JUMP-AUGMENTED RESERVOIR COMPUTING FOR TRAJECTORY FORECASTING AND MEASUREMENT CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Echo state networks (ESNs) replace gradient-based recurrent training with a fixed, randomly generated reservoir and a single linear readout. We introduce *hybrid reservoirs*, in which we augment the dynamics of the reservoir state to allow for discrete, event-triggered jumps that reset the reservoir state instantaneously. Deciding when and where to jump is a challenging problem that is usually application-specific. In this work, we provide a general formulation of hybrid reservoirs and showcase their potential on trajectory forecasting and measurement correction. For both applications, a well-chosen jump map lets the reservoir be reset into the right region of its state space, improving performance over a standard, jump-free ESN without sacrificing the closed-form, replay-free training that makes reservoir computing attractive in the first place.

1 INTRODUCTION

Reservoir computing, and echo state networks (ESNs) in particular, replace the gradient-based training of recurrent neural networks with a fixed, randomly generated recurrent reservoir and a single linear readout fit by (ridge) regression (Jaeger, 2001). Because the readout is obtained in closed form rather than by iterating gradient descent, ESNs are immune by construction to several pathologies that afflict gradient-based training, such as vanishing and exploding gradients and the sensitivity of the final solution to the optimization trajectory. This good behavior rests on the reservoir state evolving through a single, uninterrupted continuous-time (or discretized) flow, and is formalized by the *echo state property*: under mild conditions, the reservoir's internal trajectory depends asymptotically only on the history of its inputs, not on its own, essentially arbitrary, initial condition. This property is what makes the reservoir a well-defined feature map in the first place, yet it also encodes an implicit assumption: that whatever the reservoir needs to represent can always be reached smoothly, one nudge of the input at a time, from wherever the state currently sits. Many settings of practical interest violate this assumption outright. A dynamical system being forecast may itself switch abruptly between regimes; a new, exact measurement of that system may arrive mid-rollout and render the reservoir's current internal state stale rather than merely imprecise. In each case, the natural fix is the same: let the reservoir state *jump*, resetting itself discontinuously in response to an external event, instead of only being pushed toward the desired region of state space through the input channel and left for the recurrence to absorb over many steps.

We refer to a reservoir augmented with such resets as a *hybrid reservoir*, borrowing the flow-and-jump formalism of hybrid dynamical systems Goebel et al. (2012): a continuous-time flow governs the state on a portion of the state space while at the same time the state can be instantaneously reset through a jump map on a designated jump set. In this paper, we translate the broad mathematical framework of hybrid systems to reservoir dynamics and demonstrate its potential, giving intuition and specific examples for two distinct applications. We show how hybrid reservoirs can overcome the limitation of standard ESN to predict discontinuous time-dependent signals and demonstrate how hybrid reservoirs are able to correct the current prediction of a trajectory when a new measurement is available. In both cases, our approach improves significantly over a standard jump-free ESN.

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

1.1 Related work

Since the original introduction of ESNs in (Jaeger, 2001), the reservoir computing paradigm has been applied to a wide range of problems, including time series prediction for chaotic systems (Pathak et al., 2018), traffic dynamics (Ando & Chang, 2019; Fukuzaki et al., 2025), control (McCreesh & Cortés, 2024; Williams et al., 2024), continual learning and federated learning (Bacciu et al., 2021). Multiple works have focused their efforts in expanding the capabilities of a standard ESN through different architectures. The work (Du et al., 2024) suggests the use of an extended state with a discrete variable allowing the same reservoir to learn different attractors. (Gallicchio et al., 2017) introduced Deep ESNs, a multi-reservoir architecture that concatenates ESN instances to expand the representation capabilities. Other approaches include multi-readouts heads (Tanaka & Tamukoh, 2023b;a), allowing the same reservoir to learn completely different tasks. Most of these works successfully expand the capabilities of ESN at the cost of increasing the number of parameters to be trained, introducing deep architectures, or losing the closed-form training of the readout. Instead, the approach proposed here is able to keep the closed-form training of the readout, while expanding the capabilities of the reservoir through changing its dynamics, introducing discrete jumps in the reservoir space. This idea was first introduced in McCreesh et al. (2026), which formally established a notion of solution and sufficient conditions for the echo state property to hold for hybrid reservoir, with promising results on a basic hybrid benchmark, the bouncing ball. However, the definition of jump maps and sets did not include inputs, which limits the opportunities to exploit the expressivity of the network. We extend here the notion of hybrid reservoir along two axes: we allow the jump set to depend on the input as well as on the reservoir state, rather than on the state alone, and we let the jump map itself be chosen freely for each application, rather than fixed to a single functional form. We showcase the resulting versatility in two applications, trajectory forecasting and measurement correction.

1.2 Contributions

Our contributions are threefold. First, we formalize the *hybrid reservoir* (Section 3): a leaky-integrator ESN whose continuous flow is interrupted by an application-specific jump map g on a jump set $\mathcal{D}$ that may depend on both the reservoir state and the input. Second, we instantiate this framework for trajectory forecasting (Section 4.1), where the jump map resets the reservoir state at the true events of a hybrid dynamical system, letting the readout reproduce sharp, discontinuous changes that a standard, jump-free ESN cannot represent. Third, we present a novel reservoir correction strategy (Section 4.2), where the jump map uses a local linearization of the reservoir map to fold an intermittent, noise-free measurement back into the reservoir state, correcting a closed-loop rollout without retraining or replaying past data. In the forecasting task, the hybrid reservoir achieves near-perfect performance with no diverging or collapsing behavior, and in the correction task it sustains practically infinite-horizon prediction. In both applications, the hybrid reservoir yields significant improvements over conventional, jump-free ESNs, advancing the functional scope of standard ESN models.

2 Background

In this section we formally introduce the formalism of echo state networks (ESNs) and hybrid dynamical systems, which are the building blocks of our hybrid reservoir computing framework.

2.1 Echo state networks

Echo State Networks (ESNs) are a class of recurrent neural networks (RNNs). They consist of three main components: an input layer, a reservoir, and a readout layer. The input is a sequence $\{u_k\} \subset \mathbb{R}^{d_{\text{in}}}$, where d_{in} is the input dimension. The reservoir is a high-dimensional dynamical system whose internal state at time k is denoted by $r_k \in \mathbb{R}^{d_r}$, with $d_r \gg d_{\text{in}}$ typically. The fixed input matrix $W_{\text{in}} \in \mathbb{R}^{d_r \times d_{\text{in}}}$ projects the external input into the reservoir, while the recurrent connectivity matrix $W_r \in \mathbb{R}^{d_r \times d_r}$ defines how the internal state evolves over time. Finally, the readout matrix $W_{\text{out}} \in \mathbb{R}^{d_{\text{out}} \times d_r}$ linearly maps the reservoir state to the output via $y_k = W_{\text{out}} r_k$. Together, the ESN can be summarized as a dynamical system that transforms an input sequence into an output sequence through the following nonlinear dynamics:

$$\begin{cases} r_{k+1} = F(r_k, u_k) = (1-\alpha) r_k + \alpha\, \sigma(W_r r_k + W_{\text{in}} u_k + b) \\ y_k = W_{\text{out}} r_k \end{cases} \tag{1}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ is a fixed nonlinearity applied element-wise, and $b \in \mathbb{R}^{d_r}$ is a bias term that helps with its expressivity. Unless otherwise noted, we take $\sigma = \tanh$, the standard choice in the ESN literature. The scalar $\alpha \in [0, 1)$ is the leaking rate, which controls how quickly the dynamics forget previous states. Figure 1 illustrates the structure of an ESN.

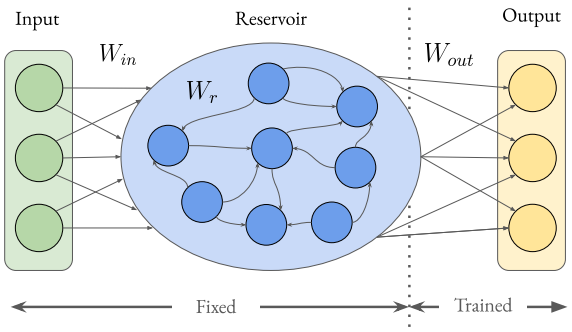

Figure 1: Echo state network structure.

