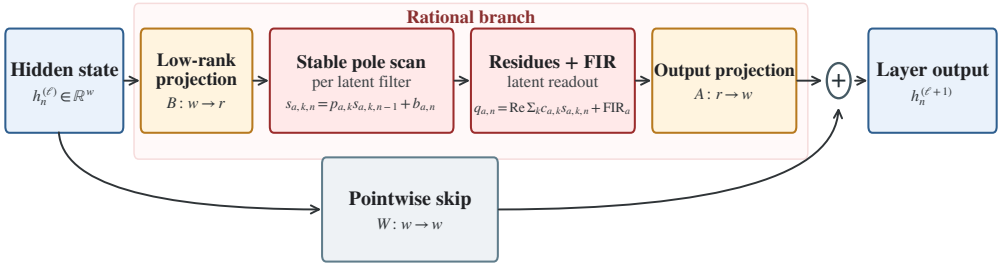


Figure 2: The ZNO architecture. (a) The full causal network: pointwise lift P, L residual ZNO layers with pointwise nonlinearities, pointwise projection Q. (b) One ZNO layer: the hidden sequence is projected to a low-rank stream; each latent rank channel is fed through its own bank of conjugate-pair stable poles, read out through learnable residues and an optional short FIR branch, projected back by $A : r \to w$, and added to the parallel pointwise skip $W h_n^{(\ell)}$.

$$\boldsymbol{s}_n^{(\ell,k)} \;=\; \boldsymbol{p}_k^{(\ell)} \odot \boldsymbol{s}_{n-1}^{(\ell,k)} \;+\; \boldsymbol{b}_n^{(\ell)}, \qquad \boldsymbol{b}_n^{(\ell)} \;=\; \boldsymbol{B}^{(\ell)}\, \boldsymbol{h}_n^{(\ell)}, \qquad \boldsymbol{s}_{-1}^{(\ell,k)} = \boldsymbol{0}, \tag{4.3}$$

where $\boldsymbol{B}^{(\ell)} \in \mathbb{R}^{r\times w}$, $\boldsymbol{A}^{(\ell)} \in \mathbb{R}^{w\times r}$, $\boldsymbol{W}^{(\ell)} \in \mathbb{R}^{w\times w}$ is the matrix of the skip operator $\mathcal{W}^{(\ell)}$, $\boldsymbol{p}_k^{(\ell)}, \boldsymbol{c}_k^{(\ell)} \in \mathbb{C}^r$, $\boldsymbol{g}_j^{(\ell)} \in \mathbb{R}^r$, and $\odot$ is the entrywise product. Each latent rank channel therefore has its own scalar pole-residue filter. The pole count K denotes total poles per latent filter. Since the implementation uses $\lfloor K/2 \rfloor$ conjugate pairs and, if K is odd, one additional real pole, the sum is real by construction. The FIR sum covers lags $0, \dots, F$; setting $F = 0$ omits the FIR branch entirely. The transfer matrix of the rational branch $\mathcal{R}^{(\ell)}$ factors as

$$\boldsymbol{H}^{(\ell)}(z) = \boldsymbol{A}^{(\ell)} \operatorname{diag}\Big(G_1^{(\ell)}(z), \dots, G_r^{(\ell)}(z)\Big)\, \boldsymbol{B}^{(\ell)}, \tag{4.4}$$

with each $G_a^{(\ell)}$ of the form (3.2). Thus the MIMO dynamics are diagonal in the latent filter space and dense only through the input/output mixing matrices $\boldsymbol{B}^{(\ell)}$ and $\boldsymbol{A}^{(\ell)}$, rather than assigning an independent recurrent filter to every input-output channel pair. The layer's parameter count and cost are given in Appendix B.

Relation to S5. Stacking the latent banks writes $\mathcal{R}^{(\ell)}$ in the diagonal state-space form (3.4) with $r\lceil K/2 \rceil$ complex modes, each latent input driving only its own bank (Appendix A). Where an S5 layer lets every mode read from and write to all channels through dense complex matrices, ZNO routes the dynamics through a real r-dimensional bottleneck, so the transfer matrix (4.4) of the rational branch has rank at most r at every frequency; the skip $\mathcal{W}^{(\ell)}$ adds a dense static path.

Stable pole parameterization. Each pole entry is stored as $(\tilde{\rho}_{a,k}^{(\ell)}, \phi_{a,k}^{(\ell)}) \in \mathbb{R}^2$ for latent channel a and pole index k, and mapped by

$$\rho_{a,k}^{(\ell)} = \rho_{\max} \operatorname{sigmoid}(\tilde{\rho}_{a,k}^{(\ell)}), \qquad p_{a,k}^{(\ell)} = \rho_{a,k}^{(\ell)}\, \mathrm{e}^{\mathrm{i}\phi_{a,k}^{(\ell)}}, \tag{4.5}$$

where $\operatorname{sigmoid}(x) = (1 + \exp(-x))^{-1}$ and $\rho_{\max} = 0.999$ is a fixed hard stability ceiling. This smooth map sends the unconstrained parameters onto $(0, \rho_{\max})$, enforces the unit-disk constraint, and eliminates the need for a hard projection step. We also define a smooth pole-safety regularizer

$$\mathcal{R}_{\mathrm{pole}}(\theta) \;=\; \frac{1}{LrK} \sum_{\ell,a,k} \big(\max(|p_{a,k}^{(\ell)}| - \rho_{\mathrm{safe}}, 0)\big)^2 \tag{4.6}$$

with $\rho_{\mathrm{safe}} = 0.95$, and a suffix term

$$\widehat{R}_{\mathrm{suf}}(\theta) = \frac{1}{N}\sum_{j=1}^{N} \frac{\|(\mathcal{G}_\theta \boldsymbol{u}^{(j)} - \boldsymbol{y}^{(j)})_{T_{\mathrm{ctx}}:T-1}\|_{\mathrm{F}}}{\|\boldsymbol{y}^{(j)}_{T_{\mathrm{ctx}}:T-1}\|_{\mathrm{F}}}, \qquad T_{\mathrm{ctx}} = \lfloor T/4 \rfloor,$$

which emphasizes errors after an initial context. ZNO's default training objective is

$$\mathcal{L}_{\mathrm{ZNO}}(\theta) = \widehat{R}(\theta) + \lambda_{\mathrm{pole}}\mathcal{R}_{\mathrm{pole}}(\theta) + \lambda_{\mathrm{suf}}\widehat{R}_{\mathrm{suf}}(\theta), \tag{4.7}$$

with $\lambda_{\mathrm{pole}} = 10^{-3}$ and $\lambda_{\mathrm{suf}} = 10^{-2}$, where $\widehat{R}$ is the task loss (3.1). The suffix term is computed from the same full forward pass and is not an autoregressive evaluation. The baselines are trained on $\widehat{R}$. In the 8.5k comparison of Section 5.1 and its timing study in Appendix N, ZNO is also trained on $\widehat{R}$; all other ZNO experiments use the default objective (4.7), apart from the diagnostic variants of Appendix K that switch the pole penalty off, and Appendix E compares the two objectives.

Recurrent evaluation and cost. For batch size B_{batch} and fixed layer dimensions, one causal pass costs $O(B_{\mathrm{batch}}T(wr + rK + rF + w^2))$ operations (low-rank projections, pole updates and residue readout, FIR branch, dense skip), and streaming inference keeps $O(B_{\mathrm{batch}}rK)$ state values plus an FIR buffer. Like recurrent S4D and S5, the cost is linear in T. We implement the recurrence as a fused CUDA/Triton [28] scan that computes the same recurrence as the PyTorch reference up to roundoff. Scan pseudocode, backward-memory trade-offs, a cost comparison with FNO, S4D, and S5, and the relation to an LNO reparameterization are given in Appendices B and C.

5 Experiments

We evaluate ZNO on a synthetic discrete-time identification benchmark (Section 5.1), a pole-radius difficulty sweep (Section 5.2), longer evaluation horizons (Section 5.3), and public nonlinear system-identification benchmarks (Section 5.4). Appendix D summarizes the protocol of every study.

Setup. Each synthetic task uses $N_{\mathrm{train}}/N_{\mathrm{val}}/N_{\mathrm{test}} = 1024/256/256$ i.i.d. trajectories of length $T = 2048$. All models train for 600 epochs with batch size 32, and each run is evaluated on the test set once, with its best-validation checkpoint. We report the relative ℓ_2 error (3.1) as mean $\pm$ standard deviation over seeds and count parameters as real degrees of freedom, with complex scalars counted twice (Appendix B). Models are implemented in PyTorch [23], except S5, which runs in JAX/Flax; hardware and compute are listed in Appendix D.

Code availability. GitHub repository: `https://github.com/hstral/ZNO-NeurIPS2026`.

5.1 Synthetic discrete-time identification benchmark

Tasks. All three tasks are driven by a randomized two-tone forcing with additive noise and a slow amplitude envelope (Appendix D). **Resonant ARMA:** a second-order autoregressive plant with a conjugate pole pair $p = \rho\,\mathrm{e}^{\mathrm{i}\phi}$, $\rho \sim \mathcal{U}(0.9, 0.995)$, $\phi \sim \mathcal{U}(0.05\pi, 0.45\pi)$, and a 4-tap moving-average numerator; the input has 7 channels (forcing and six system parameters). **Sixth-order IIR cascade:** three biquad sections with independent resonant poles of radius in $(0.88, 0.995)$; 16 input channels (forcing and five parameters per section). **Nonlinear NARX:** the scalar recurrence $y_n = a_1 y_{n-1} + a_2 y_{n-2} + \mathrm{gain}\left(\log(1 + \mathrm{e}^{x_n + 0.5x_{n-1}}) - \log 2\right) + 0.08\tanh(y_{n-1}x_n)$, with $a_1 \sim \mathcal{U}(0.2, 0.6)$, $a_2 \sim \mathcal{U}(-0.3, 0.1)$, and $\mathrm{gain} \sim \mathcal{U}(0.15, 0.45)$ sampled per trajectory; the models receive only the forcing x_n, not a_1, a_2, or the gain. The first two tasks provide the system parameters as inputs, whereas NARX is a pure input-output identification problem [17].

Models and training. We compare ZNO with FNO [16], a stabilized LNO [4], S4D [8], and S5 [27], each with about 8.5k real parameters (Table 6). FNO has width 4, 126 Fourier modes, and four layers. The stabilized LNO is a single layer of width 14 with 8 modes, whose continuous poles are constrained to $\mathrm{Re}(\mu) < 0$ (Appendix H). S4D has width 20, state size 12, and four layers. S5 has width 20, base state size 40, and four layers, with a causal regression adapter (Appendix D). ZNO has width 20, $r = 8$, $K = 40$, $F = 4$, and four layers. FNO, LNO, and ZNO are trained with Adam at learning rate 2×10^{-3} and step decay, S4D with Adam at 5×10^{-4} and gradient clipping at 1.0, and S5 with Adam/AdamW at 10^{-3}, a one-epoch warm-up, and cosine decay (Table 4). In this comparison all models minimize the task loss (3.1), and each is trained with seeds 0–4.

Task	FNO	Stabilized LNO	S4D	S5	ZNO (ours)
Resonant ARMA	$0.528_{\pm 0.004}$ (8 525)	$0.448_{\pm 0.008}$ (8 643)	$0.386_{\pm 0.011}$ (8 577)	$0.225_{\pm 0.017}$ (8 741)	$\mathbf{0.198}_{\pm 0.013}$ (8 657)
IIR cascade	$0.841_{\pm 0.031}$ (8 597)	$0.835_{\pm 0.014}$ (8 769)	$0.704_{\pm 0.011}$ (8 757)	$\mathbf{0.655}_{\pm 0.020}$ (8 921)	$0.737_{\pm 0.132}$ (8 837)
Nonlinear NARX	$0.34908_{\pm 0.00066}$ (8 477)	$0.31325_{\pm 0.00038}$ (8 559)	$0.31090_{\pm 0.00017}$ (8 457)	$0.31206_{\pm 0.00033}$ (8 621)	$\mathbf{0.30944}_{\pm 0.00006}$ (8 537)

Table 1: Synthetic benchmark with about 8.5k real parameters per model, all trained on the task loss (3.1): relative ℓ_2 test error, mean $\pm$ standard deviation over seeds 0–4, with real parameter counts in parentheses; lowest mean in bold. Stabilized LNO: single-layer LNO with continuous poles constrained to $\mathrm{Re}(\mu) < 0$.

