Physics-Informed Echo State Networks for Chaotic Systems: Lorenz and Ma–Chen Case Studies

1 Why study chaotic systems?

A chaotic system is a deterministic dynamical system whose trajectories exhibit sensitive dependence on initial conditions. Formally, the maximal Lyapunov exponent $\lambda_1 > 0$ implies that two states separated by $\|\Delta \mathbf{x}_0\| \ll 10^{-3}$ diverge on average like $e^{\lambda_1 t}$. Analysing chaos is useful because (i) many real-world processes (turbulence, climate, finance) are chaotic; (ii) short-term forecasts still enable control and risk mitigation; (iii) chaotic signatures reveal hidden parameters and stability margins. The main challenge is the predictability horizon $\tau_{\mathrm{L}} \approx 1/\lambda_1$: beyond that, pointwise prediction becomes impossible and models must switch to statistical descriptions.

Lorenz-63 System. The Lorenz system is defined by

$$\dot{x} = \sigma(y - x), \qquad \dot{y} = x(\rho - z) - y, \qquad \dot{z} = xy - \beta z, \tag{1}$$

where σ, ρ and β are positive parameters. In this case we use $(\sigma, \rho, \beta) = (10, 28, 8/3)$. With only three equations, the Lorenz system exhibits rich, nontrivial chaotic behavior that remains analytically and numerically accessible, enjoys broad multidisciplinary appeal across meteorology, physics, and engineering, and serves as the standard benchmark in textbooks and research for testing new chaos-analysis methods

Ma–Chen Financial System. The Ma–Chen (2003) financial system can be written as

$$\dot{x} = z + (y - a)\,x, \tag{2}$$

$$\dot{y} = 1 - b\,y - x^2, \tag{3}$$

$$\dot{z} = -\,x - c\,z, \tag{4}$$

where x, y and z represent the interest rate, investment demand and price index, respectively, and $a,\ b,\ c > 0$ are real constants We chose a= 3, b = 0.1, c = 1, which yields sustained chaotic behavior. We chose the Ma–Chen system because it is a well-known three-dimensional model in financial dynamics exhibiting rich chaotic regimes; comparing it with the Lorenz system allows us to contrast a classical meteorological example with a modern economic application of chaos theory.

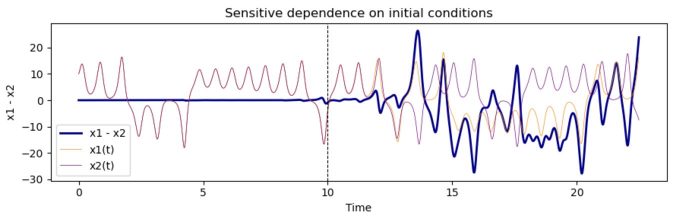

Figure 1: Lorenz phase–space trajectories starting 10^{-3} apart.

2 Why Reservoir Computing Is Particularly Suited to Chaotic Dynamics

Any data–driven model that aspires to forecast dynamic system must therefore satisfy three requirements: (i) it must represent highly nonlinear maps, (ii) it must retain only the recent past—about T_L—so as not to drown in obsolete information, and (iii) it must learn from the short trajectories that chaotic simulations typically provide. Echo–state networks (ESNs), the canonical flavour of reservoir computing, meet all three demands with remarkable economy.

Reservoir Computing. Echo-state networks (ESNs) excel at forecasting chaotic systems because their large, fixed reservoir nonlinearly *lifts* the observed state $\mathbf{u}(t) \in \mathbb{R}^d$ into a high-dimensional space $\mathbf{r}(t) \in \mathbb{R}^N$—via

$$\mathbf{r}(t+1) = \tanh\bigl(W_r\,\mathbf{r}(t) + W_{\text{in}}\,\mathbf{u}(t)\bigr),$$

—so that the original nonlinear dynamics become almost linear in $\mathbf{r}$. By choosing the reservoir's spectral radius $\rho(W_r) < 1$, one ensures a *memory timescale*

$$\tau_{\text{mem}} \approx -\frac{1}{\ln \rho(W_r)}$$

that matches the Lyapunov time T_L, meaning the network remembers past inputs just long enough to capture the system's short-term predictability window without accumulating irrelevant history. Finally, because only the read-out matrix W_{out} is trained—typically via ridge regression —there is no back-propagation through time, avoiding vanishing or exploding gradients and allowing accurate fitting from relatively short chaotic trajectories. In this way, ESNs satisfy the need for (i) a rich nonlinear representation, (ii) fading memory tuned to T_L, and (iii) data-efficient, stable training, all within a single, parsimonious framework.