The ESN aims to approximate a true relationship between the inputs and the outputs (i.e the output map $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$), but never has direct access to f itself: instead, an input sequence $u_{1:k}$ is first lifted into the high-dimensional reservoir space through the dynamical system of Eq. (1), producing a state $r_k = \mathcal{L}(u_{1:k}, r_0)$ that also depends on the initial reservoir state r_0, then the readout matrix W_{out} is trained to map this high-dimensional state back down to a prediction, $y_k = W_{\text{out}} r_k$, that should approximate $f(u_k)$. We note that the relationship between u and r is not a direct function but the action of a dynamical system governed by (1). The *Echo State Property* ensures that the reservoir state asymptotically depends only on the history of the inputs rather than on the initial condition.

Definition 2.1 (Echo State Property). *For $k \geq 1$, let $\mathcal{L}(\cdot, \cdot) : \left(\mathbb{R}^{d_{\text{in}}}\right)^k \times \mathbb{R}^{d_r} \to \mathbb{R}^{d_r}$ be the reservoir lifting map that takes an input sequence $u_{1:k}$ and an initial state r_0 to the reservoir state reached after k steps of Eq.* (1)*, i.e., $\mathcal{L}(u_{1:k}, r_0) := r_k$, with $r_j = F(r_{j-1}, u_j)$ for $j = 1, \ldots, k$. The reservoir dynamics satisfy the echo state property if, for every input sequence $(u_j)_{j \geq 1}$ in $\mathbb{R}^{d_{\text{in}}}$ and every pair of initial states $r_0, r_0' \in \mathbb{R}^{d_r}$, it holds that $\lim_{k \to \infty} \left\| \mathcal{L}(u_{1:k}, r_0) - \mathcal{L}(u_{1:k}, r_0') \right\| = 0$.*

Property 2.1 makes the reservoir a well-defined feature map: it guarantees that the influence of the initial state r_0 is washed out as k grows, so that $\mathcal{L}(x_{1:k}, r_0)$ becomes, asymptotically, a function of the input history alone. Without it, two runs on the same input sequence but different random initializations produce different reservoir trajectories, and hence different predictions, which makes the readout trained on one initialization meaningless for another. Under the echo state property, we make an abuse of notation and write $\mathcal{L}(x_{1:k})$, without the dependence on the initial condition, since all initial conditions will produce asymptotically similar results for high enough k value.

2.1.1 Training

Since the reservoir itself or the input matrix W_{in} is typically randomly initialized and never trained, fitting an ESN reduces to a single linear regression problem. We take the design-matrix row to be the reservoir state itself, $\phi_k := r_k \in \mathbb{R}^{d_r}$, produced by Eq. (1). Given T such states together with matching targets $y_k \in \mathbb{R}^{n_y}$, $k = 1, \ldots, T$, collect them into a design matrix and a target matrix,

$$X = \begin{bmatrix} \phi_1^\top \\ \phi_2^\top \\ \vdots \\ \phi_T^\top \end{bmatrix} \in \mathbb{R}^{T \times d_r}, \qquad Y = \begin{bmatrix} y_1^\top \\ y_2^\top \\ \vdots \\ y_T^\top \end{bmatrix} \in \mathbb{R}^{T \times n_y}.$$

The readout weights are fit by ridge regression, i.e., by solving the regularized least-squares minimization problem

$$W_{\text{out}} = \underset{W \in \mathbb{R}^{n_y \times d_r}}{\arg\min} \; \|XW^\top - Y\|_F^2 + \beta \|W\|_F^2, \tag{2}$$

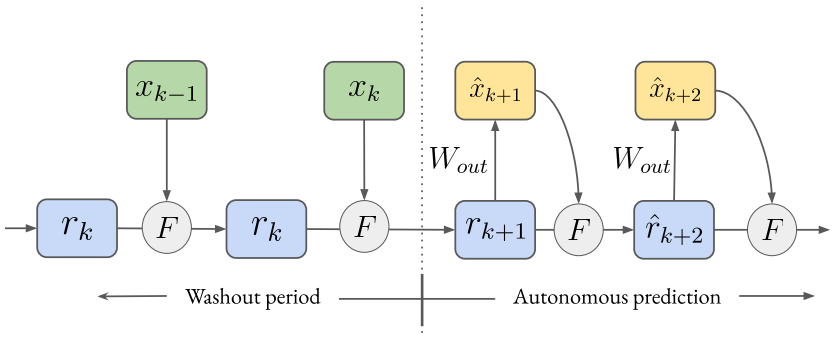

Figure 2: Transition from teacher-forced washout to autonomous, closed-loop rollout in prediction mode. The reservoir is first driven by the true signal x_k for $k = 1, \ldots, k_0$, discarding its outputs, and then switched over to closed-loop rollout, where the predicted output $\hat{x}_k$ is fed back as the next input.

where $\beta > 0$ is a regularization parameter and $\| \cdot \|_F$ the Frobenius norm. Because Eq. (2) is a quadratic in W, setting its gradient to zero yields the exact closed-form solution

$$W_{\text{out}} = \left(X^\top X + \beta I\right)^{-1} X^\top Y. \tag{3}$$

Eq. (3) is entirely agnostic to what the targets y_k represent: it is exactly this freedom that lets the same reservoir serve as either a predictor of time series or a classifier of whole sequences, depending only on how y_k is chosen.

We focus our attention on trajectory prediction (see Bianchi et al. (2021) for the implementation of RC for trajectory classification instead). When the task is to forecast the evolution of a continuous-time signal $x(t) \in \mathbb{R}^{d_{\text{out}}}$, we work with its value $x_k := x(t_k)$ at discrete times t_k, $k = 0, 1, 2, \ldots$, at which the reservoir is driven and read out, and write $x_{1:k} = (x_1, \ldots, x_k)$ for the resulting discrete trajectory. Identifying the generic ESN input with the signal itself, $u_k = x_k$, the common choice for the target is $y_k = x_{k+1}$ (or a target derived from it, e.g., the increment $x_{k+1} - x_k$), so that the readout learns to predict the next state from the current reservoir state. During training, the reservoir is always driven by the true signal x_k (teacher forcing), giving reservoir states $r_k = \mathcal{L}(x_{1:k})$ in the notation of Property 2.1. At inference time, the network is instead run in closed loop where instead of the true signal, the predicted output $\hat{x}_k = W_{\text{out}}\hat{r}_k$ is fed back as the next input, turning the reservoir update of Eq. (1) into an autoregressive map

$$\hat{r}_{k+1} = F(\hat{r}_k, \hat{x}_k), \qquad \hat{x}_{k+1} = W_{\text{out}}\hat{r}_{k+1}. \tag{4}$$

We write $\hat{r}_k = \mathcal{L}(\hat{x}_{1:k})$ for the closed-loop reservoir state reached this way, to distinguish it from the teacher-forced state $r_k = \mathcal{L}(x_{1:k})$ above. Before switching over to closed-loop rollout, the reservoir is first run for a short *washout period* of k_0 steps under teacher forcing, driven by the true signal $x_{1:k_0}$ with its outputs discarded: by Property 2.1, this drives the reservoir state away from its (typically random) initial condition r_0, so that $\hat{r}_{k_0} = r_{k_0}$ no longer depends on r_0 and closed-loop rollout can be seeded with a well-defined reservoir state rather than an arbitrary one. Figure 2 illustrates this transition from the teacher-forced washout to the autonomous, fed-back rollout. Because each predicted output becomes part of the next input, prediction errors compound over the rollout: $\hat{x}_{1:k}$ drifts away from $x_{1:k}$ as k grows, and correspondingly $\hat{r}_k$ drifts away from the teacher-forced target r_k (see Figure 5). Section 4.2 studies a complementary way, resetting the reservoir state itself whenever a ground-truth measurement becomes available.