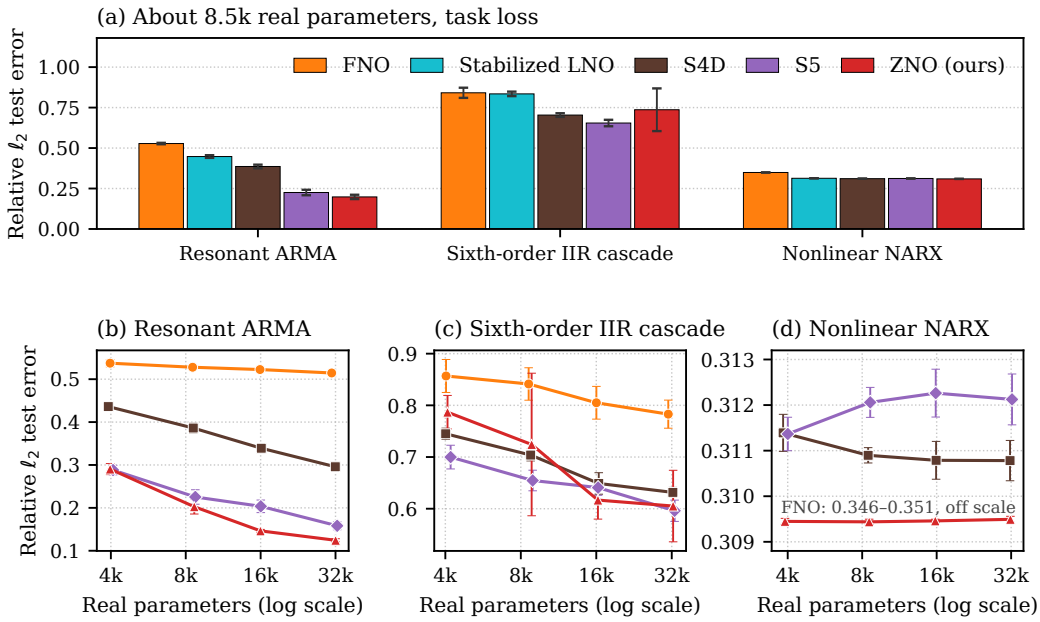


Figure 3: Synthetic benchmark. (a) Table 1. (b–d) Capacity study at about 4k, 8.5k, 16k, and 32k real parameters, plotted against the real parameter count (ZNO with its default objective). Error bars: standard deviation over seeds 0–4. In (d) the vertical axis is zoomed and FNO lies off scale.

Results. FNO and the stabilized LNO are less accurate than S4D, S5, and ZNO on all three tasks, and the ranking among the recurrent models depends on the task (Table 1, Figure 3a). ZNO has the lowest error on resonant ARMA ($0.198_{\pm 0.013}$, against $0.225_{\pm 0.017}$ for S5) and on the noise-limited NARX task, where the recurrent models approach a floor near $\mathrm{rel}\,\ell_2 \approx 0.31$ and ZNO ($0.30944_{\pm 0.00006}$) leads S4D ($0.31090_{\pm 0.00017}$) and S5 ($0.31206_{\pm 0.00033}$). On the sixth-order IIR cascade, S5 has the lowest error ($0.655_{\pm 0.020}$), followed by S4D ($0.704_{\pm 0.011}$) and ZNO ($0.737_{\pm 0.132}$). ZNO's spread on this task comes from one seed that stays at the error of a zero prediction (1.000); the other four seeds reach 0.660–0.680 (Appendix E). Training ZNO with its default objective (4.7) changes its mean errors by at most 0.013 and leaves every ranking unchanged (Appendix E).

Sensitivity to the parameter budget. We also train FNO, S4D, S5, and ZNO at about 4k, 16k, and 32k real parameters. Each model's width and resolution variable (Fourier modes, state size, or rank and pole count) are set at depth four from parameter counts alone before training, placing every model within 5% of each budget (Appendix F); ZNO uses its default objective. On resonant ARMA, S5 and ZNO are within one standard deviation at 4k ($0.288_{\pm 0.011}$ and $0.290_{\pm 0.013}$), and ZNO has the lowest error at 8.5k, 16k, and 32k (Figure 3b–d, Table 13). On the IIR cascade, S5 has the lowest error at 4k, 8.5k, and 32k, and ZNO at 16k. On NARX, ZNO has the lowest error at every budget, by at most

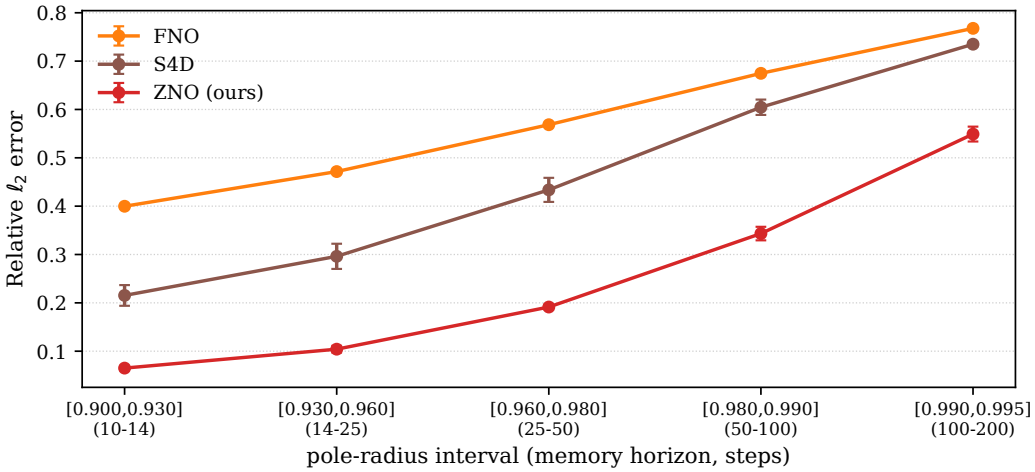


Figure 4: Near-unit-circle difficulty sweep for FNO, S4D, and ZNO. The pole-radius bins control the memory horizon $\tau \approx -1/\log \rho$, from roughly 10 steps to 100–200 steps. Error bars: standard deviation over five seeds. ZNO has the lowest mean error in every bin; the gap to S4D narrows in the most challenging long-memory regime but does not close.

0.002. FNO is the least accurate model at every budget. The task-dependent pattern thus persists across budgets from about 4k to 32k real parameters, although no model leads at every budget.

5.2 Near-unit-circle difficulty scaling

This study isolates how pole radius, and therefore memory horizon, affects FNO, S4D, and ZNO. The resonant-ARMA task draws radii from $(0.9, 0.995)$ and thus mixes short and long memory. We construct five narrower bins, $\rho \in \{[0.90, 0.93], [0.93, 0.96], [0.96, 0.98], [0.98, 0.99], [0.99, 0.995]\}$, whose dominant time constants $\tau \approx -1/\log \rho$ span $\{10\text{–}14, 14\text{–}25, 25\text{–}50, 50\text{–}100, 100\text{–}200\}$ steps. For each bin we train each model with seeds 0–4 and the resonant-ARMA configuration of Table 7 (FNO width 8 with 256 modes, S4D state size 24, ZNO $r = 12$, $K = 64$, $F = 0$).

Results. Figure 4 shows three findings. First, every model degrades as the memory horizon grows. Second, ZNO has the lowest mean error in every bin, from the shortest-memory bin (ZNO $0.065_{\pm 0.004}$, S4D $0.215_{\pm 0.021}$, FNO $0.400_{\pm 0.002}$) to the longest (ZNO $0.549_{\pm 0.015}$, S4D $0.735_{\pm 0.005}$, FNO $0.768_{\pm 0.001}$). Third, the ZNO-to-S4D error ratio grows from 0.30 to 0.75: the advantage is largest for short and intermediate memory and narrows for very long memory, which remains hard for every model.

5.3 Long-horizon evaluation

We evaluate FNO, S4D, and ZNO checkpoints trained at $T = 2048$ with the fixed configurations of Table 7 on test sequences of length 4096 and 8192 (seeds 0–4; Appendix I). ZNO and S4D, whose causal recurrences have length-independent parameters, are essentially length-invariant: from $T = 2048$ to 8192 their errors change by factors of $\{0.90, 1.06, 1.05\}$ and $\{0.96, 1.04, 1.07\}$ on {ARMA, IIR, NARX}, and ZNO keeps the lowest absolute error at every length. FNO's error grows by $3.6\times$, $2.7\times$, and $1.9\times$, consistent with its frequency grid being tied to the training length. The same pattern holds in every difficulty bin (Appendix I).

5.4 Public nonlinear system-identification benchmarks

We evaluate five public benchmarks loaded through the `nonlinear-benchmarks` package [1]: Silverbox [29] (a Duffing-like electronic circuit), Wiener–Hammerstein [25] (a static nonlinearity between two linear dynamic blocks), CED [30] (coupled electric drives), Cascaded Tanks [26] (fluid levels with overflow), and EMPS [13] (an electro-mechanical positioning system). All use the

Benchmark	Classical	FNO	S4D	S5	ZNO (ours)
Silverbox	0.041 (10 terms)	$0.977_{\pm 0.047}$ (99 769)	$0.067_{\pm 0.021}$ (10 377)	$0.031_{\pm 0.003}$ (12 605)	$\mathbf{0.021}_{\pm 0.001}$ (14 569)
Wiener–Hammerstein	0.1865 (8 terms)	$0.0341_{\pm 0.0016}$ (99 769)	$0.0329_{\pm 0.0073}$ (11 657)	$0.0104_{\pm 0.0015}$ (9 949)	$\mathbf{0.0031}_{\pm 0.0003}$ (15 609)
CED	0.221 (10 terms)	$0.198_{\pm 0.024}$ (4 315)	$0.202_{\pm 0.041}$ (7 817)	$0.365_{\pm 0.077}$ (7 293)	$\mathbf{0.162}_{\pm 0.021}$ (7 513)
Cascaded Tanks	0.489 (5 terms)	$0.389_{\pm 0.051}$ (5 467)	$\mathbf{0.331}_{\pm 0.012}$ (8 457)	$0.548_{\pm 0.051}$ (4 637)	$0.385_{\pm 0.071}$ (9 769)
EMPS	$\mathbf{0.080}$ (6 terms)	$0.340_{\pm 0.029}$ (67 001)	$0.342_{\pm 0.026}$ (8 457)	$0.357_{\pm 0.021}$ (4 637)	$1.025_{\pm 0.199}$ (8 025)

Table 2: Public nonlinear system-identification benchmarks: relative ℓ_2 test error. Neural models: mean $\pm$ standard deviation over seeds 3–7 after validation-based selection on seeds 0–2, with real parameter counts in parentheses. Classical: validation-selected NARMAX model (single deterministic fit), with its number of terms, i.e., selected regressors or basis functions, which is not a real parameter count. Lowest mean in bold.

released train/test split, a sequential $80/20$ train/validation split of the training stream, train-stream normalization, and the benchmark-provided initialization prefix, which is excluded from the error (Appendix J). Each neural model's configuration, including S5's state size, is selected from its own grid by mean validation error over seeds 0–2 and reported on seeds 3–7, with capacities and training recipes chosen per model (Table 20). The classical column is the NARMAX estimator family [15] with the lowest validation error.

Results. ZNO has the lowest mean error on Silverbox, Wiener–Hammerstein, and CED (Table 2), with S5 second on the first two. S4D performs best on Cascaded Tanks, and a low-order classical model has the lowest error on EMPS, substantially below all neural models. On EMPS, ZNO also shows a large validation/test gap (0.242 validation vs. $1.025_{\pm 0.199}$ test; Table 22).

Why ZNO falls short on EMPS and Cascaded Tanks. Diagnostic studies on seeds 0–4 (Appendix K) point to two contributing factors. On EMPS, the learned poles do not crowd the 0.999 ceiling (per-seed maximum radius 0.959–0.992), yet relaxing the radius bound lowers ZNO's test error from 0.987 to 0.585–0.806 and shrinks the validation/test gap, while the final poles remain inside the unit disk. The optimization geometry of the bounded map thus appears to matter, alongside a strong shift between the training and test excitations; the best variant remains far from the classical model. On Cascaded Tanks, ZNO learns much faster poles than S4D (median radius 0.590 vs. 0.994), and initializing its radii in $[0.90, 0.995]$ lowers its error from 0.370 to 0.306 on the same seeds, which points to sensitivity to memory initialization.

6 Conclusion

ZNO builds each operator layer from a discrete-time rational transfer matrix: latent pole-residue filters with poles in the unit disk, an optional FIR branch, and low-rank channel mixing, evaluated exactly by a causal recurrence. With about 8.5k parameters per model, it is more accurate than FNO and a stabilized LNO on all three synthetic tasks; against the closest MIMO state-space model, S5, its advantage is clear on resonant ARMA, narrow on NARX, and absent on the IIR cascade, a pattern that largely persists from 4k to 32k parameters. As the poles approach the unit circle, ZNO has lower error than FNO and S4D in every memory regime, with the largest margins for short and intermediate memory. At four times the training length, its error stays within about 10% of its value at the training length and remains the lowest on every task and every radius bin, whereas FNO's grows by up to $3.6\times$ on the three tasks and up to $8.1\times$ in the radius bins. On public benchmarks ZNO leads on three of five systems. ZNO is therefore a useful model for stable, lightly damped discrete-time dynamics, alongside state-space and classical identification methods.

Limitations. Our experiments compare complete layers: one-factor ablations show that rank and pole count matter most (Appendix L), but no factorial study separates the contributions of the low-rank factorization, the pole-residue form, the FIR branch, and the recurrence. The rational recurrence can

be sensitive to optimization, as one failed IIR seed shows (Appendix E). On EMPS and Cascaded Tanks ZNO is not competitive, and our explanations of these cases are exploratory. Finally, all experiments concern one-dimensional sequences.

Future work. Promising directions are smoother pole maps without a hard ceiling, initializations that cover long memories, component-level controlled studies, and spatio-temporal operator learning, where FNO and LNO are established but ZNO has not yet been evaluated.

Acknowledgments and Disclosure of Funding

X. Zhu and J. Yin were partially supported by Shanghai Rising-Star Program under Grant No. 24QA2700600 and National Natural Science Foundation of China under Grant No. 12571422. The authors declare no relevant competing interests.

References

[1] Gerben I. Beintema. nonlinear-benchmarks: The official dataloader of nonlinearbenchmark.org, 2025. Python package, version 1.0.1.

[2] Gerben I. Beintema, Maarten Schoukens, and Roland Tóth. Deep subspace encoders for nonlinear system identification. *Automatica*, 156:111210, 2023.