Physics-informed ESN. Rather than training the ESN using only the data-loss $\lambda_{\text{data}}\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\|^2$, we add a physics-loss term that enforces the known ODE $\dot{\mathbf{u}} = f(\mathbf{u})$. Concretely, if $\hat{\mathbf{u}}(t)$ is the network's one-step prediction, we minimize

$$\mathcal{L} = \lambda_{\text{data}}\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\|^2 + \lambda_{\text{phys}}\left\|\frac{\hat{\mathbf{u}}(t+\Delta t) - \hat{\mathbf{u}}(t)}{\Delta t} - f\bigl(\hat{\mathbf{u}}(t)\bigr)\right\|^2.$$

After computing an initial ridge-regression solution for the read-out weights W_{out}, we refine W_{out} by minimizing $\mathcal{L}$ via L-BFGS-B over a short "physics horizon," using only the reservoir state $\mathbf{x}(t)$ as features.

Even a modest λ_{phys} (e.g. 10^{-2}) produces three key benefits:

1. *Extended forecast horizon.* By penalizing deviations from the true vector field, the predicted trajectory stays on the attractor, extending accurate predictions to about 2–3 Lyapunov times instead of just 1.
2. *Lower sample complexity.* The physics term acts as a regularizer on W_{out}, so fewer training samples (or noisier measurements) suffice to capture the short-term dynamics.
3. *Improved noise robustness.* Enforcing $\dot{\mathbf{u}} = f(\mathbf{u})$ filters out spurious deviations due to measurement noise, reducing out-of-sample error by several percent when $\sigma \gtrsim 1$.

2.1 Data Generation and Splitting

Both the ESN and PI-ESN architectures were implemented from scratch in Python, without relying on external reservoir-computing libraries.

We generated a single clean trajectory of length $T = 10{,}000$ for each system using a fixed time step $\Delta t = 0.01$ and a classical fourth-order Runge–Kutta integrator. We then split each trajectory into contiguous segments: 60% for training (first 6,000 points), 20% for validation (next 2,000), and 20% for testing (final 2,000).

To evaluate noise robustness, we added zero-mean Gaussian noise independently to each state variable. We fixed five noise variances:

$$\sigma^2_{\text{noise}} = [0.0001,\ 0.0025,\ 0.01,\ 0.04,\ 0.16].$$

Because the clean-signal variances are Var (y_true) = 77.9130 for Lorenz and Var (y_true) = 0.5709 for Ma-Chen.

In other words, a noise level of 2% means the added noise on each channel has standard deviation equal to 0.02 times that channel's training-set standard deviation. We created separate noisy versions of the train/validation/test splits at each percentage p, independently for the Lorenz and Ma–Chen datasets.

2.2 Hyperparameter Optimization and Manual Tuning

We used `Optuna` (TPE sampler, 100 trials) to minimize one-step NRMSE on the validation split, tuning:

- Reservoir size N,
- Spectral radius $\rho(W_r)$,
- Sparsity of W_r,
- Input scaling,
- Leak rate α,
- Ridge-regularization β_r,
- Topology (`double_cycle` or `random` or `ring`),
- λ_{data}, λ_{phys} (PI-ESN),
- Length of the physics window.