Compared to gradient-based recurrent neural networks, ESNs offer several advantages: 1) training reduces to ridge regression on the readout, a convex problem with a unique closed-form solution, which avoids the difficulties of non-convex optimization such as local minima and sensitivity to initialization; and 2) because the reservoir weights are fixed, backpropagation through time is not required, sidestepping the vanishing and exploding gradient problem (Bengio et al., 1994). In gradient-trained RNNs, exploding gradients are typically mitigated by gradient clipping (Pascanu et al., 2013), while vanishing gradients are addressed by gated architectures such as Long Short Term Memories (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Neural Networks (Chung et al., 2014).

Despite these advantages, ESN performance is sensitive to reservoir design choices, such as spectral radius, sparsity, input scaling, and reservoir dimension. Next-generation reservoir computing (Gauthier et al., 2021; Kent et al., 2024) is a deterministic reformulation of the traditional ESN with fewer hyperparameters, showcasing a more robust and efficient approach. Nonetheless, sufficiently large reservoirs often exhibit remarkable expressive power thanks to the diversity of their internal dynamical responses. In this work we focus on expanding the capabilities of a classic ESN by allowing the reservoir to undergo discrete jumps.

2.2 Hybrid dynamical systems

Hybrid dynamical systems combine continuous-time state evolution with discrete state transitions Goebel et al. (2012); Sanfelice (2021). In general, the state $x \in \mathcal{X} \subseteq \mathbb{R}^d$ evolves continuously according to an input $u \in \mathcal{U}$, except when the state-input pair (x, u) enters a designated region, in which case the state may undergo an instantaneous reset. These two modes of evolution are characterized by the *flow set* $\mathcal{C} \subseteq \mathcal{X} \times \mathcal{U}$ and the *jump set* $\mathcal{D} \subseteq \mathcal{X} \times \mathcal{U}$. On $\mathcal{C}$, the state evolves according to the *flow map* $f : \mathcal{C} \to \mathbb{R}^d$, whereas on $\mathcal{D}$ it is reset according to the *jump map* $g : \mathcal{D} \to \mathcal{X}$. Formally, we represent this as

$$\begin{cases} \dot{x}(t) = f(x, u) & \text{if } (x, u) \in \mathcal{C} \\ x^+ = g(x, u) & \text{if } (x, u) \in \mathcal{D} \end{cases} \tag{5}$$

where x^+ denotes the post-jump state. Since the input u is not yet specified here, this describes an *open* hybrid control system, one whose evolution is driven by an external signal rather than closed under its own dynamics; and because $\mathcal{C}$ and $\mathcal{D}$ are defined jointly over the state-input pair (x, u), continuous evolution and jumps may be determined by the state alone, by the input alone, or by any combination of the two.

Nothing in this formalism restricts x to be the physical state of the system being modeled: x may instead contain auxiliary *logic variables* that the jump condition depends on, without changing Eq. (5) itself. A timer is one such variable: letting x include a component τ that flows as $\dot{\tau} = 1$ under f and resets to $\tau^+ = 0$ under g whenever a jump occurs lets the jump set $\mathcal{D}$ include states where τ exceeds a threshold. This allows us to define, for instance, time-triggered jumps in addition to event-triggered ones, all still within the same formalism of Eq. (5). In Section 4.2, we use such a timer to trigger jumps based on the time since the last measurement.

For simplicity, we restrict attention to solutions for which at most one jump can occur at any given time; i.e., consecutive jumps without flowing are excluded. This assumption allows us to use the simpler notation $x(t)$ for the state trajectory. When a jump occurs at time t_j, we distinguish the state immediately before and after the jump, with $x^-(t_j)$ and $x^+(t_j)$ respectively. Appendix C relates this notation to the more formal concept of hybrid time domain (Goebel et al., 2012, Section 2.2).

3 Hybrid reservoir computing

We now specialize the general hybrid dynamical system of Section 2.2 to the reservoir, augmenting the ordinary echo state network of Section 2.1 with a jump set and map. We first recall the reservoir's continuous-time flow on its own, then discuss the jumps, and finally discretize the resulting hybrid system into the dynamics we actually implement.

Identifying the hybrid state x with the reservoir state, $x = r \in \mathbb{R}^{d_r}$, the continuous-time limit of the leaky-integrator update of Eq. (1) gives the flow map

$$\dot{r} = f(r, u) := -Lr + \tanh(W_r r + W_{\text{in}} u + b), \tag{6}$$

where the continuous-time rate L plays the role of the leak rate α, an identification we make precise below. Taken on its own, Eq. (6) is exactly the jump-free reservoir of Section 2.1, evolving through the single, uninterrupted flow for which Property 2.1 is stated.

We now let the pair (r, u) trigger a reset, rather than restricting the jump set to reservoir states alone: letting it depend on the current input too, $\mathcal{D} \subseteq \mathbb{R}^{d_r} \times \mathcal{U}$, mirrors the general flow and jump sets $\mathcal{C}, \mathcal{D} \subseteq \mathcal{X} \times \mathcal{U}$ of Section 2.2. This lets a jump be triggered by an event carried by the input itself, such as a task switch or the arrival of a new measurement, rather than only by where the reservoir currently sits in state space. Writing $g : \mathcal{D} \to \mathbb{R}^{d_r}$ for the accompanying jump map, the hybrid

reservoir dynamics are

$$\begin{cases} \dot{r} = -Lr + \tanh(W_r r + W_{\text{in}} u + b) & \text{if } (r,u) \notin \mathcal{D} \\ r^+ = g(r,u) & \text{if } (r,u) \in \mathcal{D} \end{cases} \tag{7}$$

Both $\mathcal{D}$ and g are left unspecified for now: it is exactly this freedom in choosing what triggers a jump, and where it lands, that we exploit differently in each application in Section 4.

In practice, the continuous-time flow of Eq. (7) is discretized with a forward-Euler step, giving the implementable dynamics

$$\begin{cases} r_{k+1} = (1-\alpha) r_k + \alpha \tanh(W_r r_k + W_{\text{in}} u_k + b) & \text{if } (r_k,u_k) \notin \mathcal{D} \\ r_k^+ = g(r_k,u_k) & \text{if } (r_k,u_k) \in \mathcal{D} \end{cases} \tag{8}$$

where the identification $\alpha = hL$, with h interpreted as a step size, recovers the leak rate of Eq. (1) from the continuous-time rate L. Eq. (8) is the object we actually train and run in every experiment of the paper.

(McCreesh et al., 2026, Proposition4.1) establish that if (i) the jump map and jump set are both bounded, (ii) the number of jumps is finite, and (iii) the continuous-time dynamics alone satisfy the echo state property, then the hybrid reservoir dynamics of Eqs. (7)–(8) satisfy the echo state property of Definition 2.1. The hybrid reservoirs implemented in Section 4 satisfy all the conditions above except for the finiteness of jumps, which is not guaranteed in Application II (Subsection 4.2) since the jump set is defined to trigger whenever a new measurement arrives which can potentially be infinite. Nonetheless, the fact that they work well empirically is itself indirect evidence that the echo state property continues to hold for those jump maps as well.

Choosing the jump set $\mathcal{D}$ and jump map g is generally application-dependent and, in the context of reservoir computing, non-trivial: the high-dimensional and seemingly unstructured nature of the reservoir state space makes it difficult to interpret individual reservoir states, or to design meaningful jump conditions directly in that space. This is precisely why we let $\mathcal{D}$ depend on the input u as well as on the state r, since reasoning about the input space is typically far more intuitive. The jump map g itself still remains to be chosen, however: although the readout relates r to the output linearly, that relation is not uniquely invertible, since $d_r \gg d_{\text{out}}$. The next section proposes choices for $\mathcal{D}$ and g for the two tasks: hybrid dynamical systems forecasting and measurement correction.

4 Applications

4.1 Application I: Hybrid dynamics forecasting

A natural application of the HRC framework is its ability to accurately forecast the evolution of a hybrid dynamical system. Such systems exhibit in practice sharp changes in their dynamics (usually in their velocity vector fields) that are hard to capture with a standard ESN. Note that, if the evolution of the reservoir is continuous, the output will be a continuous function since it is linearly dependent on the reservoir state. This means that a standard ESN will have a fundamental limitation on forecasting hybrid dynamical systems, especially discontinuous states.