[3] Stephen A. Billings. *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.

[4] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.

[5] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, 1995.

[6] Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 933–941. PMLR, 2017.

[7] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487, 2020.

[8] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 35971–35983, 2022.

[9] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations (ICLR)*, 2022.

[10] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems*, volume 35, pages 22982–22994, 2022.

[11] Bjorn Gustavsen and Adam Semlyen. Rational approximation of frequency domain responses by vector fitting. *IEEE Transactions on Power Delivery*, 14(3):1052–1061, 1999.

[12] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). arXiv preprint arXiv:1606.08415, 2016.

[13] Alexandre Janot, Maxime Gautier, and Mathieu Brunot. Data set and reference models of EMPS. In *2019 Workshop on Nonlinear System Identification Benchmarks*, Eindhoven, The Netherlands, 2019. April 10–12, 2019.

[14] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Animashree Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

[15] Wilson Rocha Lacerda, Luan Pascoal Costa da Andrade, Samuel Carlos Pessoa Oliveira, and Samir Angelo Milani Martins. SysIdentPy: A Python package for system identification using NARMAX models. *Journal of Open Source Software*, 5(54):2384, 2020.

[16] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Animashree Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[17] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[18] Lennart Ljung, Carl Andersson, Koen Tiels, and Thomas B. Schön. Deep learning and system identification. *IFAC-PapersOnLine*, 53(2):1175–1181, 2020.

[19] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[20] Alan V. Oppenheim, Ronald W. Schafer, and John R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1999.

[21] Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26670–26698. PMLR, 2023.

[22] Rom Parnichkun, Stefano Massaroli, Alessandro Moro, Jimmy T. H. Smith, Ramin Hasani, Mathias Lechner, Qi An, Christopher Ré, Hajime Asama, Stefano Ermon, Taiji Suzuki, Michael Poli, and Atsushi Yamashita. State-free inference of state-space models: The Transfer Function approach. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39834–39860. PMLR, 2024.

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037, 2019.

[24] Johan Schoukens and Lennart Ljung. Nonlinear system identification: A user-oriented road map. *IEEE Control Systems Magazine*, 39(6):28–99, 2019.

[25] Johan Schoukens, Johan Suykens, and Lennart Ljung. Wiener–Hammerstein benchmark. In *15th IFAC Symposium on System Identification (SYSID 2009)*, St. Malo, France, July 2009. July 6–8, 2009.

[26] Maarten Schoukens and Jean-Philippe Noël. Three benchmarks addressing open challenges in nonlinear system identification. *IFAC-PapersOnLine*, 50(1):446–451, 2017.

[27] Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: An intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[29] Torbjörn Wigren and Johan Schoukens. Three free data sets for development and benchmarking in nonlinear system identification. In *2013 European Control Conference (ECC)*, pages 2933–2938. IEEE, 2013.

[30] Torbjörn Wigren and Maarten Schoukens. Coupled electric drives data set and reference models. Technical Report 2017-024, Department of Information Technology, Uppsala University, 2017.

A State-space form of the ZNO layer

Let $\nu = \lceil K/2 \rceil$ be the number of stored modes per latent channel (one per conjugate pair, plus one real pole when K is odd), and collect the states $s_{a,n}^{(\ell,k)}$ of all latent channels $a = 1, \dots, r$ in $\boldsymbol{x}_n \in \mathbb{C}^{r\nu}$. The rational branch (4.2)–(4.3) is then the diagonal state-space system (3.4) with

$$\bar{\boldsymbol{\Lambda}} = \operatorname{diag}\big(p_{a,k}^{(\ell)}\big)_{a,k}, \qquad \bar{\boldsymbol{B}} = (\boldsymbol{I}_r \otimes \mathbf{1}_\nu)\, \boldsymbol{B}^{(\ell)}, \qquad \tilde{\boldsymbol{C}} = \boldsymbol{A}^{(\ell)}\ \operatorname{blkdiag}\big(\tilde{\boldsymbol{c}}_1^{(\ell)\top}, \dots, \tilde{\boldsymbol{c}}_r^{(\ell)\top}\big), \quad \text{(A.1)}$$

where $\tilde{\boldsymbol{c}}_a^{(\ell)} \in \mathbb{C}^\nu$ holds the residues of latent channel a (doubled for stored conjugate pairs), and $\operatorname{blkdiag}$ places the listed row vectors in block-diagonal positions: $\tilde{\boldsymbol{c}}_a^{(\ell)\top} \in \mathbb{C}^{1\times\nu}$ occupies row a and columns $(a-1)\nu + 1, \dots, a\nu$ of an $r \times r\nu$ matrix whose other entries are zero, so that $\tilde{\boldsymbol{C}} \in \mathbb{C}^{w\times r\nu}$. Each latent input drives only the ν modes of its own bank, and every mode is read out through its latent channel and then mixed by $\boldsymbol{A}^{(\ell)}$. The FIR branch adds the finite-memory path $\boldsymbol{A}^{(\ell)} \sum_{j=0}^{F} \operatorname{diag}(\boldsymbol{g}_j^{(\ell)}) \boldsymbol{B}^{(\ell)} \boldsymbol{h}_{n-j}$ for $F > 0$, and the skip adds $\boldsymbol{W}^{(\ell)} \boldsymbol{h}_n$, so the feedthrough of the full layer is $\boldsymbol{W}^{(\ell)} + \mathbf{1}_{F>0} \boldsymbol{A}^{(\ell)} \operatorname{diag}(\boldsymbol{g}_0^{(\ell)}) \boldsymbol{B}^{(\ell)}$. Because $\bar{\boldsymbol{B}}$ and $\tilde{\boldsymbol{C}}$ factor through $\boldsymbol{B}^{(\ell)} \in \mathbb{R}^{r\times w}$ and $\boldsymbol{A}^{(\ell)} \in \mathbb{R}^{w\times r}$, the transfer matrix (4.4) of the rational branch has rank at most r at every frequency.

An S5 layer uses the same diagonal form with dense complex $\bar{\boldsymbol{B}} \in \mathbb{C}^{P/2\times H}$ and $\tilde{\boldsymbol{C}} \in \mathbb{C}^{H\times P/2}$, so each of its modes reads from and writes to all channels. The two layers also differ in the pole parameterization (discrete-time radius and phase through (4.5) versus continuous-time eigenvalues discretized by zero-order hold with learned steps), the optional FIR branch, the static path (a dense pointwise skip versus a diagonal feedthrough followed by gating and layer normalization), and the initialization (radii drawn uniformly in $[0.30, 0.95]$ and phases in $(0, \pi)$ versus eigenvalues from HiPPO). HiPPO (High-order Polynomial Projection Operators) [7] specifies continuous-time state matrices whose state holds the coefficients of a polynomial approximation of the input history; S5 uses this construction as its state-space initialization, taking its eigenvalues from a diagonalized HiPPO matrix. S4D corresponds to block-diagonal $\bar{\boldsymbol{B}}$ and $\tilde{\boldsymbol{C}}$ with one SISO block per channel, followed by pointwise channel mixing.

B Parameter counting and computational cost

Counting convention. Parameter counts are real degrees of freedom: each complex scalar counts twice, and lift, readout, normalization, and gating parameters are included. S4D, S5, and ZNO store real and imaginary parts as separate real tensors; FNO and LNO store complex spectral weights, each entry of which contributes two real parameters. For example, the 8.5k FNO on resonant ARMA (width 4, 126 modes) has four complex $4 \times 4 \times 64$ spectral weights, which contribute $8\,192$ of its $8\,525$ real parameters. The budget labels 4k, 8.5k, 16k, and 32k refer to these real counts.

ZNO layer. For the even pole counts used by the reported configurations, layer ℓ has rw (in-projection) + rw (out-projection) + $w^2 + w$ (skip) + Kr (complex-residue real and imaginary parameters) + Kr (pole radii and phases) + $\mathbf{1}_{F>0}\, r(F+1)$ (FIR) real parameters; for example, $1\,420$ for $w = 20$, $r = 8$, $K = 40$, $F = 4$. A dense $w \times w$ rational transfer matrix with K poles per entry would need $O(w^2 K)$ dynamic coefficients, whereas ZNO routes the dynamics through r latent filters with $O(rK)$ dynamic and $O(rw)$ mixing parameters.

Cost comparison. Table 3 compares per-layer parameters, sequence work, and streaming state at fixed layer dimensions. *Work per sequence* is the order of arithmetic operations for one layer processing one example of length T. *Streaming state* is the information a layer retains from one time step to the next during step-by-step inference; "cplx." denotes complex-valued entries, and ZNO's rF

real entries hold the FIR input history. FNO has no recurrent state: its FFT acts on the whole input window, so a new output step requires the whole window again. All recurrent layers are linear in T; they differ in how the dynamics are shared across channels, in their constants, and in their parallel depth. The entries are orders of growth, not measured run times.

Layer	Real parameters per layer	Work per sequence	Streaming state	Causal
ZNO (w, r, K, F)	$2rw + w^2 + w + 2rK + r(F{+}1)$	$O\big(T(wr + rK + rF + w^2)\big)$	$r\lceil K/2\rceil$ cplx. $+ rF$	yes
S5 (H, P)	$2PH + O(H^2 + P)$	$O(TPH)$	$P/2$ cplx.	yes
S4D (H, N)	$HN + O(H^2)$	$O(THN + TH \log T + TH^2)$	$HN/2$ cplx.	yes
FNO (w, m_r)	$2w^2 m_r + O(w^2)$	$O(wT \log T + w^2 m_r + Tw^2)$	whole window	no

Table 3: Per-layer cost for one example at fixed dimensions. ZNO is evaluated by a sequential scan, S5 by a parallel scan of depth $O(\log T)$, S4D by kernel generation and FFT convolution (or recurrently), and FNO by FFTs over the whole window. P is the S5 base state size before conjugate reduction, N the S4D state size per channel, and $m_r = \lfloor m/2 \rfloor + 1$ the number of retained FNO frequencies for m modes. Multiply work and state by B_{batch} for a batch.

C Fused recurrent kernel and relation to LNO

Scan pseudocode. Algorithm 1 gives the forward scan for one ZNO layer (complex conjugate pairs, no FIR). The recurrence is sequential in n but parallel over batch size B_{batch}, rank r, and pole count K. A PyTorch-level loop of length T therefore incurs many kernel launches and dominates per-epoch wall time for $T \geq 512$. The fused implementation admits two exact backward modes: `save_history` stores every intermediate state at $O(B_{\text{batch}} TrK)$ memory and performs one additional sequential pass; `recompute` stores only the final state, reruns the forward scan, and performs two sequential passes at $O(B_{\text{batch}} rK)$ memory. We verified the fused path against a PyTorch reference implementation, obtaining a maximum absolute deviation of order 10^{-8} on both forward and gradient. On a single NVIDIA A30 (24 GB) the fused scan is 6–8$\times$ faster than our reference Python scan at $T = 2048$, $r = 12$, $K = 64$, $B_{\text{batch}} = 32$. Because it computes the same recurrence, the fused kernel changes run time but not the model or its predictions beyond numerical roundoff.

Algorithm 1 Forward scan for one ZNO layer (complex conjugate pairs, no FIR).

Require: hidden sequence $\boldsymbol{h} \in \mathbb{R}^{B_{\text{batch}} \times T \times w}$, poles $\{\boldsymbol{p}_k\}_{k=1}^{K/2} \subset \mathbb{C}^r$, residues $\{\boldsymbol{c}_k\}_{k=1}^{K/2} \subset \mathbb{C}^r$, in-proj $\boldsymbol{B}$, out-proj $\boldsymbol{A}$, skip $\boldsymbol{W}$

$\boldsymbol{b} \leftarrow \boldsymbol{h}\,\boldsymbol{B}^\top$ ▷ $B_{\text{batch}} \times T \times r$, real

$\boldsymbol{s}_0 \leftarrow \boldsymbol{0} \in \mathbb{C}^{B_{\text{batch}} \times r \times (K/2)}$

for $n = 0, \ldots, T-1$ **do**

  for each pole pair $k = 1, \ldots, K/2$ **do**

    $\boldsymbol{s}_n^{(k)} \leftarrow \boldsymbol{p}_k \odot \boldsymbol{s}_{n-1}^{(k)} + \boldsymbol{b}_n$

  end for

  $\boldsymbol{q}_n \leftarrow 2\,\mathrm{Re}\Big(\sum_k \boldsymbol{c}_k \odot \boldsymbol{s}_n^{(k)}\Big)$

  $\boldsymbol{o}_n \leftarrow \boldsymbol{A}\,\boldsymbol{q}_n + \boldsymbol{W}\,\boldsymbol{h}_n$

end for

return $\boldsymbol{o} \in \mathbb{R}^{B_{\text{batch}} \times T \times w}$

Timing protocol. All wall-clock numbers follow one protocol: (i) the Triton kernel backend; (ii) the fused save-history backward mode; (iii) a Triton extension compiled before the timed region starts; (iv) the first CUDA forward/backward step run as an untimed warm-up; and (v) training time measured as the wall clock of the epoch loop only, excluding data loading, model construction, checkpointing, and test evaluation.

Relation to an LNO reparameterization. The change of variables $\mu \leftrightarrow (\log p)/\Delta t$ does not turn LNO into ZNO; three differences remain.