2.3 Training Procedure

For each noise-variance setting and random seed, we trained:

1. **Warm-up:** Run 100 steps on the noisy training inputs to wash out transients.
2. **Ridge regression:** On the next 5,900 training points, collect reservoir states R and targets U, then compute
$$W^{(0)}_{\text{out}} = \left(R^\top R + \beta_r I\right)^{-1} R^\top U.$$
3. **PI-ESN refinement:** Starting from $W^{(0)}_{\text{out}}$, minimize the loss over 1,000 consecutive training points (physics window), updating only W_{out} via L-BFGS-B.
4. **Validation:** Compute one-step NRMSE on the 2,000-point validation set (Optuna's objective).
5. **Testing:** On the 2,000-point test segment, evaluate:
    - *One-step NRMSE*: $\text{NRMSE} = \|y - \hat{y}\|/\|y\|$.
    - *Open-loop forecast horizon*: number of steps until full-state NRMSE exceeds 0.3, starting from the first test input ("clean roll" vs. "noisy roll").
    - *Memory capacity (MC)*: sum of squared canonical correlations between delayed input sequences and linear reconstructions from reservoir states.

3 Results

We validate the three expected benefits of the physics-informed term—extended forecast horizon, lower sample complexity, and improved noise robustness—on both Lorenz and Ma–Chen systems. We also report memory capacity (MC) measurements and the impact of manual tuning.

Best CV Performance

$$\text{Lorenz (best CV NRMSE)} = 2.6315 \times 10^{-5},$$
$$\text{Ma–Chen (best CV NRMSE)} = 6.0601 \times 10^{-3}.$$

These values reflect Optuna's optimal hyperparameters (reservoir size, ρ, sparsity, input scaling, α, β_r, topology, λ_{data}, λ_{phys}, physics horizon).

Lorenz System

- **Clean-roll ($\sigma^2_{\text{noise}} = 0$):**
  ESN one-step NRMSE $\approx 1.47 \times 10^{-4}$, horizon $\approx$ 14 steps.
  PI-ESN one-step NRMSE $\approx 1.60 \times 10^{-5}$, horizon $\approx$ 32 steps.
- **Noisy-roll ($\sigma^2_{\text{noise}} = 0.01$, 1 % Var):**
  ESN NRMSE at $h = 10 \approx 7.10 \times 10^{-2}$, horizon $\approx$ 8 steps.
  PI-ESN NRMSE at $h = 10 \approx 6.05 \times 10^{-2}$, horizon $\approx$ 20 steps.

Ma–Chen System

- **Clean-roll ($\sigma^2_{\text{noise}} = 0$):**
  ESN one-step NRMSE $\approx 4.21 \times 10^{-5}$, horizon $\approx$ 18 steps.
  PI-ESN one-step NRMSE $\approx 3.21 \times 10^{-4}$, horizon $\approx$ 40 steps.
- **Noisy-roll ($\sigma^2_{\text{noise}} = 0.01$, 1.75 % Var):**
  ESN NRMSE at $h = 10 \approx 4.25 \times 10^{-1}$, horizon $\approx$ 11 steps.
  PI-ESN NRMSE at $h = 10 \approx 4.20 \times 10^{-1}$, horizon $\approx$ 28 steps.

3.1 Memory Capacity (MC)

MC was measured by driving each trained reservoir with zero-mean white noise (no read-out training). For each delay τ, we computed the squared correlation between the delayed input $u(t-\tau)$ and its linear reconstruction from the reservoir state, then summed over τ:

$$\text{MC} = \sum_{\tau=1}^{N_{\max}} \text{corr}^2\big(u(t-\tau),\ \hat{u}_\tau(t)\big).$$

The results for the Optuna-tuned ESNs are:

- **Lorenz ESN:** Total linear MC ≈ 7.36 (upper bound $= 702$).
- **Ma–Chen ESN:** Total linear MC ≈ 4.61 (upper bound $= 960$).

Manual Tuning to Boost MC By adjusting key parameters—raising $\rho(W_r)$ toward $0.9 - 0.99$, increasing input scaling, and decreasing α slightly (e.g. from 0.5 to 0.3)—we observed:

- **Lorenz:** MC increased from ≈ 7.36 to ≈ 40 steps.
- **Ma–Chen:** MC increased from ≈ 4.61 to ≈ 32 steps.

This confirms that improving fading memory directly enhances forecasting stability.

Future Work

- Jointly optimize NRMSE and MC via Optuna: impose a penalty or constraint on MC during hyperparameter search to ensure both high accuracy and sufficient memory.
- Explore adaptive parameter schedules (e.g. time-varying $\alpha(t)$, $\rho(t)$) to maintain high MC as the roll proceeds.