4.1.1 Hybrid reservoir design

For this case, we let the jump set $\mathcal{D}$ to be the same as the true hybrid dynamics and define the jump map of Eq. (8) to be

$$g(r_k,u_k) = \tanh(W_r r_k + W_{\text{in}} u_k + b) \tag{9}$$

This allows the reservoir state to jump and cause a discontinuity of the prediction, making the network able to accurately capture the hybrid dynamics. One can show that this jump map satisfies the echo state property under certain standard conditions (McCreesh et al., 2026).

Since forcing a standard ESN to follow discontinuous dynamics is not feasible, a possible workaround is to train only on the continuous (but usually non-differentiable) states. We refer to this modification as *RC-Continuous*. This reduces the complexity of the problem and stabilizes training, but at the cost of losing the ability to forecast the discontinuous state: because the readout is linearly related to the reservoir state, the resulting predictions will always be smooth, and any sharp change will be smoothed out.

4.1.2 EXPERIMENTS

We implement the hybrid reservoir of Eq. (8) with the jump map of Eq. (9) to forecast a hybrid, time-dependent dynamical system described in Appendix A.1.2: a driven mass-spring-damper oscillator that hits a rigid barrier, undergoing an instantaneous, energy-dissipating velocity reset on impact and producing a discontinuous trajectory. We compare three algorithms: *RC*, a standard ESN trained on both the continuous and discontinuous states; *RC-Continuous*, a standard ESN trained only on the continuous state (i.e. $x(t)$); and *HRC*, the hybrid reservoir trained on both. Figure 3 shows the minimal RMSE prediction for each algorithm: *HRC* clearly outperforms the other two, giving an almost perfect prediction. *HRC* is not only more accurate but also far more robust to hyperparameter choices and initial conditions (Figure 4): roughly 70% of standard-RC predictions collapse, and the best RC trajectory by RMSE is itself a collapsed one. Across hyperparameters, RC and RC-Continuous also diverge or produce physically inconsistent predictions, while *HRC* only ever suffers a moderate drop in performance. This illustrates a fundamental limitation of a jump-free ESN, whose continuous output cannot represent genuinely discontinuous behavior. Figure 8 (Appendix) visualizes the full range of behaviors for all three algorithms.

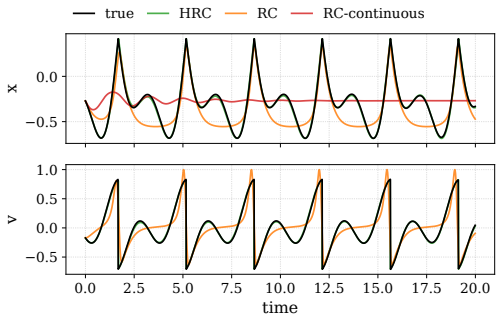

Figure 3: Minimal RMSE predicted trajectories of the impact oscillator system for three different algorithms explained in Section 4.1. Details in Appendix A.3.

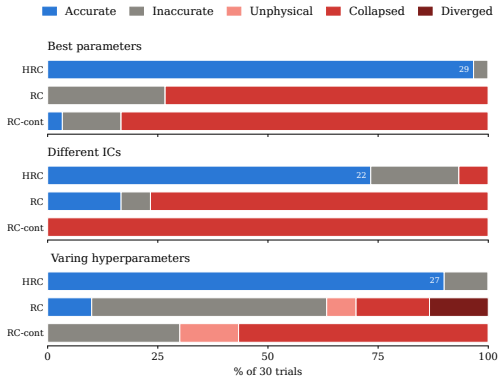

Figure 4: Robustness analysis of the impact oscillator forecasting task over 30 independent trials. Details in Appendix A.3.

4.2 APPLICATION II: TRAJECTORY FORECASTING CORRECTION

In this second application we showcase how the hybrid reservoir computing can be leveraged to update in real time a prediction whenever a noise-free measurement of the true state is available.

4.2.1 PROBLEM FORMULATION

Recall from Section 2.1.1 the continuous-time trajectory $x(t)$, its discretization $x_k := x(t_k)$, and the two reservoir states it induces: the teacher-forced state $r_k = \mathcal{L}(x_{1:k})$, and the closed-loop state $\hat{r}_k = \mathcal{L}(\hat{x}_{1:k})$ actually reached by feeding back the network's own predictions (Eq. (4)), which drifts away from r_k as the closed-loop rollout progresses (cf. Figure 5).

Suppose that, at some discrete time k during this closed-loop rollout, we receive a noise-free measurement $\bar{x}_k = x_k$ of the true state; measurements need not arrive at every step, and between arrivals the reservoir simply continues to flow in closed loop. The naive way to use $\bar{x}_k$ is to overwrite the forecast, $\hat{x}_k \leftarrow \bar{x}_k$, similar to the approach described in (Tkach & Andreev, 2026, Periodic Injections). This corrects the physical-space error, but leaves the reservoir state untouched, so the reservoir keeps carrying an internal state $\hat{r}_k$ that satisfies $W_{\text{out}}\hat{r}_k = \hat{x}_k \neq \bar{x}_k$, unlike the target state $r_k = \mathcal{L}(x_{1:k})$, which would satisfy $W_{\text{out}}r_k = \bar{x}_k$. What we would like instead is to change the reservoir state to this target r_k, the state it would be in had it been driven by the true trajectory all along. This is not achievable exactly, since computing r_k requires the entire true history $x_{1:k}$, of which only the latest sample $\bar{x}_k$ is ever observed; the only other information available is the estimated reservoir trajectory $\hat{r}_{1:k}$ already computed.

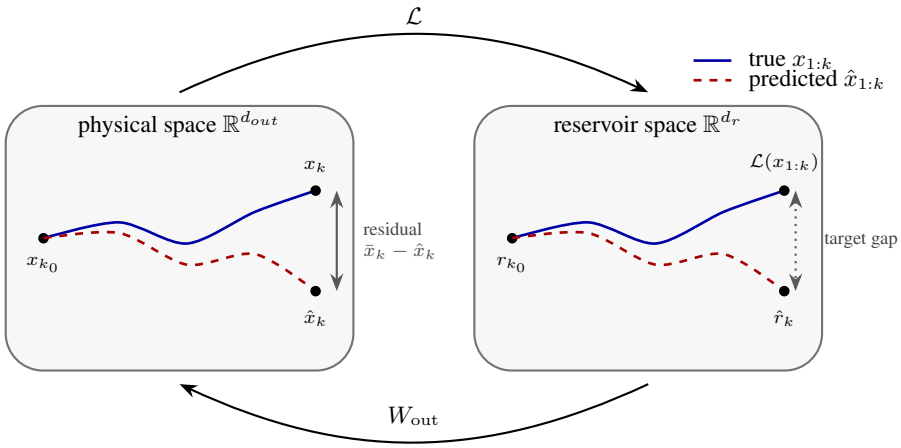

Figure 5: Relationship scheme between the reservoir space and the state space. The readout W_{out} maps the reservoir state r_k to the physical state $\bar{x}_k$ and the operator $\mathcal{L}$ lifts the physical state to the reservoir space. In open-loop, the predicted trajectory $\hat{x}_{1:k}$ drifts from the true trajectory $x_{1:k}$ in physical space (left), and the corresponding trajectories in reservoir space (right) also drift apart. A measurement $\bar{x}_k$ reveals the residual, but the target reservoir state $r_k = \mathcal{L}(x_{1:k})$ is never directly observed; only $\hat{r}_k$ is available.

The reset of the reservoir state fits exactly the framework of hybrid reservoirs laid out in Section 2.2. In fact, we augment the state with a timer τ and allow the system to jump any time a new measurement is received. Formally, we let the augmented reservoir state be $\tilde{r} = [\tau, r]$ and use the reset map $g(\tilde{r}, u) = [0, r^{\text{corr}}]$ whenever $(\tilde{r}, u) \in \mathcal{D} := \{(\tilde{r}, u) : \tau \geq \gamma\}$, where γ is the inter-sample time. According to the jump map, the timer τ is reset to zero, and the reservoir state is corrected to r^{corr}. We devote the remainder of this section to specify how to construct r^{corr} as a function of $u = \bar{x}_k$ and $r = \hat{r}_{1:k}$ to correct the reservoir state.