1. **Direct z-plane parameterization.** For a continuous-time pole μ, the discrete pole is $p = \exp(\mu\Delta t)$, so an s-plane implementation must choose a sampling interval Δt before evaluating the discrete recurrence. For native discrete-time data this interval is an interface choice rather than a physical modeling variable. ZNO parameterizes p directly in the unit disk via (4.5), enforcing stability as $|p| < 1$ and making the learned poles directly readable in the discrete-time domain.
2. **Low-rank MIMO rational layer.** The reference LNO layer applies a dense per-channel multiplication in the Laplace domain. The ZNO dynamic branch (4.2)–(4.3) is a low-rank factorization of a MIMO rational transfer matrix with $r \ll w$ latent channels, and the full residual layer adds a dense pointwise skip.
3. **Causal recurrent implementation.** The reference LNO evaluates pole-residue terms using FFTs and dense frequency-by-pole contractions over the time grid and retained modes. In contrast, ZNO evaluates fixed-rank, fixed-pole filters by a causal recurrent scan whose cost is linear in T for fixed r and K.

D Training protocols and configurations

Training recipes. Table 4 lists the training recipe of each model, which is fixed across studies except where noted. Every run uses batch size 32, trains for 600 epochs on the synthetic and public tasks, and keeps the checkpoint with the lowest validation error; learning-rate schedules depend only on the epoch. The 1D LNO benchmark (Appendix M) uses the epoch budgets of the original LNO release. Table 5 summarizes the studies, the ZNO objective in each, and their seeds.

Model	Optimizer	Learning rate	Weight decay	Schedule	Clip
FNO	Adam	2×10^{-3}	10^{-4}	StepLR(100, 0.5)	–
S4D	Adam, parameter groups	5×10^{-4}	10^{-4}	StepLR(150, 0.5)	1.0
ZNO	Adam	2×10^{-3}	10^{-4}	StepLR(150, 0.5)	–
S5	Adam / AdamW	10^{-3} (public 2×10^{-3})	0 / 0.05	warm-up, cosine	–
Stabilized LNO	Adam	2×10^{-3}	10^{-4}	StepLR(150, 0.5)	–

Table 4: Training recipes. StepLR(s, γ) multiplies the learning rate by γ every s epochs. S5 trains its SSM parameters with Adam without weight decay and all other parameters with AdamW, using a one-epoch warm-up and cosine decay to 10^{-6}. All models use the task loss (3.1); ZNO's default objective adds the pole-safety and suffix terms (4.7). Two S4D tuning candidates use learning rate 3×10^{-4} (Appendix G).

Study	Models	Configuration	Seeds	Where
8.5k comparison	FNO, LNO, S4D, S5, ZNO	about 8.5k each (Tab. 6)	0–4	Tab. 1
Objective controls	ZNO; FNO, S4D (suffix)	8.5k ZNO and S4D; FNO $w = 4$, 256 modes	0–4	App. E
Capacity study	FNO, S4D, S5, ZNO	4k–32k (Tab. 12)	0–4	Fig. 3, App. F
Tuning study	FNO, S4D, ZNO	44 candidates, validation selection	0–2 / 0–4	App. G
LNO reference capacity	stabilized LNO	about 20k	0–4	App. H
Difficulty, long horizon	FNO, S4D, ZNO	fixed configs. (Tab. 7)	0–4	Secs. 5.2–5.3
Timing	FNO, LNO, S4D, S5, ZNO	8.5k configs. (Tab. 6)	0–4	App. N
Pole maps and traces	FNO, S4D, ZNO	fixed configs. (Tab. 7)	0–4	App. O
Public benchmarks	4 neural + classical	per-model validation selection	0–2 / 3–7	Tab. 2
Failure diagnostics	ZNO	radius map, initialization, pole penalty varied	0–4	App. K
Ablation	ZNO	one knob at a time from an anchor	0–4	App. L
1D ODE benchmark	FNO, LNO, ZNO	Matched-budget configurations (Appendix M)	0–2	Fig. 7

Table 5: Studies in this paper. "0–2 / 3–7" denotes selection / reporting seeds. ZNO minimizes its default objective (4.7) in every study except the 8.5k comparison and its timing study, where all models use the task loss (3.1); the objective controls also train ZNO on the task loss and without its auxiliary terms, and the failure diagnostics vary the pole penalty. The other models minimize the task loss, except that the suffix-loss control adds the suffix term to FNO and S4D.

Synthetic tasks. All three tasks share the forcing generator. For each trajectory and time grid $t_n = n/T$, $n = 0, \dots, T-1$, the forcing is

$$x_n = \big[0.8\sin(2\pi f_1 t_n + \varphi_1) + 0.4\sin(2\pi f_2 t_n + \varphi_2) + 0.2\,\varepsilon_n\big]\big[0.75 + 0.25\sin(2\pi f_e t_n + \varphi_e)\big],$$

with $f_1 \sim \mathcal{U}(1,5)$, $f_2 \sim \mathcal{U}(5,12)$, and $f_e \sim \mathcal{U}(0.25,1.5)$ cycles per sequence, phases $\varphi_1, \varphi_2, \varphi_e \sim \mathcal{U}(0, 2\pi)$, and i.i.d. standard normal ε_n. All plants start from zero initial conditions, and the output is a single channel.

- **Resonant ARMA.** $y_n = a_1 y_{n-1} + a_2 y_{n-2} + \sum_{j=0}^{3} \beta_j x_{n-j}$ with $a_1 = 2\rho\cos\phi$ and $a_2 = -\rho^2$, i.e., poles $\rho e^{\pm i\phi}$, where $\rho \sim \mathcal{U}(0.9, 0.995)$ and $\phi \sim \mathcal{U}(0.05\pi, 0.45\pi)$. The moving-average coefficients are drawn as $\beta_j \sim \mathcal{U}(-0.18, 0.18)$, after which β_0 is shifted by 0.35 away from zero (in the direction of its sign). The input has 7 channels: x_n and the constant parameters $(\rho, \phi/\pi, \beta_0, \ldots, \beta_3)$.
- **Sixth-order IIR cascade.** Three biquad sections in series, each $v_n = a_1 v_{n-1} + a_2 v_{n-2} + b_0 e_n + b_1 e_{n-1} + b_2 e_{n-2}$ with $a_1 = 2\rho\cos\phi$, $a_2 = -\rho^2$, $\rho \sim \mathcal{U}(0.88, 0.995)$, $\phi \sim \mathcal{U}(0.04\pi, 0.35\pi)$, $b_0 \sim \mathcal{U}(0.18, 0.85)$, $b_1 \sim \mathcal{U}(-0.35, 0.35)$, and $b_2 \sim \mathcal{U}(-0.2, 0.2)$. The first section is driven by $e_n = x_n$, each later section by the previous section's output, and the target is the output of the third section. The input has 16 channels: x_n and the constant parameters $(\rho, \phi/\pi, b_0, b_1, b_2)$ of each section.
- **Nonlinear NARX.** The recurrence of Section 5.1 with $a_1 \sim \mathcal{U}(0.2, 0.6)$, $a_2 \sim \mathcal{U}(-0.3, 0.1)$, and $\text{gain} \sim \mathcal{U}(0.15, 0.45)$ per trajectory. These parameters generate the target but are not supplied as inputs: the input is the single forcing channel x_n.

On each task, every model receives the same input channels. Each task has 1024 training, 256 validation, and 256 test trajectories of length $T = 2048$, generated from independent random streams. The difficulty sweep of Section 5.2 uses the resonant-ARMA generator with the radius range restricted to each bin and the same split sizes.

Implementations. FNO uses the `neuraloperator` package.[2] S4D uses a wrapper around the S4D kernel parameterization of the `state-spaces/s4` repository,[3] with its parameter-group metadata for the optimizer. The stabilized LNO is our variant of the reference LNO layer [4][4] (Appendix H). The classical baselines use SysIdentPy 0.8.0. S5 uses the upstream JAX/Flax implementation of [27][5] between a linear input encoder and a pointwise linear readout. At the 8.5k budget the model has width $H = 20$, four layers, and base state size $P = 40$ (before conjugate reduction, so 20 complex modes are stored per layer), with eigenvalues initialized from a single diagonalized HiPPO block, causal (unidirectional) operation, conjugate symmetry, zero-order-hold discretization with initial steps in $[10^{-3}, 10^{-1}]$, and eigenvalue clipping. Each layer applies S5's gated activation, a GELU [12] followed by a sigmoid gate computed from the same features, as in gated linear units [6], with layer pre-normalization and without batch normalization or dropout. Its topology is set by parameter count alone. On the synthetic tasks S5 uses the upstream default learning rate of 10^{-3}, without gradient clipping; the public runs use 2×10^{-3} (Appendix J). The JAX pipeline reads the same arrays, splits, sample order, and channel layout as the PyTorch pipeline, computes the same metric, and is causal, and its parameter count covers the full parameter tree.

Configurations at the 8.5k budget. Table 6 lists the topologies of Table 1. Each model uses one topology for all three tasks; parameter counts differ only through the input channels.

Model	Topology	ARMA	IIR	NARX
FNO	$w = 4$, 126 modes, $L = 4$	8 525	8 597	8 477
Stabilized LNO	$w = 14$, 8 modes, $L = 1$	8 643	8 769	8 559
S4D	$w = 20$, state size $N = 12$, $L = 4$	8 577	8 757	8 457
S5	$H = 20$, base state size $P = 40$, $L = 4$	8 741	8 921	8 621
ZNO	$w = 20$, $r = 8$, $K = 40$, $F = 4$, $L = 4$	8 657	8 837	8 537

Table 6: Configurations at the 8.5k budget (Table 1) with real parameter counts; L is the number of layers.

[2] `https://github.com/neuraloperator/neuraloperator`
[3] `https://github.com/state-spaces/s4`
[4] `https://github.com/qianyingcao/Laplace-Neural-Operator`
[5] `https://github.com/lindermanlab/S5`

Fixed configurations. The difficulty sweep, the long-horizon evaluation, and the pole maps and traces use one fixed configuration per task and model (Table 7). All have four layers, S4D and ZNO have width 20, and ZNO uses $\rho_{\max} = 0.999$ and $\rho_{\text{safe}} = 0.95$. Training follows Table 4, with ZNO on its default objective, for seeds 0–4. The difficulty sweep uses the resonant-ARMA configurations in every radius bin.

Task	FNO	S4D	ZNO
Resonant ARMA	$w = 8$, $m = 256$ (67 097)	$N = 24$ (10 497)	$r = 12$, $K = 64$, $F = 0$ (12 721)
IIR cascade	$w = 8$, $m = 256$ (67 241)	$N = 16$ (9 397)	$r = 12$, $K = 80$, $F = 4$ (14 677)
Nonlinear NARX	$w = 4$, $m = 256$ (16 797)	$N = 16$ (9 097)	$r = 8$, $K = 16$, $F = 4$ (7 001)

Table 7: Fixed configurations of the difficulty sweep (Section 5.2), the long-horizon evaluation (Section 5.3), and the pole maps and traces (Appendix O), with real parameter counts in parentheses. FNO: width w and modes m; S4D: state size N; ZNO: rank, poles per latent filter, and FIR order.

Seeds and uncertainty. The synthetic-task studies use seeds 0–4 (the tuning study selects on seeds 0–2), the public benchmarks seeds 0–2 for selection and 3–7 for reporting, and the 1D LNO benchmark seeds 0–2 with the original $200/50/130$ split of [4]. All $\pm$ values and error bars are population standard deviations over seeds; with five seeds, we treat differences within about one standard deviation as ties.

Compute. Table 8 lists the compute of every reported study by run set: the number of runs and their cumulative epoch-loop time in worker-hours, measured with the timing protocol of Appendix C. Each neural-network run uses a single NVIDIA A30 GPU, with PyTorch/CUDA for FNO, LNO, S4D, and ZNO and JAX/Flax for S5. The classical baselines run on CPU. The table includes the configuration-selection runs.

Study	Run set	Worker	Runs	Hours
8.5k comparison (Tab. 1)	FNO, LNO, S4D, S5, ZNO	GPU	75	7.2
Objective controls (App. E)	ZNO, default objective	GPU	15	2.3
	suffix-loss variants	GPU	60	7.7
	IIR learning-rate and clipping variants	GPU	20	14.1
Capacity study (App. F)	4k, 16k, 32k	GPU	180	18.2
Tuning study (App. G)	selection, seeds 0–2	GPU	132	15.9
	selected configurations, seeds 0–4	GPU	45	5.5
Stabilized LNO (App. H)	about 20k	GPU	15	0.9
Fixed configurations (Tab. 7)	FNO, S4D, ZNO, seeds 0–4	GPU	45	5.6
Difficulty sweep (Sec. 5.2)	five radius bins	GPU	75	8.1
Long-horizon evaluation (Sec. 5.3)	synthetic tasks	GPU	90	9.9
	difficulty bins	GPU	150	16.2
Public benchmarks (Sec. 5.4)	selection, FNO/S4D/ZNO grids	GPU	339	9.1
	selection, S5 state sizes	GPU	75	2.1
	FNO/S4D/ZNO selected, seeds 0–4	GPU	75	2.2
	FNO/S4D/ZNO selected, seeds 5–7	GPU	45	1.3
	S5 selected, seeds 3–7	GPU	25	0.6
Diagnostics (App. K)	EMPS and Cascaded Tanks variants	GPU	35	0.5
ZNO ablation (App. L)	one knob at a time	GPU	100	39.8
1D ODE benchmark (App. M)	matched-budget configurations	GPU	54	2.3
Total on GPUs		GPU	1650	169.5
Public SysIdentPy baselines (Sec. 5.4)	classical fits	CPU	20	0.7

Table 8: Compute by study and run set: number of runs and cumulative epoch-loop worker-hours. Long-horizon runs retrain each configuration with its seed before testing at the longer lengths.

E Objective and optimization controls

The controls in this appendix use the synthetic tasks, the protocol of Table 1, seeds 0–4, and the 8.5k ZNO and S4D configurations of Table 6; the FNO of the suffix-loss control has its own topology, given below.

Default objective. Table 1 trains every model on the task loss. Table 9 compares these ZNO runs with ZNO trained on its default objective (4.7) at the same configuration and seeds; the other models

always use the task loss. ZNO's mean errors change by $+0.0045$ (ARMA), -0.0124 (IIR), and less than 10^{-5} (NARX), and the ranking of Table 1 is the same with either objective.

Task	ZNO, task loss (Table 1)	ZNO, default objective	Difference
Resonant ARMA	$0.198_{\pm 0.013}$	$0.203_{\pm 0.017}$	$+0.00451$
IIR cascade	$0.737_{\pm 0.132}$	$0.724_{\pm 0.138}$	-0.01236
Nonlinear NARX	$0.30944_{\pm 0.00006}$	$0.30944_{\pm 0.00006}$	$< 10^{-5}$

Table 9: ZNO at the 8.5k budget trained on the task loss and on its default objective (pole-safety and suffix terms): relative ℓ_2 test error over seeds 0–4.

Suffix loss for the baselines. Giving FNO and S4D the same suffix loss changes their errors by less than 0.001 (Table 10), and removing ZNO's suffix term, or both auxiliary terms, leaves its error within one standard deviation. This control uses FNO with width 4 and 256 modes ($16\,845$/$16\,917$/$16\,797$ real parameters on ARMA/IIR/NARX) and S4D and ZNO with the configurations of Table 6.

Task	FNO + suffix	S4D + suffix	ZNO base	ZNO no suffix	ZNO no suffix/pole
Resonant ARMA	$0.515_{\pm 0.005}$	$0.387_{\pm 0.011}$	$0.203_{\pm 0.017}$	$0.199_{\pm 0.013}$	$0.198_{\pm 0.013}$
IIR cascade	$0.851_{\pm 0.012}$	$0.704_{\pm 0.011}$	$0.727_{\pm 0.136}$	$0.736_{\pm 0.132}$	$0.737_{\pm 0.132}$
Nonlinear NARX	$0.34523_{\pm 0.00122}$	$0.31098_{\pm 0.00009}$	$0.30944_{\pm 0.00006}$	$0.30944_{\pm 0.00006}$	$0.30944_{\pm 0.00006}$

Table 10: Suffix-loss control (seeds 0–4): relative ℓ_2 test error. Adding the suffix loss to FNO and S4D, or removing ZNO's auxiliary terms, leaves the ordering of the three models unchanged. "ZNO base" is an independent run of the default objective.

Optimization failures. No run in the 8.5k comparison, the capacity study, or the tuning study produced a non-finite loss. One run did not train: ZNO with seed 4 on the IIR cascade at the 8.5k budget, whose best validation and test errors remain at 1.000, the error of a zero prediction, with either objective; the next worst run in these studies reaches a validation error of 0.864. This is the only such run (1 of 15 ZNO runs in Table 1, 1 of 60 in the capacity study, 0 of 15 in the tuning study); FNO, the stabilized LNO, S4D, and S5 have none. Table 11 trains this configuration with four combinations of learning rate and gradient clipping (default objective, seeds 0–4). With learning rate 10^{-3} and clipping at 0.5 all five seeds train, giving $0.706_{\pm 0.023}$ against $0.724_{\pm 0.138}$ with the recipe of Table 4, which all other experiments use.

Model	Variant	IIR cascade (6th) rel. ℓ_2 (5 seeds)
ZNO (Table 9)	recipe of Table 4	$0.724_{\pm 0.138}$
ZNO	lr1e-3, clip 0.25	$0.744_{\pm 0.033}$
ZNO	lr1e-3, clip 0.5	$\mathbf{0.706_{\pm 0.023}}$
ZNO	lr8e-4, clip 0.25, safe 0.90	$0.742_{\pm 0.042}$
ZNO	lr8e-4, clip 0.5	$0.725_{\pm 0.032}$

Table 11: ZNO on the IIR cascade at the 8.5k budget with modified learning rate and gradient clipping (default objective, seeds 0–4).

F Capacity sensitivity

The capacity study trains FNO, S4D, S5, and ZNO at about 4k, 8.5k, 16k, and 32k real parameters with seeds 0–4. Data, splits, epochs, batch size, checkpointing, and metric match Table 1, and training follows Table 4, with ZNO on its default objective. All models have four layers, and each model uses one topology per budget for the three tasks (Table 12). The topologies are fixed from parameter counts before training. For S4D, S5, and ZNO, the width follows a monotone path from the 8.5k topology, and each model's resolution variable (S4D: state size; S5: base state size; ZNO: rank and poles per latent filter, with $F = 4$) is adjusted until all three task-specific counts lie within 5% of the target. FNO uses widths 3, 4, 6, and 6 with 102, 126, 104, and 208 modes at 4k, 8.5k, 16k, and 32k, likewise set from parameter counts. Every model is thus within 5% of each budget in real parameters.

The 8.5k entries are those of Table 1 for FNO, S4D, and S5 and those of Table 9 for ZNO. Table 13 gives the results; no run produced a non-finite loss.

Budget	FNO	S4D	S5	ZNO
4k	w=3, m=102 3 936–4 026	w=12, N=6 3 833–4 013	H=13, P=28 4 004–4 199	w=12, r=5, K=22, F=4 3 901–4 081
8.5k	w=4, m=126 8 477–8 597	w=20, N=12 8 457–8 757	H=20, P=40 8 621–8 921	w=20, r=8, K=40, F=4 8 537–8 837
16k	w=6, m=104 15 835–16 015	w=30, N=16 15 917–16 367	H=27, P=56 15 862–16 267	w=30, r=12, K=50, F=4 15 797–16 247
32k	w=6, m=208 30 811–30 991	w=44, N=26 31 673–32 333	H=39, P=80 32 266–32 851	w=42, r=17, K=94, F=4 31 777–32 407

Table 12: Capacity-study topologies (four layers) and the range of real parameter counts over the three tasks. FNO: width w and modes m; S4D: width and state size N; S5: width H and base state size P; ZNO: width, rank, poles per latent filter, and FIR order.

Budget	Task	FNO	S4D	S5	ZNO (ours)
4k	ARMA	$0.537_{\pm 0.009}$ (3 972)	$0.436_{\pm 0.004}$ (3 905)	$\mathbf{0.288}_{\pm 0.011}$ (4 082)	$0.290_{\pm 0.013}$ (3 973)
	IIR	$0.857_{\pm 0.032}$ (4 026)	$0.745_{\pm 0.011}$ (4 013)	$\mathbf{0.700}_{\pm 0.023}$ (4 199)	$0.787_{\pm 0.032}$ (4 081)
	NARX	$0.34962_{\pm 0.00074}$ (3 936)	$0.31139_{\pm 0.00041}$ (3 833)	$0.31137_{\pm 0.00037}$ (4 004)	$\mathbf{0.30945}_{\pm 0.00007}$ (3 901)
8.5k	ARMA	$0.528_{\pm 0.004}$ (8 525)	$0.386_{\pm 0.011}$ (8 577)	$0.225_{\pm 0.017}$ (8 741)	$\mathbf{0.203}_{\pm 0.017}$ (8 657)
	IIR	$0.841_{\pm 0.031}$ (8 597)	$0.704_{\pm 0.011}$ (8 757)	$\mathbf{0.655}_{\pm 0.020}$ (8 921)	$0.724_{\pm 0.138}$ (8 837)
	NARX	$0.34908_{\pm 0.00066}$ (8 477)	$0.31090_{\pm 0.00017}$ (8 457)	$0.31206_{\pm 0.00033}$ (8 621)	$\mathbf{0.30944}_{\pm 0.00006}$ (8 537)
16k	ARMA	$0.522_{\pm 0.001}$ (15 907)	$0.339_{\pm 0.006}$ (16 097)	$0.204_{\pm 0.015}$ (16 024)	$\mathbf{0.146}_{\pm 0.003}$ (15 977)
	IIR	$0.805_{\pm 0.032}$ (16 015)	$0.649_{\pm 0.021}$ (16 367)	$0.641_{\pm 0.015}$ (16 267)	$\mathbf{0.617}_{\pm 0.037}$ (16 247)
	NARX	$0.35071_{\pm 0.00074}$ (15 835)	$0.31079_{\pm 0.00041}$ (15 917)	$0.31226_{\pm 0.00052}$ (15 862)	$\mathbf{0.30946}_{\pm 0.00004}$ (15 797)
32k	ARMA	$0.514_{\pm 0.003}$ (30 883)	$0.296_{\pm 0.010}$ (31 937)	$0.158_{\pm 0.009}$ (32 500)	$\mathbf{0.124}_{\pm 0.004}$ (32 029)
	IIR	$0.783_{\pm 0.027}$ (30 991)	$0.632_{\pm 0.010}$ (32 333)	$\mathbf{0.596}_{\pm 0.021}$ (32 851)	$0.605_{\pm 0.069}$ (32 407)
	NARX	$0.34619_{\pm 0.00076}$ (30 811)	$0.31078_{\pm 0.00044}$ (31 673)	$0.31212_{\pm 0.00056}$ (32 266)	$\mathbf{0.30949}_{\pm 0.00007}$ (31 777)

Table 13: Capacity study (ZNO with its default objective): relative ℓ_2 test error over seeds 0–4, real parameter counts in parentheses, lowest mean per row in bold.

G Tuning study

Grid and selection. The tuning study selects one configuration per task for FNO, S4D, and ZNO from a predefined grid of 44 candidates (Table 15): four FNO candidates per task (width 2 or 4 with 128 modes; width 4 or 8 with 256 modes), six S4D candidates per task (state size 12, 16, 24, 32 at learning rate 5×10^{-4}, and 24, 32 at 3×10^{-4}), and five, five, and four ZNO candidates on ARMA, IIR, and NARX. All ZNO candidates use $w = 20$, $L = 4$, $\rho_{\max} = 0.999$, learning rate 2×10^{-3}, and the default objective; one IIR candidate uses $\rho_{\text{safe}} = 0.90$, and one ARMA candidate omits the FIR branch ($F = 0$). Each candidate is trained with seeds 0–2 for 600 epochs with a fixed schedule and best-validation checkpointing, without access to the test split, and scored by its mean best-validation error. Each selected configuration is then trained with seeds 0–4 and evaluated on the test set.

Results. ZNO has the lowest mean test error of the three models on all three tasks (Table 14), for example $0.171_{\pm 0.006}$ on resonant ARMA against $0.358_{\pm 0.022}$ for S4D.

H Stabilized LNO

The reference LNO layer [4] evaluates a pole-residue transform whose continuous poles μ are trained without constraint, so a pole can enter the right half-plane, $\mathrm{Re}(\mu) > 0$, where its mode grows along the sequence. With the unmodified layer at the topology of the larger setting below (width 12, 32 modes), our runs on resonant ARMA and NARX (seeds 0–2) ended with relative ℓ_2 test errors between 2.6×10^5 and 2.9×10^{11}. The stabilized LNO of Table 1 is our variant of the open-source LNO implementation: it parameterizes the real part of each continuous pole as $\mathrm{Re}(\mu) = -\,\mathrm{softplus}(\tilde{a})$ with an unconstrained raw decay $\tilde{a}$, which keeps every pole in the open left

Task	Model	Selected configuration	Real params	Val. (seeds 0–2)	Test (seeds 0–4)
Resonant ARMA	FNO	w=8, m=256	67 097	0.502	$0.505_{\pm 0.001}$
	S4D	N=24	10 497	0.385	$0.358_{\pm 0.022}$
	ZNO	w=20, r=12, K=64, F=4	12 961	0.193	$\mathbf{0.171}_{\pm 0.006}$
IIR cascade	FNO	w=8, m=256	67 241	0.726	$0.756_{\pm 0.011}$
	S4D	N=16	9 397	0.660	$0.703_{\pm 0.007}$
	ZNO	w=20, r=12, K=80, F=4	14 677	0.572	$\mathbf{0.618}_{\pm 0.025}$
Nonlinear NARX	FNO	w=4, m=256	16 797	0.37003	$0.34480_{\pm 0.00076}$
	S4D	N=32	11 657	0.33943	$0.31075_{\pm 0.00030}$
	ZNO	w=20, r=12, K=16, F=4	8 233	0.33756	$\mathbf{0.30941}_{\pm 0.00003}$

Table 14: Tuning study (FNO, S4D, ZNO): configurations selected by mean best-validation error over seeds 0–2, test error over seeds 0–4. All models have depth four; S4D has width 20. Lowest test mean per task in bold.