4.2.2 Correction via local linearization

Since the equation $W_{\text{out}} r_k = \bar{x}_k$ has infinitely many solutions in r_k ($d_r \gg d_{\text{out}}$), there is no unique way of constructing r_k^{corr} from the readout alone. The simplest option sets $r_k^{\text{corr}} \leftarrow W_{\text{out}}^{+} \bar{x}_k$, the readout's pseudo-inverse; this guarantees $W_{\text{out}} r_k^{\text{corr}} = \bar{x}_k$, but ignores the reservoir's own dynamics entirely and can place r_k^{corr} off the low-dimensional manifold that its flow actually visits, potentially destabilizing the closed-loop rollout.

We instead propose to construct r_k^{corr} from a local linearization of the problem, keeping the correction on the manifold that the reservoir's own dynamics actually visits rather than setting r_k^{corr} directly in an arbitrary direction. This approach keeps the correction on the reservoir's manifold by working in input space rather than setting r_k^{corr} directly: it asks what input, in place of $\hat{x}_{k-1}$, would have produced a state whose readout matches $\bar{x}_k$, and maps that input perturbation back through the reservoir's own recurrent map F. Writing $u_{k-1} := \hat{x}_{k-1}$ for the value actually fed back at the previous step, and linearizing $u \mapsto F(\hat{r}_{k-1}, u)$ around u_{k-1}, gives the Jacobian

$$J_k = \frac{\partial F}{\partial u}\Big|_{(\hat{r}_{k-1},\, u_{k-1})} = \alpha \,\mathrm{diag}\big(1 - \tanh^2(z_{k-1})\big)\, W_{\text{in}} \in \mathbb{R}^{d_r \times d_{\text{out}}} \tag{10}$$

where $z_{k-1} = W_r \hat{r}_{k-1} + W_{\text{in}} u_{k-1} + b$ is the pre-activation already computed when producing $\hat{r}_k$, so J_k is essentially free to obtain. Composing with the readout gives the local sensitivity of the predicted output $\hat{x}_k = W_{\text{out}} \hat{r}_k$ to the same input perturbation, $G_k = W_{\text{out}} J_k \in \mathbb{R}^{d_{\text{out}} \times d_{\text{out}}}$. Solving the resulting linear system with a pseudo-inverse (in case G_k is not invertible) for the input perturbation gives

$$\delta u_k = G_k^{+}\big(\bar{x}_k - \hat{x}_k\big), \qquad \hat{r}_k^{\text{corr}} = \mathrm{clip}\big(\hat{r}_k + J_k\, \delta u_k,\, -1,\, 1\big), \tag{11}$$

i.e., a single Gauss–Newton step, taken locally around the operating point $(\hat{r}_{k-1}, u_{k-1})$. The clip forces the corrected state to remain within the bounds of the reservoir's state space. Note that for a dynamics following Definition 2.1 with $\tanh$ as activation function the reservoir states are forced to remain bounded between -1 and 1. After the correction, the timer τ is reset to zero, and the closed-loop rollout resumes from the corrected reservoir state $\hat{r}_k^{\text{corr}}$ and the measured input $\bar{x}_k$.

Unlike the pseudo-inverse correction, both J_k and G_k depend on W_r and W_{in}, i.e., on the actual local geometry of the recurrent dynamics, so this correction only ever moves $\hat{r}_k$ along directions that some locally admissible input could have produced. Being a first-order approximation, its accuracy degrades when $\bar{x}_k - \hat{x}_k$ is large relative to the curvature of $\tanh$ at z_{k-1}, e.g., under sparse measurements with long uncorrected drift in-between, in the same way that an extended Kalman filter degrades relative to a full nonlinear update as the linearization is stretched further from its operating point (Wan & Van Der Merwe, 2000).

4.2.3 Experiments

We evaluate the correction schemes for the reservoir state described in the previous section predicting the evolution of the Lorenz and Rössler systems. A noiseless measurement of the true state is injected every k steps; between measurements the network runs open-loop on its own predictions. We compare the 3 conditions mentioned in the previous section: *free-running*, *input-corrected* (Tkach & Andreev, 2026, Periodic Injections) and the *reservoir-correction* presented in the previous section. The pseudoinverse correction was also tested but it is not shown in the results since it was consistently outperformed by the Jacobian correction. Results on the pseudoinverse correction and implementation details can be found in Appendix.

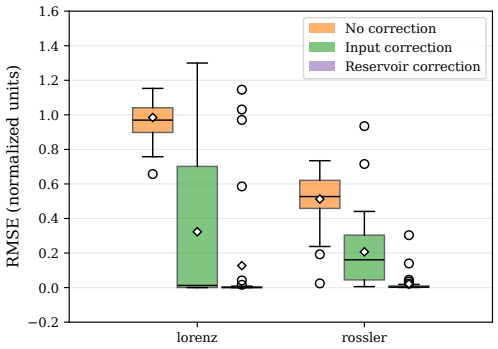

Figure 6: Boxplot reporting the normalized RMSE across 30 random seeds for each correction scheme. Details in Appendix A.3.

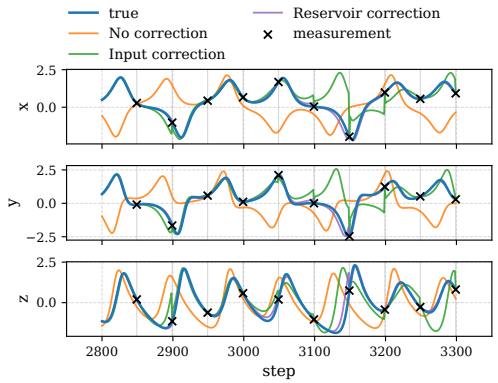

Figure 7: Lorenz trajectory predictions for the different correction schemes. Measurements are indicated by Xs. Details in Appendix A.3

Figure 6 shows the RMSE for 30 predicted trajectories. As can be seen, correcting the input alone induces an improvement over the no correction however, *Reservoir-Correction* clearly outperforms the other approaches achieving almost perfect tracking of the true trajectory. For some trials our approach was able to achieve perfect tracking for unlimited horizons being able to stay close to the true trajectory for at least 8000 steps. Figure 7 shows a zoom in on a portion of such trajectory one can see how even if the state is properly reset the mismatch between r and $\hat{r}$ makes the *Input-Corrected* trajectory drift away even just after a correction (e.g., Step 3100 in Figure 7).

5 Conclusion

We introduced the hybrid reservoir, an echo state network whose otherwise continuous flow is interrupted by discrete, event-triggered jumps, and showed that this single mechanism, instantiated through different choices of jump set and jump map, addresses two structurally distinct problems in reservoir computing: forecasting a dynamical system that is itself hybrid, and correcting a closed-loop trajectory forecast against intermittent ground-truth measurements. Across both applications, letting the reservoir be reset into the right region of its state space, rather than only ever nudging it there through the input channel, improved performance over a standard, jump-free ESN, while preserving the closed-form, replay-free readout training that makes reservoir computing attractive in the first place. Relatedly, all jump maps studied here were constructed by hand from domain knowledge of each problem; an important direction for future work is learning the jump set and jump map themselves from data, rather than specifying them a priori.

REFERENCES

H. Ando and H. Chang. Road traffic reservoir computing. *arXiv preprint arXiv:1912.00554*, 2019.

Davide Bacciu, Daniele Di Sarli, Pouria Faraji, Claudio Gallicchio, and Alessio Micheli. Federated Reservoir Computing Neural Networks. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, July 2021. doi: 10.1109/IJCNN52387.2021.9534035.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Filippo Maria Bianchi, Simone Scardapane, Sigurd Løkse, and Robert Jenssen. Reservoir Computing Approaches for Representation and Classification of Multivariate Time Series. *IEEE Transactions on Neural Networks and Learning Systems*, 32(5):2169–2179, May 2021. ISSN 2162-2388. doi: 10.1109/TNNLS.2020.3001377. URL https://ieeexplore.ieee.org/document/9127499/.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Yao Du, Haibo Luo, Jianmin Guo, Jinghua Xiao, Yizhen Yu, and Xingang Wang. Multi-functional reservoir computing, September 2024. URL http://arxiv.org/abs/2409.16719. arXiv:2409.16719 [nlin].