Task	Model	Candidate	Real params	Mean best val. (seeds 0–2)
Resonant ARMA	FNO	w=2, m=128	2 199	0.548
Resonant ARMA	FNO	w=4, m=128	8 653	0.525
Resonant ARMA	FNO	w=4, m=256	16 845	0.516
Resonant ARMA	FNO	w=8, m=256	67 097	**0.502**
Resonant ARMA	S4D	N=12	8 577	0.391
Resonant ARMA	S4D	N=16	9 217	0.401
Resonant ARMA	S4D	N=24	10 497	**0.385**
Resonant ARMA	S4D	N=24, lr 3×10^{-4}	10 497	0.417
Resonant ARMA	S4D	N=32	11 777	0.388
Resonant ARMA	S4D	N=32, lr 3×10^{-4}	11 777	0.416
Resonant ARMA	ZNO	r=8, K=32, F=4	8 145	0.239
Resonant ARMA	ZNO	r=12, K=32, F=4	9 889	0.208
Resonant ARMA	ZNO	r=8, K=64, F=4	10 193	0.205
Resonant ARMA	ZNO	r=12, K=64, F=0	12 721	0.193
Resonant ARMA	ZNO	r=12, K=64, F=4	12 961	**0.193**
IIR cascade	FNO	w=2, m=128	2 235	0.842
IIR cascade	FNO	w=4, m=128	8 725	0.825
IIR cascade	FNO	w=4, m=256	16 917	0.794
IIR cascade	FNO	w=8, m=256	67 241	**0.726**
IIR cascade	S4D	N=12	8 757	0.692
IIR cascade	S4D	N=16	9 397	**0.660**
IIR cascade	S4D	N=24	10 677	0.686
IIR cascade	S4D	N=24, lr 3×10^{-4}	10 677	0.722
IIR cascade	S4D	N=32	11 957	0.691
IIR cascade	S4D	N=32, lr 3×10^{-4}	11 957	0.727
IIR cascade	ZNO	r=8, K=48, F=4	9 349	0.644
IIR cascade	ZNO	r=8, K=80, F=4	11 397	0.642
IIR cascade	ZNO	r=12, K=48, F=4	11 605	0.635
IIR cascade	ZNO	r=12, K=80, F=4	14 677	**0.572**
IIR cascade	ZNO	r=12, K=80, F=4, ρ_{safe}=0.90	14 677	0.597
Nonlinear NARX	FNO	w=2, m=128	2 175	0.37273
Nonlinear NARX	FNO	w=4, m=128	8 605	0.37388
Nonlinear NARX	FNO	w=4, m=256	16 797	**0.37003**
Nonlinear NARX	FNO	w=8, m=256	67 001	0.37037
Nonlinear NARX	S4D	N=12	8 457	0.33982
Nonlinear NARX	S4D	N=16	9 097	0.33948
Nonlinear NARX	S4D	N=24	10 377	0.33947
Nonlinear NARX	S4D	N=24, lr 3×10^{-4}	10 377	0.33963
Nonlinear NARX	S4D	N=32	11 657	**0.33943**
Nonlinear NARX	S4D	N=32, lr 3×10^{-4}	11 657	0.33966
Nonlinear NARX	ZNO	r=8, K=16, F=4	7 001	0.33757
Nonlinear NARX	ZNO	r=8, K=32, F=4	8 025	0.33760
Nonlinear NARX	ZNO	r=12, K=16, F=4	8 233	**0.33756**
Nonlinear NARX	ZNO	r=12, K=32, F=4	9 769	0.33761

Table 15: Candidate grid of the tuning study with real parameter counts and mean best-validation error over seeds 0–2 (selected candidate in bold). FNO entries list width w and modes m; S4D entries list state size N (width 20); ZNO entries list rank, poles per latent filter, and FIR order (width 20).

half-plane, so the linear pole-residue modes decay on long sequences. No normalization or gradient clipping is added. The model is a single LNO layer of width 14 with 8 modes (8 643/8 769/8 559 real parameters on ARMA/IIR/NARX), trained with the task loss, the recipe of Table 4, and seeds 0–4. A second setting with width 12 and 32 modes (20 477/20 585/20 405 real parameters) serves as a larger reference. All runs train to finite errors, and the larger setting gives nearly the same errors as the 8.5k setting (Table 16).

Task	w=14, m=8 (Table 1)	w=12, m=32
Resonant ARMA	$0.448_{\pm 0.008}$ (8 643)	$0.449_{\pm 0.014}$ (20 477)
IIR cascade	$0.835_{\pm 0.014}$ (8 769)	$0.836_{\pm 0.006}$ (20 585)
Nonlinear NARX	$0.31325_{\pm 0.00038}$ (8 559)	$0.31302_{\pm 0.00013}$ (20 405)

Table 16: Stabilized LNO at two capacities (single layer, task loss): relative ℓ_2 test error over seeds 0–4, real parameter counts in parentheses.

I Long-horizon extrapolation: full numbers

We evaluate the FNO, S4D, and ZNO checkpoints with the fixed configurations of Table 7, trained at $T = 2048$ with seeds 0–4, on test sequences of length 4096 and 8192 drawn from the same task generators. Each long-horizon run retrains its configuration at $T = 2048$ with the same seed, which reproduces the best validation error of the corresponding $T = 2048$ run, and tests the best-validation checkpoint at the longer length with the same metric. Figure 5 shows the errors at the three lengths, and Table 17 lists them with standard deviations.

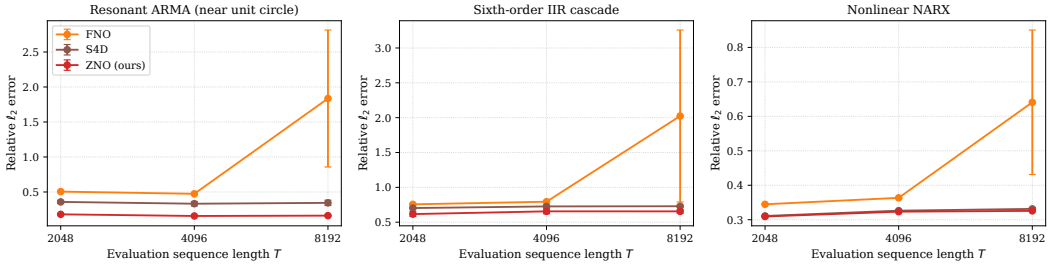


Figure 5: Long-horizon evaluation at $T \in \{2048, 4096, 8192\}$ (five seeds). ZNO and S4D are essentially length-invariant because their layers are causal recurrences with length-independent parameters, and ZNO retains the lowest error on every task at every length. FNO's error grows sharply at $T = 8192$, consistent with its frequency grid being tied to the training length.

Extrapolation across the difficulty sweep. We evaluate the checkpoints of Section 5.2 at $T \in \{4096, 8192\}$ in all five pole-radius bins. Both ZNO and S4D remain within $[0.91, 1.01]$ of their $T = 2048$ error (5-seed means), while FNO's error grows on every bin at $T = 8192$ (bin 0: $8.1\times$; bin 4: $2.6\times$). ZNO has the lowest mean error of the three models on every bin at every length.

J Public nonlinear system-identification benchmarks: protocol and full results

Protocol. All datasets are loaded through the `nonlinear-benchmarks` Python package [1] using the released train/test partitions and converted to scalar input/output streams. A sequential $80/20$ split of the training stream forms the train/validation partitions and is shared by all models and seeds. Inputs and outputs are normalized by the train-stream mean and standard deviation and windowed into sequences with a scored segment and a short initialization prefix (Table 18); the prefix is excluded from the reported error, consistent with the benchmark convention.

Task	Model	$T = 2048$	$T = 4096$	$T = 8192$
Resonant ARMA	FNO	$0.505_{\pm 0.001}$	$0.474_{\pm 0.001}$	$1.835_{\pm 0.978}$
	S4D	$0.358_{\pm 0.022}$	$0.332_{\pm 0.032}$	$0.345_{\pm 0.038}$
	ZNO	$\mathbf{0.181_{\pm 0.011}}$	$\mathbf{0.157_{\pm 0.008}}$	$\mathbf{0.162_{\pm 0.008}}$
IIR cascade (6th)	FNO	$0.756_{\pm 0.011}$	$0.794_{\pm 0.006}$	$2.023_{\pm 1.234}$
	S4D	$0.703_{\pm 0.007}$	$0.726_{\pm 0.017}$	$0.729_{\pm 0.014}$
	ZNO	$\mathbf{0.616_{\pm 0.035}}$	$\mathbf{0.655_{\pm 0.035}}$	$\mathbf{0.655_{\pm 0.027}}$
Nonlinear NARX	FNO	$0.345_{\pm 0.001}$	$0.363_{\pm 0.001}$	$0.641_{\pm 0.210}$
	S4D	$0.311_{\pm 0.0004}$	$0.326_{\pm 0.001}$	$0.331_{\pm 0.003}$
	ZNO	$\mathbf{0.309_{\pm 0.0001}}$	$\mathbf{0.323_{\pm 0.0001}}$	$\mathbf{0.326_{\pm 0.0001}}$

Table 17: Long-horizon evaluation (seeds 0–4): relative ℓ_2 test error. From $T = 2048$ to 8192, S4D changes by factors $\{0.96, 1.04, 1.07\}$ and ZNO by $\{0.90, 1.06, 1.05\}$ on {ARMA, IIR, NARX}; FNO grows by $3.6\times$, $2.7\times$, and $1.9\times$.

Benchmark	loaded channels	train / val / test windows	scored T + prefix	sampling period
Silverbox	1/1	391 / 86 / 256	$2048 + 50$	1.64×10^{-3} s
Wiener–Hammerstein	1/1	512 / 128 / 256	$2048 + 50$	1.95×10^{-5} s
Cascaded Tanks	1/1	81 / 4 / 64	$128 + 50$	4.00×10^{0} s
CED	1/1	128 / 16 / 24	$48 + 10$	2.00×10^{-2} s
EMPS	1/1	140 / 23 / 178	$2048 + 20$	1.00×10^{-3} s

Table 18: Public benchmark statistics. Window counts are after windowing each released train/test partition. The scored-T column lists the time steps included in the reported metric plus the initialization prefix supplied to the model but excluded from the metric.

Configuration selection. Each neural model is selected per benchmark from its own grid by mean validation error over seeds 0–2: FNO width $\in \{4, 6, 8\}$ and modes $\in \{24, 32, 128, 256, 384\}$; S4D state size $\in \{8, 12, 16, 24, 32\}$; ZNO pole count $K \in \{16, 24, 32, 48, 64, 80\}$, rank $r \in \{8, 12, 16\}$, and FIR order $F \in \{0, 4, 8\}$; and S5 base state size $\in \{16, 24, 32, 48, 64\}$ at width 20 and depth four, with the smaller size preferred on ties (Table 19). Training follows Table 4, with S5 at learning rate 2×10^{-3} on these benchmarks. The selected configurations (Table 20) are trained with seeds 3–7, which Table 2 reports.

Benchmark	P=16 (4 637)	P=24 (5 965)	P=32 (7 293)	P=48 (9 949)	P=64 (12 605)
Silverbox	0.0214	0.0216	0.0212	0.0209	**0.0193**
Wiener–Hammerstein	0.0100	0.0093	0.0093	**0.0086**	0.0087
CED	0.473	0.471	**0.466**	0.475	0.476
Cascaded Tanks	**0.216**	0.254	0.274	0.243	0.257
EMPS	**0.181**	0.189	0.200	0.200	0.204

Table 19: S5 state-size selection on the public benchmarks: mean best-validation error over seeds 0–2 for each base state size P (width 20, depth four; real parameter counts in parentheses). Selected size in bold.

Classical baselines. We use four NARMAX estimators from SysIdentPy [15]: AOLS polynomial, AOLS Fourier, FROLS polynomial, and FROLS Fourier. Each is swept over a small grid of output/input lags, degree or basis order, and number of terms, and the candidate with the lowest validation error is kept. Table 2 reports, per benchmark, the family with the lowest validation error; Table 21 lists all four.

Validation/test gap. Table 22 lists the validation and test errors of each selected neural model. (i) On Silverbox and Wiener–Hammerstein the gap is small for S4D, S5, and ZNO. (ii) On CED it is negative for all four models: the released test split is easier than the held-out tail of the training stream. (iii) On Cascaded Tanks all gaps are positive and largest for S5. (iv) On EMPS the gap is large for all models and largest for ZNO ($+0.783$).

Benchmark	Model	Selected configuration	Real params
Silverbox	FNO	w=8, m=384	99 769
	S4D	w=20, N=24	10 377
	S5	H=20, P=64	12 605
	ZNO	w=20, r=12, K=80, F=8	14 569
Wiener–Hammerstein	FNO	w=8, m=384	99 769
	S4D	w=20, N=32	11 657
	S5	H=20, P=48	9 949
	ZNO	w=20, r=16, K=64, F=4	15 609
CED	FNO	w=6, m=24	4 315
	S4D	w=20, N=8	7 817
	S5	H=20, P=32	7 293
	ZNO	w=20, r=8, K=24, F=4	7 513
Cascaded Tanks	FNO	w=6, m=32	5 467
	S4D	w=20, N=12	8 457
	S5	H=20, P=16	4 637
	ZNO	w=20, r=12, K=32, F=4	9 769
EMPS	FNO	w=8, m=256	67 001
	S4D	w=20, N=12	8 457
	S5	H=20, P=16	4 637
	ZNO	w=20, r=8, K=32, F=4	8 025

Table 20: Selected configurations behind Table 2 (depth four), with real parameter counts.

Benchmark	Family	Configuration	Terms	Validation	Test
Silverbox	AOLS polynomial	`yl2x2-d2-k3`	3	0.119	0.157
	AOLS Fourier	`yl6x6-n3-k8`	8	0.078	0.104
	FROLS polynomial*	`yl4x4-d3-ni10`	10	**0.037**	**0.041**
	FROLS Fourier	`yl4x4-n3-ni8`	8	0.061	0.064
Wiener–Hammerstein	AOLS polynomial	`yl6x6-d2-k8`	8	0.178	0.187
	AOLS Fourier*	`yl6x6-n3-k8`	8	**0.178**	**0.187**
	FROLS polynomial	`yl4x4-d2-ni10`	10	0.240	0.244
	FROLS Fourier	`yl4x4-n3-ni8`	8	0.272	0.277
CED	AOLS polynomial	`yl6x6-d2-k8`	8	0.624	0.506
	AOLS Fourier	`yl6x6-n3-k8`	8	0.471	0.244
	FROLS polynomial*	`yl4x4-d3-ni10`	10	**0.365**	**0.221**
	FROLS Fourier	`yl4x4-n3-ni8`	8	0.460	0.320
Cascaded Tanks	AOLS polynomial	`yl2x2-d2-k3`	3	0.424	0.493
	AOLS Fourier*	`yl4x4-n5-k5`	5	**0.337**	**0.489**
	FROLS polynomial	`yl3x3-d2-ni6`	6	0.476	0.536
	FROLS Fourier	`yl4x4-n3-ni8`	8	0.396	0.480
EMPS	AOLS polynomial	`yl6x6-d2-k8`	8	0.098	0.105
	AOLS Fourier	`yl6x6-n3-k8`	8	0.098	0.105
	FROLS polynomial	`yl4x4-d3-ni10`	10	0.083	0.090
	FROLS Fourier*	`yl2x2-n3-ni6`	6	**0.043**	**0.080**

Table 21: Classical NARMAX families (one deterministic fit each): selected configuration, number of terms, and relative ℓ_2 error. *Lowest validation error, reported in Table 2.

Benchmark	Model	best-val rel. ℓ_2	test rel. ℓ_2	test − val
Silverbox	FNO	0.884	0.977	+0.093
	S4D	0.049	0.067	+0.018
	S5	0.022	0.031	+0.010
	ZNO	0.016	0.021	+0.005
Wiener–Hammerstein	FNO	0.0343	0.0341	−0.0002
	S4D	0.0327	0.0329	+0.0003
	S5	0.0098	0.0104	+0.0005
	ZNO	0.0030	0.0031	+0.0001
CED	FNO	0.446	0.198	−0.248
	S4D	0.379	0.202	−0.177
	S5	0.457	0.365	−0.092
	ZNO	0.320	0.162	−0.158
Cascaded Tanks	FNO	0.265	0.389	+0.124
	S4D	0.271	0.331	+0.060
	S5	0.308	0.548	+0.240
	ZNO	0.337	0.385	+0.048
EMPS	FNO	0.056	0.340	+0.284
	S4D	0.262	0.342	+0.080
	S5	0.200	0.357	+0.157
	ZNO	0.242	1.025	+0.783

Table 22: Validation and test errors of the selected neural models on the public benchmarks (means over seeds 3–7). Validation: train-stream validation split used for selection; test: released test split.

K Diagnostics for EMPS and Cascaded Tanks

We analyze the two public benchmarks on which ZNO is not competitive, using the protocol of Appendix J, the selected ZNO configuration of each benchmark (Table 20), and seeds 0–4. We pair runs by seed and report the mean error reduction and paired t-test p-values without correction for multiple comparisons.

K.1 EMPS

Learned poles. In the ZNO checkpoints of seeds 0–4, the pooled pole radii have median 0.507 and 95th percentile 0.730; 0.23% of poles exceed 0.95 and none exceeds 0.995. The per-seed maximum radii are 0.959, 0.970, 0.985, 0.992, and 0.968. The poles therefore do not accumulate at the $\rho_{\max} = 0.999$ ceiling.

Radius-map variants. We train ZNO ($w = 20$, $r = 8$, $K = 32$, $F = 4$; $8\,025$ parameters) under four radius maps: the default sigmoid with ceiling 0.999, a sigmoid with ceiling 1.05, and an unbounded softplus radius with and without the pole penalty (Table 23). With the default map these runs reach 0.987 (Table 2 reports 1.025 on seeds 3–7). Relative to the default map, the test error falls on average by 0.18 (ceiling 1.05), 0.40 (unbounded, with penalty), and 0.36 (unbounded, without penalty) over the same five seeds ($p = 0.049$, 0.033, 0.010); the validation/test gap shrinks as well, and both unbounded variants also lower the validation error. The final learned poles stay inside the unit disk in every run (largest radius 0.998), although the unbounded maps do not enforce this during training. Relaxing the bound thus improves accuracy while the learned poles remain stable, which points to the optimization geometry of the bounded map rather than to a need for unstable poles. The best variant (0.585) remains well above the other neural models and the classical model in Table 2.

Excitation shift. We compare the normalized training and test streams of EMPS with four statistics: the root mean square (RMS), the RMS of the first difference $x_n - x_{n-1}$, the fraction of spectral power at normalized frequencies of at least 0.25 cycles per sample, and the spectral centroid (the power-weighted mean frequency). The test input has nearly the same RMS as the training input ($1.08\times$) but $4.6\times$ its first-difference RMS, $17.3\times$ its high-frequency fraction, and a $2.6\times$ higher spectral centroid: the test excitation changes faster and carries more high-frequency content. The

Radius map	Pole penalty	Validation	Test	Test − val.	Max $\lvert p\rvert$
$\rho_{\max}$=0.999 (sigmoid)	on	$0.230_{\pm 0.051}$	$0.987_{\pm 0.168}$	$0.758_{\pm 0.122}$	0.9909
$\rho_{\max}$=1.05 (sigmoid)	on	$0.211_{\pm 0.062}$	$0.806_{\pm 0.172}$	$0.595_{\pm 0.144}$	0.9980
none (softplus)	on	$0.174_{\pm 0.032}$	$0.585_{\pm 0.173}$	$0.410_{\pm 0.184}$	0.9966
none (softplus)	off	$0.179_{\pm 0.050}$	$0.626_{\pm 0.196}$	$0.446_{\pm 0.227}$	0.9965

Table 23: EMPS radius-map variants (seeds 0–4). "Max $|p|$" is the largest final pole radius over all layers, channels, and runs.

output statistics are nearly unchanged (ratios 0.99–1.01), and for Cascaded Tanks the input ratios are 0.87–1.22.

Initialization prefix. A longer initialization prefix does not lower ZNO's error. On the seed 0–4 checkpoints, its test error is 1.041 with the 20-step prefix of the benchmark and 1.143 with a 1000-step prefix, whereas S4D's falls from 0.338 to 0.168, so a start-up transient does not explain ZNO's error on EMPS.

Adaptation to the test excitation. This diagnostic uses test labels and is separate from the benchmark evaluation. Starting from the same five ZNO checkpoints, we fine-tune on the labeled test windows 0–31, at the start of the released test stream (50 epochs, learning rate 10^{-4}), and evaluate on the later, non-overlapping test windows 64–177. Their error falls from 1.025 to 0.764, whereas fine-tuning equally long on the training stream leaves it at 1.025. Adapting to the test excitation thus recovers part of the gap, consistent with the excitation shift above.

K.2 Cascaded Tanks

Learned poles. Pooling all layers, channels, and seeds 0–4 of the selected ZNO and S4D configurations, the learned discrete pole radii have median 0.590 for ZNO (one representative per conjugate pair) and 0.994 for S4D; no ZNO pole exceeds 0.943, whereas the S4D 95th percentile is 0.9994. The corresponding median time constants are 1.9 and 178 steps. These distributions describe learned modes, not identified plant poles.

Initialization. ZNO initializes pole radii uniformly in $[0.30, 0.95]$. We train ZNO ($w = 20$, $r = 12$, $K = 32$, $F = 4$; 9 769 parameters) with radii initialized uniformly in $[0.90, 0.995]$ and no pole penalty, together with two controls that keep the default initialization, one without the penalty and one with $\rho_{\text{safe}} = 0.99$. The two controls give identical predictions (0.370), because no pole reaches the penalty threshold. The high-radius initialization lowers the mean test error over the same five seeds from 0.370 to $0.306_{\pm 0.019}$ and moves the learned radii toward the unit circle (Table 24), so ZNO is sensitive to memory initialization on this benchmark.

Initial radii	Pole penalty	Validation	Test	Test − val.	Max $\lvert p\rvert$
$\mathcal{U}[0.30, 0.95]$	on, ρ_{safe}=0.99	$0.283_{\pm 0.061}$	$0.370_{\pm 0.037}$	$0.088_{\pm 0.088}$	0.9428
$\mathcal{U}[0.30, 0.95]$	off	$0.283_{\pm 0.061}$	$0.370_{\pm 0.037}$	$0.088_{\pm 0.088}$	0.9428
$\mathcal{U}[0.90, 0.995]$	off	$0.250_{\pm 0.034}$	$0.306_{\pm 0.019}$	$0.056_{\pm 0.040}$	0.9948

Table 24: Cascaded Tanks initialization variants (seeds 0–4). "Max $|p|$" is the largest final pole radius over all runs.

L Internal ZNO ablation

We ablate the four internal design knobs: number of poles per latent filter (K), low-rank factor (r), FIR order (F), and safe radius (ρ_{safe}), one at a time, on the resonant ARMA and sixth-order IIR cascade tasks. The anchor configuration ($w = 20$, $L = 4$, $r = 8$, $F = 4$, $\rho_{\text{safe}} = 0.95$; $K = 32$ for ARMA and $K = 48$ for IIR) has 8 145 and 9 349 real parameters. Every setting is trained with the ZNO recipe of Table 4, the default objective, and seeds 0–4. Figure 6 shows the effect of each knob, and Table 25 lists the full results. The rank and the pole count matter most: raising r from 8 to 12 or doubling K lowers the error on both tasks, whereas the FIR order and the safe radius change it little.

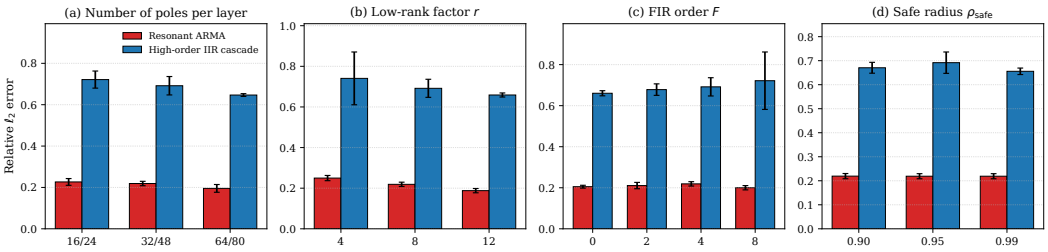


Figure 6: Internal ablation of ZNO on resonant ARMA and sixth-order IIR cascade (5 seeds). (a) Doubling K lowers error on both tasks. (b) Raising r from 4 to 12 yields a clear improvement. The $r = 4$ setting has one failing IIR seed ($0.741_{\pm 0.130}$), and $r = 12$ gives the lowest ARMA error. (c) FIR order is least sensitive: $F = 0$ matches the anchor on ARMA; $F = 8$ increases variance and yields the least stable IIR result among the FIR settings. (d) Safe radius $\rho_{\text{safe}} \in \{0.90, 0.95, 0.99\}$ has almost no effect, as expected for a soft pole-safety penalty.

Configuration	Resonant ARMA	IIR cascade (6th)
anchor ($r = 8, F = 4, K = \{32, 48\}, \rho_{\text{safe}} = 0.95$)	$0.219_{\pm 0.011}$	$0.692_{\pm 0.044}$
rank $r = 4$	$0.250_{\pm 0.013}$	$0.741_{\pm 0.130}$
rank $r = 12$	$\mathbf{0.188_{\pm 0.011}}$	$0.659_{\pm 0.010}$
poles $K = \{16, 24\}$	$0.227_{\pm 0.016}$	$0.721_{\pm 0.041}$
poles $K = \{64, 80\}$	$0.195_{\pm 0.019}$	$\mathbf{0.647_{\pm 0.007}}$
FIR order $F = 0$	$0.205_{\pm 0.007}$	$0.660_{\pm 0.012}$
FIR order $F = 2$	$0.211_{\pm 0.015}$	$0.678_{\pm 0.028}$
FIR order $F = 8$	$0.200_{\pm 0.010}$	$0.722_{\pm 0.140}$
safe radius $\rho_{\text{safe}} = 0.90$	$0.219_{\pm 0.011}$	$0.671_{\pm 0.023}$
safe radius $\rho_{\text{safe}} = 0.99$	$0.219_{\pm 0.011}$	$0.656_{\pm 0.013}$

Table 25: Internal ablation (5 seeds). Relative ℓ_2 test error on the resonant ARMA and sixth-order IIR cascade tasks with one knob perturbed at a time away from the ZNO anchor. Bold indicates the lowest error per column. The low-rank factor and the pole count are the dominant knobs; FIR order and safe radius are comparatively insensitive.

M 1D LNO ODE benchmark

We also evaluate ZNO on the six ODE cases of the LNO benchmark [4]: the pendulum with damping $c \in \{0, 0.5\}$, the Duffing oscillator with $c \in \{0, 0.5\}$, and the Lorenz system with $\rho \in \{5, 10\}$. Each case uses the data released with the LNO code, split into $200/50/130$ training/validation/test trajectories of 2048 samples, and every model is trained with seeds 0–2. The data are sampled trajectories of continuous-time ODE systems, so for this benchmark we follow [4] and report the relative L_2 error of the trajectories ($\text{rel}\, L_2$), computed on the sampling grid as in (3.1).