R. Fukuzaki, T. Noguchi, and H. Ando. Harvested reservoir computing from road traffic dynamics. *Scientific Reports*, 2025.

C. Gallicchio, A. Micheli, and L. Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa. Next generation reservoir computing. *Nature Communications*, 12(1):5564, 2021.

R. Goebel, R. G. Sanfelice, and A. R. Teel. *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton University Press, Princeton, NJ, 2012. ISBN 978-0-691-15389-6.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34):13, 2001.

R.M. Kent, W. A. S. Barbosa, and D. J. Gauthier. Controlling chaotic maps using next-generation reservoir computing. *Chaos*, 34(2), 2024.

M. McCreesh and J. Cortés. Control of linear-threshold brain networks via reservoir computing. *IEEE Open Journal of Control Systems*, 3:325–341, 2024.

M. McCreesh, A. Marzabal, A. Ganlath, R. Gupta, J. Sanchez, and J. Cortés. Reservoir computing with hybrid systems: establishing the echo state property. In *American Control Conference*, pp. 2838–2843, New Orleans, LA, May 2026.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2): 024102, 2018.

R. G. Sanfelice. *Hybrid Feedback Control*. Princeton University Press, Princeton, NJ, 2021.

Yuichiro Tanaka and Hakaru Tamukoh. Ensemble learning of multiple readouts for reservoir computing, 2023a.

Yuichiro Tanaka and Hakaru Tamukoh. Self-organizing multiple readouts for reservoir computing. *IEEE Access*, 11:138839–138849, 2023b. doi: 10.1109/ACCESS.2023.3340311.

Sofia Tkach and Andrey Andreev. State correction via data injection for a reservoir computing-based digital twin of the Lorenz system. *The European Physical Journal Special Topics*, July 2026. ISSN 1951-6401. doi: 10.1140/epjs/s11734-026-02537-y. URL https://doi.org/10.1140/epjs/s11734-026-02537-y.

E.A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pp. 153–158, 2000. doi: 10.1109/ASSPCC.2000.882463.

J. P. Williams, J. N. Kutz, and K. Manohar. Reservoir computing for system identification and predictive control with limited data. *arXiv preprint arXiv:2411.05016*, 2024.

A Implementation details

In this appendix we present the implementation details for the experiments sections.

A.1 Dynamical systems

We use three dynamical systems as benchmarks: the chaotic Lorenz and Rössler systems for the trajectory correction experiments of Section 4.2.3, and the impact oscillator, a hybrid system with discontinuous dynamics, for the hybrid dynamics forecasting experiments of Section 4.1.

A.1.1 Chaotic systems

Lorenz system. The Lorenz system is the three-dimensional ODE

$$\begin{aligned} \dot{x} &= \sigma(y - x), \\ \dot{y} &= x(\rho - z) - y, \\ \dot{z} &= xy - \beta z, \end{aligned} \tag{12}$$

with state $(x, y, z) \in \mathbb{R}^3$, a quadratic truncation of Rayleigh–Bénard convection whose classical chaotic regime, $\sigma = 10$, $\rho = 28$, $\beta = 8/3$, produces the familiar two-lobed "butterfly" attractor.

Rössler system. The Rössler system is the three-dimensional ODE

$$\begin{aligned} \dot{x} &= -y - z, \\ \dot{y} &= x + ay, \\ \dot{z} &= b + z(x - c), \end{aligned} \tag{13}$$

with state $(x, y, z) \in \mathbb{R}^3$. Its single quadratic term, zx in the $\dot{z}$ equation, gives a simpler, single-fold chaotic attractor than the Lorenz system's; we use the standard chaotic regime $a = 0.2$, $b = 0.2$, $c = 5.7$.

Numerical integration. For each system we numerically integrate the corresponding ODE with `scipy.integrate.solve_ivp`, using the adaptive-step, explicit Runge–Kutta method RK45 with tight tolerances ($\mathrm{rtol} = \mathrm{atol} = 10^{-9}$) and evaluating the solution on a uniform grid of step Δt. To ensure the returned trajectory lies on the attractor rather than reflecting the arbitrary initial condition x_0, we first integrate an initial transient of $n_{\mathrm{transient}}$ steps from x_0 at the same tolerances, discard it, and restart the integration from the state reached at its end. Table 1 lists Δt, x_0, and $n_{\mathrm{transient}}$ for each system.

Normalization. Each simulated trajectory (training segment and forecast horizon concatenated) is rescaled per axis to zero mean and unit standard deviation before training or evaluation. Reported RMSE values are in these normalized units, so $\mathrm{RMSE} \approx 1$ corresponds to no better than predicting the per-axis mean.

Table 1: Integration settings for the Lorenz and Rössler systems.

	Lorenz	Rössler
Step size Δt	0.01	0.1
Initial condition x_0	$(1, 1, 1)$	$(1, 1, 1)$
Transient steps $n_{\text{transient}}$	1000	1000

Lyapunov exponents. We estimate each system's leading Lyapunov exponent from nearby-trajectory divergence, with the same procedure for both systems: $\lambda_{\text{Lorenz}} \approx 0.868\,\text{time}^{-1}$ and $\lambda_{\text{Rössler}} \approx 0.095\,\text{time}^{-1}$, i.e. Lorenz is roughly $9\times$ more chaotic.

A.1.2 Hybrid system

Impact oscillator. Application I (Section 4.1) forecasts the trajectory of a driven mass-spring-damper system with a rigid barrier at $x = x_{\max}$, In its hybrid form with state $\mathbf{z} = (x, v)^\top$, $v = \dot{x}$:

$$\mathcal{H}: \begin{cases} \dot{\mathbf{z}} = f(t, \mathbf{z}) = \begin{pmatrix} v \\ \frac{1}{m}\big(A\sin(\omega t) - c\,v - k\,x\big) \end{pmatrix}, & \mathbf{z} \in \mathcal{C} = \{(x, v) : x < x_{\max} \text{ or } v \le 0\}, \\ \mathbf{z}^+ = g(\mathbf{z}) = \begin{pmatrix} x_{\max} \\ -e\,v \end{pmatrix}, & \mathbf{z} \in \mathcal{D} = \{(x, v) : x \ge x_{\max},\ v > 0\}, \end{cases} \tag{14}$$

where x is the position, v the velocity, k the spring constant, c the damping coefficient, m the mass, and $e \in (0, 1)$ the coefficient of restitution: on impact ($x \ge x_{\max}$, $v > 0$) the position is clamped to the barrier and the velocity is instantaneously reversed and damped, producing the discontinuous trajectory used to demonstrate the limits of a jump-free ESN. We use $m = 1$, $c = 0.2$, $k = 4$, $A = 6$, $\omega = 1.8$, $x_{\max} = 1$, $e = 0.85$, integrated with fixed-step RK4 ($\Delta t = 0.01$), checking the guard $\mathcal{D}$ and applying the reset g after every step.

A.2 Reservoir hyperparameters

A.2.1 Application I

For the first application the hyperparameters were independently grid-searched for each model (HRC, RC, and RC-cont) over the ranges given in Table 2. The table reports the best configuration for each model, selected as the one with the lowest closed-loop forecast RMSE over 2000 steps among configurations that did not diverge. The lower block of Table 2 lists the fixed hyperparameters shared by all models.

Table 2: Reservoir hyperparameters for the impact oscillator. The upper block was grid-searched separately for each model (648 configurations for HRC, 108 each for RC and RC-cont). Each model's value is the configuration with the lowest closed-loop forecast RMSE over 2000 steps, among configurations that did not diverge. The jump map exists only in HRC. The lower block is fixed and shared by all models. W_r and W_J have entries drawn from $\mathcal{U}(-1,1)$, are masked to the given connection density and rescaled to their spectral radius; $W_{\text{in}} \sim \mathcal{U}(-s_{\text{in}}, s_{\text{in}})$, $b \sim \mathcal{U}(-s_b, s_b)$ and $b_J \sim \mathcal{U}(-s_J, s_J)$.