Matched-budget configurations. Table 26 lists the configurations. FNO has width 2, four layers, and 1025 modes; LNO is a single layer whose width and modes are set per case; ZNO has width 20, four layers, $r = 8$, $F = 4$, and a per-case number of poles K. The models have 6.5k–8.3k parameters, counting each complex weight once. Every run uses the complex-valued Adam implementation of the LNO code with weight decay 10^{-4}, a step decay that halves the learning rate every 100 epochs, and the checkpoint with the lowest validation error. Epochs, batch size, and the learning rate of LNO and ZNO follow the released LNO settings of each case; FNO uses learning rate 2×10^{-3} in every case. FNO and LNO minimize the task loss and ZNO its default objective (4.7).

Results. ZNO has the lowest error on Lorenz $\rho = 5$ ($\text{rel}\, L_2 = 0.007$, against 0.18 for LNO and 0.026 for FNO) and on Pendulum $c = 0.5$, LNO on Lorenz $\rho = 10$, and FNO on the two Duffing cases and on Pendulum $c = 0$ (Figure 7). These continuous-time ODE systems suit LNO and FNO; ZNO trains stably on all six cases, has the lowest error on two, and trails on the Duffing cases.

Case	Epochs	Batch	LR (LNO, ZNO)	LNO: width, modes	ZNO: K
Pendulum, $c = 0$	1200	40	5×10^{-3}	12, 20 (7 745)	20 (7 277)
Pendulum, $c = 0.5$	1200	40	2×10^{-3}	16, 8 (6 721)	8 (6 509)
Duffing, $c = 0$	1000	20	2×10^{-3}	13, 16 (7 537)	16 (7 001)
Duffing, $c = 0.5$	1000	20	2×10^{-3}	13, 16 (7 537)	16 (7 001)
Lorenz, $\rho = 5$	1000	20	5×10^{-3}	13, 16 (7 550)	16 (7 021)
Lorenz, $\rho = 10$	1000	10	2×10^{-3}	6, 84 (7 133)	32 (8 045)

Table 26: Matched-budget configurations of the 1D LNO ODE benchmark, with parameter counts (each complex weight counted once) in parentheses. FNO (width 2, four layers, 1025 modes) has 8 303 parameters on Duffing and 8 307 on the other cases and uses learning rate 2×10^{-3}; ZNO has width 20, four layers, $r = 8$, and $F = 4$.

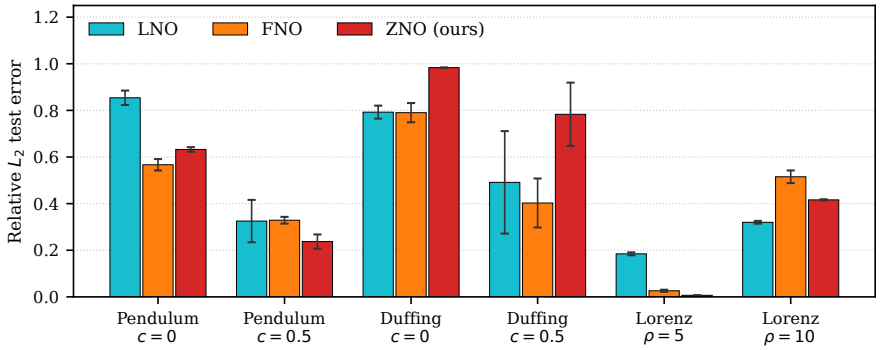


Figure 7: 1D LNO ODE benchmark with the matched-budget configurations of Table 26: relative L_2 test error (mean $\pm$ standard deviation over seeds 0–2).

N Accuracy vs training-time trade-off

Figure 8 compares the wall-clock training time and test error of FNO, the stabilized LNO, S4D, S5, and ZNO using the configurations and task-loss runs of Table 1 (seeds 0–4, one NVIDIA A30 GPU). Each point pairs the mean test error with the mean training time of the same five runs. Training time follows the epoch-loop protocol of Appendix C. ZNO uses the fused Triton scan, FNO, LNO, and S4D their reference PyTorch implementations, and S5 its JAX/Flax implementation, whose recorded time also includes compilation.

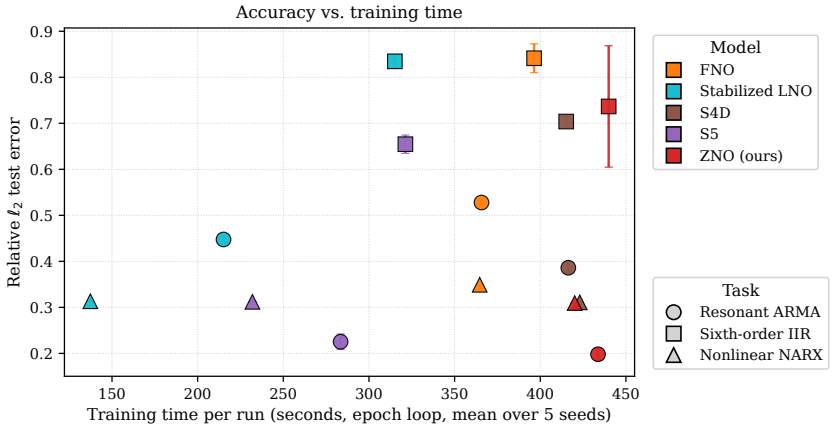


Figure 8: Accuracy vs training time on the three tasks (one NVIDIA A30 GPU; five-seed means, error bars: standard deviation). All five models use the configurations and task-loss runs of Table 1.

O Learned z-plane pole maps and qualitative traces

Because ZNO parameterizes the poles p of the recurrence (4.3) directly, they can be read on the unit disk. Figure 9 shows, for each task, the ZNO checkpoint with the lowest best-validation error among seeds 0–4 of the fixed configuration (Table 7). On resonant ARMA, the outermost learned poles are oscillatory and move outward with depth: the largest radius is 0.82, 0.87, 0.91, and 0.94 from the first to the fourth layer, and the 22 learned poles with $|p| > 0.8$ have phases between 0.05π and 0.43π, inside the phase range of the generator's plant poles, whereas only five of the $1\,536$ stored poles reach the generator's radius range $|p| > 0.9$. On the IIR cascade, the largest learned radius is 0.91, with no consistent trend across layers, and on NARX it is 0.65. In every task, most poles carry small residues and sit near $z \approx \rho_{\max}/2 \approx 0.5$, the pole given by a zero radius logit and a zero phase in (4.5), so the dense cluster on the positive real axis contributes little to the output. No learned pole exceeds the safe radius. LNO and S4D use continuous-time parameters and FNO uses fixed Fourier modes, so this read-out is less direct for them, although their discrete poles can be computed (Appendix K). Figure 10 uses, for each task and model, the checkpoint chosen in the same way and one test trajectory drawn at random.

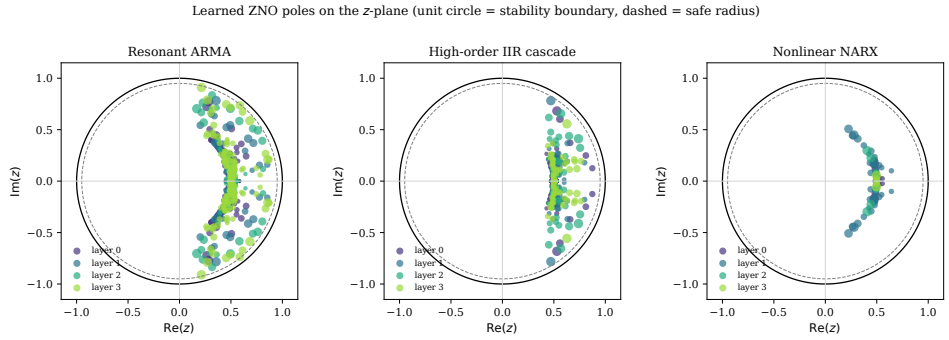


Figure 9: Learned ZNO poles on the z-plane per task. Color: layer depth; marker size: residue magnitude relative to the largest residue in the same layer. Solid/dashed circles: stability boundary and safe radius $\rho_{\text{safe}} = 0.95$. Resonant ARMA has the outermost oscillatory poles, in the deeper layers; the IIR and NARX poles stay well inside the safe radius. The dense cluster near $z \approx 0.5$ consists of low-residue poles.

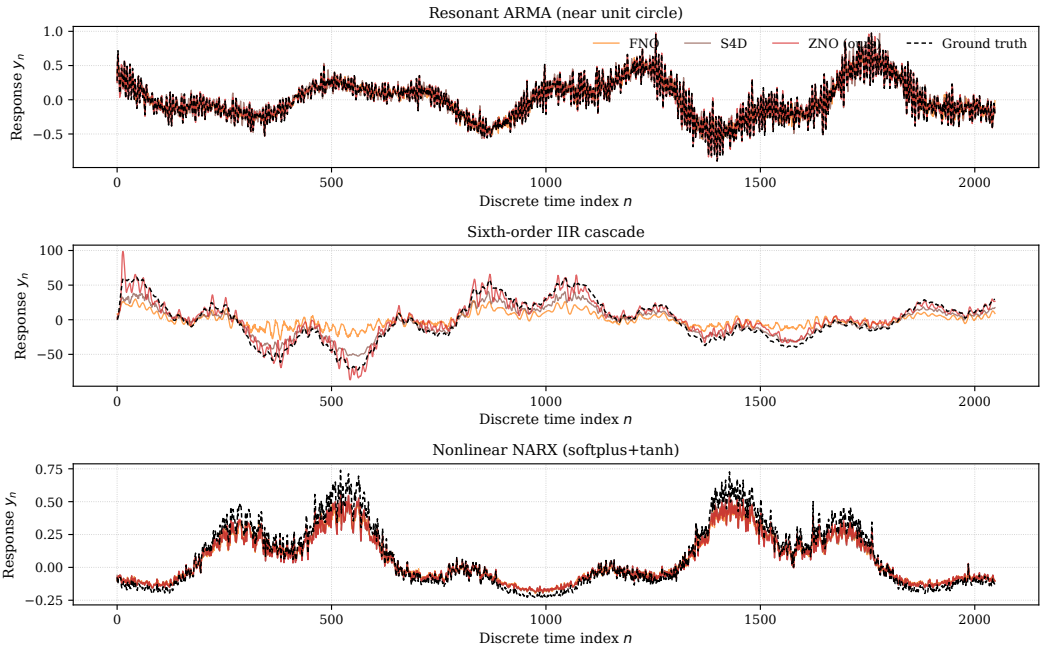


Figure 10: Qualitative comparison on one held-out test trajectory per task. Black dashed: ground truth. Colored: FNO (orange), S4D (brown), ZNO (red). On the resonant ARMA panel, ZNO has the smallest trajectory error and most closely overlays the ground truth. On the sixth-order IIR cascade, FNO substantially underestimates the response amplitude, while S4D and ZNO are closer to the ground truth on this trajectory. On NARX, all three models follow the main low-frequency response, with S4D and ZNO nearly overlapping.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the method, empirical scope, public-benchmark outcomes, and limitations; the claims are matched by the experiments in Sections 5.1–5.4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations paragraph of Section 6 states that no factorial study separates the components of the layer, that the rational recurrence can be sensitive to optimization

(one failed IIR seed), that ZNO is not competitive on EMPS and Cascaded Tanks, and that all experiments concern one-dimensional sequences.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is methodological and empirical; it derives the recurrence from the transfer function (Section 3) and gives the state-space form of the layer (Appendix A), but it states no theorems.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: The architecture (Sections 3–4), data generators and splits, training recipes, configurations of every study, selection procedures, seeds, and metrics are given in Section 5 and Appendices D–M; the public code repository provides scripts and instructions to reproduce the experiments and regenerate the tables and figures.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The supplementary code package includes scripts for generating the synthetic benchmarks, loading the public benchmarks, training ZNO and the neural baselines, evaluating the SysIdentPy baselines, and reproducing the tables and figures from the result assets or reruns.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 and Appendix D specify the optimizers and schedules, parameter budgets and configurations, validation-based selection where it applies, data splits, sequence lengths, seeds, and public-benchmark preprocessing; Appendix M lists the settings of the 1D ODE benchmark.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Neural results report the mean and population standard deviation over five seeds (three on the 1D ODE benchmark), and error bars show the same standard deviation; the diagnostic comparisons of Appendix K use paired t-tests over five seeds with p-values unadjusted for multiple comparisons. The deterministic classical baselines are single fits.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table 8 in Appendix D lists, for every reported study, the number of runs and the cumulative epoch-loop worker-hours on NVIDIA A30 GPUs (24 GB), including configuration-selection runs, and the CPU time of the classical baselines; Appendix N compares accuracy and wall-clock training time. Exploratory runs that do not enter the reported studies required additional compute and are not included in the table.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses synthetic data and public system-identification benchmarks, reports limitations and failures, and does not involve private personal data or human-subject experiments.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The paper is foundational work on neural operators for discrete-time dynamics and does not introduce an application-specific deployed system with direct societal impact.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out

that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The released assets are code, synthetic benchmark generators, and scripts for public system-identification datasets; they do not include high-risk generative models, scraped personal data, or sensitive data.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the public benchmark sources, the `nonlinear-benchmarks` loader, SysIdentPy, FNO, LNO, S4D, and S5; the supplementary code package includes `LICENSE` and `THIRD_PARTY_NOTICES.md` files documenting software dependencies and third-party assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces new code assets for ZNO, the fused recurrent kernel, synthetic benchmark generation, experiment launch scripts, and result processing; these assets are documented in the supplementary package README and reproduction guide.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing or human-subject experiments, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs were not used as an important, original, or non-standard component of the core method. Any use was limited to writing/editing, formatting, routine code assistance, and result organization; all technical choices, experiments, numerical results, and claims were verified by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.