Hyperparameter	Grid	HRC	RC	RC-cont
Grid-searched				
Spectral radius $\rho(W_r)$	$\{0.8,\ 0.95,\ 1.1\}$	1.1	0.95	1.1
Input scaling s_{in}	$\{0.3,\ 0.5,\ 0.7\}$	0.5	0.3	0.7
Leak rate α	$\{0.6,\ 0.8,\ 1.0\}$	0.6	1.0	1.0
Ridge regularization β	$\{10^{-4}, \ldots, 10^{-1}\}$	10^{-2}	10^{-1}	10^{-2}
Jump-map spectral radius $\rho(W_J)$	$\{0.2,\ 0.4,\ 0.6\}$	0.2	–	–
Jump-map bias scale s_J	$\{0.2,\ 0.5\}$	0.2	–	–
Fixed				
Inputs (fed back)		(x, v)	(x, v)	x
Reservoir size d_r			300	
Connection density of W_r, W_J			0.2	
Bias scale s_b			1.0	
Washout length k_0			200 steps	
Training / forecast length			6000 / 2000 steps	
Integration step h			0.01 s	

A.2.2 Application II

Architecture and training. We use the leaky-integrator ESN of Eq. (1). W_r is drawn i.i.d. $\mathcal{U}(-1,1)$, sparsified with a Bernoulli mask at the density given in Table 3, then rescaled to the target spectral radius; W_{in} is drawn i.i.d. $\mathcal{U}(-s_{\text{in}}, s_{\text{in}})$ and b i.i.d. $\mathcal{U}(-s_b, s_b)$. The reservoir is teacher-forced on the training segment, the first k_0 states are discarded as washout, and the readout is fit in closed form by Eq. (3). For the hyperparameter choice we performed an independent grid search over 512 combinations for the relevant parameters, for a fixed reservoir size $d_r = 300$; Table 3 shows the best parameters found for each system, together with the trajectory lengths and measurement period used in the correction experiments of Section 4.2.3. The reservoir-weight seed used for these single-trial results is 100.

Table 3: Best hyperparameters used for the Lorenz and R"ossler systems.

Hyperparameter	Lorenz	R"ossler
Reservoir size d_r	300	300
Spectral radius $\rho(W_r)$	0.65	0.5
Input scaling s_{in}	0.5	0.2
Connection density of W_r	0.2	0.2
Leak rate α	0.4	0.6
Bias scale s_b	1.0	1.0
Ridge regularization β	10^{-8}	10^{-8}
Washout length k_0	200	200
Discretization step h	0.01	0.1
Training length	4000 steps (40 s)	1000 steps (100 s)
Forecast length	4000 steps (40 s)	4000 steps (400 s)
RMSE horizon	2000 steps (20 s)	2000 steps (200 s)
Measurement period	0.5 s (50 steps)	30 s (300 steps)

A.3 Figures details

Best parameters prediction (Figure 3) . For the impact oscillator, we run a single trial with the best found parameters (given by Table 2). The plot shows the resulting predicted trajectory for the position and velocity variables. HRC is able to accurately predict the trajectory with minimal deviation. RC

predicts a periodic signal but that is far from the actual true trajectory and RC-continuous fails to predict any trajectory collapsing to a single point. This results shows the minimal RMSE that was found for all methods after running a hyperparameter grid-search (see Table 2). The RMSE values for each method are reported in Table 4.

Training fragility (Figure 4) Each model (HRC, RC, RC-cont) keeps its grid-search optimum of Table 2. Each panel is one perturbation condition with $N = 30$ trials. From top to bottom: (i) the reservoir seed (30 new seeds, default trajectory); (ii) the initial condition of the simulated trajectory, $x_0 \sim \mathcal{U}(-1.5,\, 0.95)$, $v_0 \sim \mathcal{U}(-3,\, 3)$; (Varing hyperparameters) each hyperparameter is multiplied by $e^{\sigma z}$ with $z \sim \mathcal{N}(0, 1)$ drawn per trial and hyperparameter, for $\sigma = 0.05,\, 0.1,\, 0.2$. In (iv–vi) the ridge parameter moves by $10^{5\sigma z}$ instead, the leak rate is capped at 1, and the seed and trajectory stay fixed; HRC's jump-map hyperparameters are jittered as well. Each model then forecasts in closed loop for 20 s. Every forecast gets exactly one outcome, and the first matching class wins: *Diverged* if any fed-back output exceeds five times its largest training magnitude or becomes non-finite; *Collapsed* if the forecast stops oscillating, i.e. the standard deviation of $\hat{x}$ over the last 5 s (more than one forcing period) is below $0.1\times$ that of the true x; *Unphysical* if the forecast passes through the rigid stop, $\max_t \hat{x} > x_{\max} + 0.05$; *Inaccurate* if none of these occurs but the forecast RMSE_x exceeds 0.3 (scaled units); and *Accurate* otherwise. Each bar shows the share of the 30 trials falling in each class. The number in the blue segment is the count of accurate trials (shown when there are at least 6).

Multi-seed evaluation (Figure 6). For each system we run $N = 30$ independent trials. In each trial the initial condition is perturbed, $x_0 = (1, 1, 1) + \mathcal{N}(0, 0.5^2 I_3)$, and a fresh reservoir is initialized with its own weight seed, so every trial pairs an independently sampled trajectory with an independently initialized reservoir; all three schemes (free-running, input-corrected, reservoir-corrected) are then forecast from that trial's trained reservoir, and the RMSE is recorded as in Table 3, averaged over x, y, z. Box plots report the resulting per-scheme distribution across the $N = 30$ trials: box = interquartile range, line = median, whiskers extend to $1.5\times$IQR, and the mean is marked with a diamond.

Trajectory with correction (Figure 7) . For the Lorenz system, we run a random trial and zoom into steps $k \in [2800, 3300]$ of the forecasted trajectory. The free-running trajectory has already diverged from the true value. The *Input-Corrected* trajectory is initially able to follow the true trajectory, but drifts off again even with the periodic corrections: this supports the theoretical intuition of Section 4.2.1, that correcting only the output is not sufficient to prevent divergence, since the reservoir state itself is never corrected. In this particular trial, the *Reservoir-Corrected* trajectory closely tracks the true trajectory indefinitely, with only a minor divergence between measurements that is corrected away at the next one.

B Complementary results

B.1 Trajectory prediction experiments

Figure 8 shows the impact of varying initial conditions and reservoir hyperparameters on the closed-loop forecast of the impact oscillator, presenting 30 independent trials for each of the three models (HRC, RC, and RC-continuous). One can see the different behaviors explained in the previous section. Table 4 reports the success rate of each model for the different conditions. The table also reports the number of trials that diverged, collapsed, or produced unphysical trajectories, as well as the RMSE_x for the successful trials.

B.2 Trajectory correction experiments

Here we present additional results for the trajectory correction experiments of Section 4.2.3. Beyond the correction schemes presented in the main text, we also include the *Pseudoinverse* correction scheme discussed in Section 4.2.2, and rename what we previously called the *Reservoir-Corrected* scheme to *Linear approximation*. Figures 9 and 10 show the predicted trajectories for each correction scheme on the Lorenz and Rössler systems, respectively, over steps 2000–4000 of the forecasted trajectory: the proposed method achieves near-perfect tracking. Figure 11 shows the RMSE boxplots for each correction scheme. As predicted, the *Pseudoinverse* correction provides essentially no benefit, performing similarly to no correction at all.

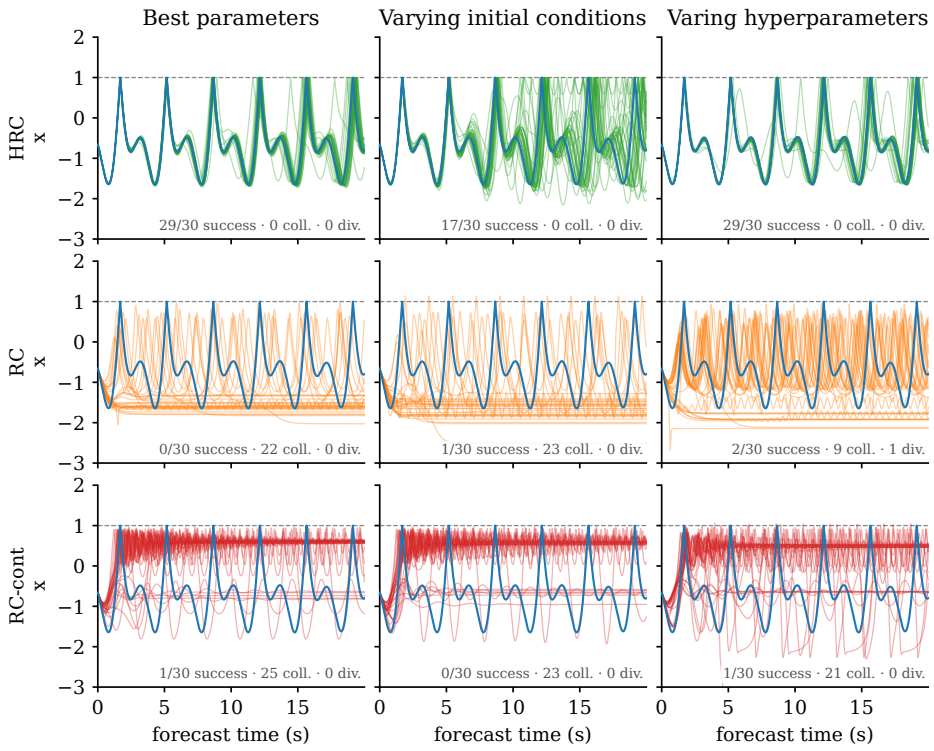

Figure 8: Different runs for the different implementations of the reservoir for varying initial conditions and reservoir hyperparameters.

Table 4: Impact oscillator forecasting results for the different correction methods. % Jitter indicates the percentage in which the parameters were deviated from the best observed parameters.

Condition	Model	Success	Inacc.	Stop	Coll.	Div.	RMSE_x
Best parameters							
(Figure 3)	HRC	1/1	0	0	0	0	0.01
	RC	1/1	0	0	0	0	0.17
	RC-continuous	0/1	0	0	1	0	0.25
Multiple seeds	HRC	29/30	1	0	0	0	0.08
	RC	0/30	8	0	22	0	0.44
	RC-continuous	1/30	4	0	25	0	0.55
Best parameters	HRC	30/30	0	0	0	0	0.04
+ 5% jitter	RC	2/30	17	0	11	0	0.36
	RC-continuous	2/30	4	0	24	0	0.46
Best parameters	HRC	29/30	1	0	0	0	0.06
+ 10% jitter	RC	2/30	18	0	9	1	0.36
	RC-continuous	1/30	8	0	21	0	0.52
Best parameters	HRC	27/30	3	0	0	0	0.09
+ 20% jitter	RC	3/30	16	2	5	4	0.36
	RC-continuous	0/30	9	4	17	0	0.49

C HYBRID TIME FORMALISM

A solution to (5) therefore consists of alternating intervals of continuous evolution and discrete jumps. This hybrid structure requires a notion of time that accounts not only for the elapsed continuous time but also for the number of jumps that have occurred. A standard formalism is given by a *hybrid*

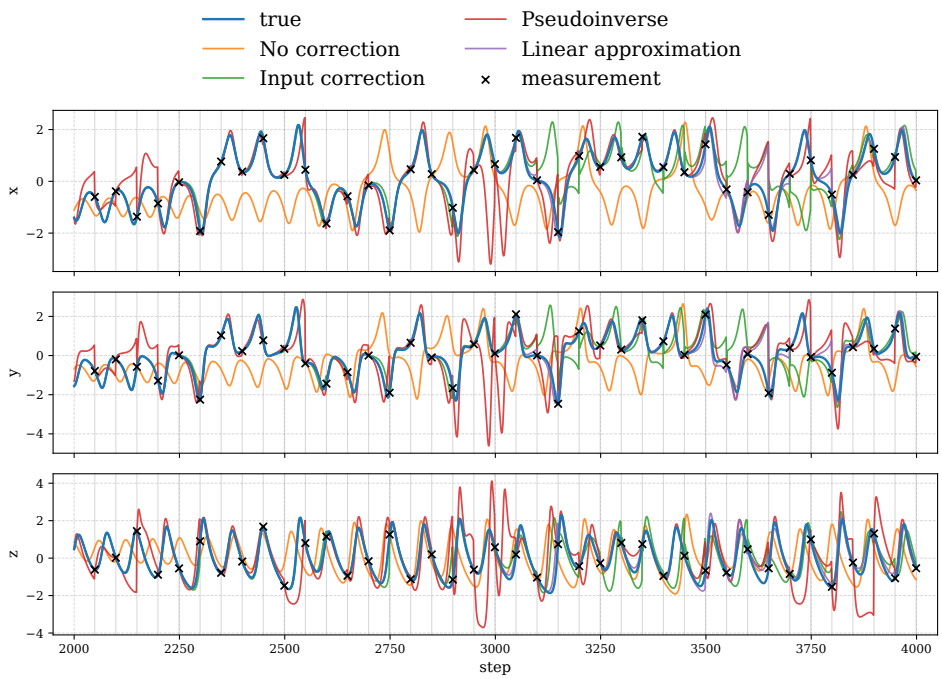

Figure 9: Predicted trajectories for the different correction schemes on the Lorenz system, over steps 2000–4000 of the forecasted trajectory. Measurements are indicated with X's.

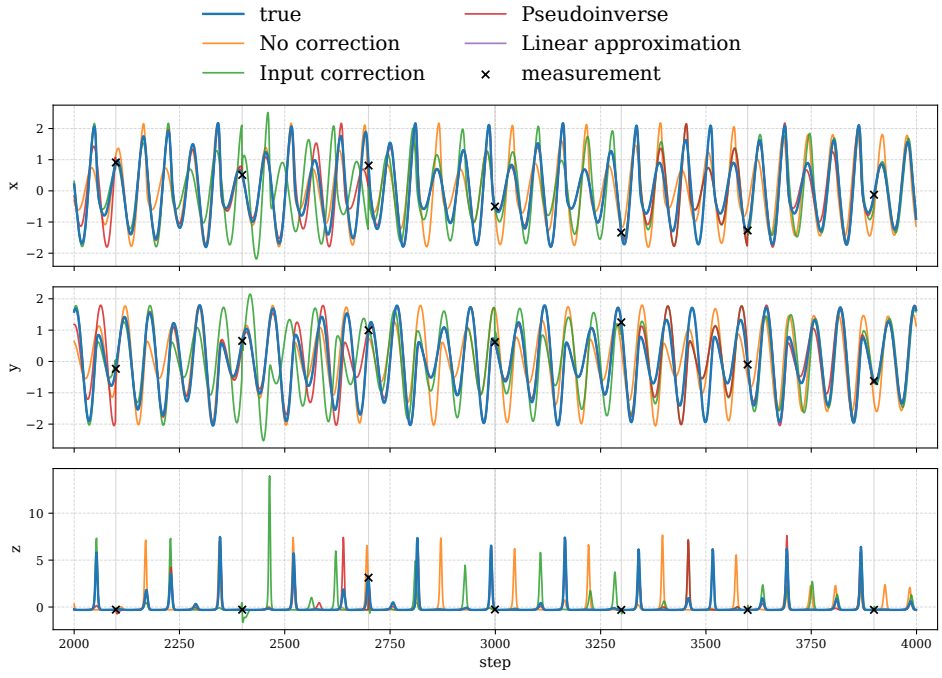

Figure 10: Predicted trajectories for the different correction schemes on the Rössler system, over steps 2000–4000 of the forecasted trajectory. Measurements are indicated with X's.

time domain, represented as a sequence of intervals $[t_j, t_{j+1}]$, with $j = 0, \ldots, J$, where $t_0 = 0$ and $t_{j+1} > t_j$. The state flows according to f on each interval and, when the jump condition is satisfied,

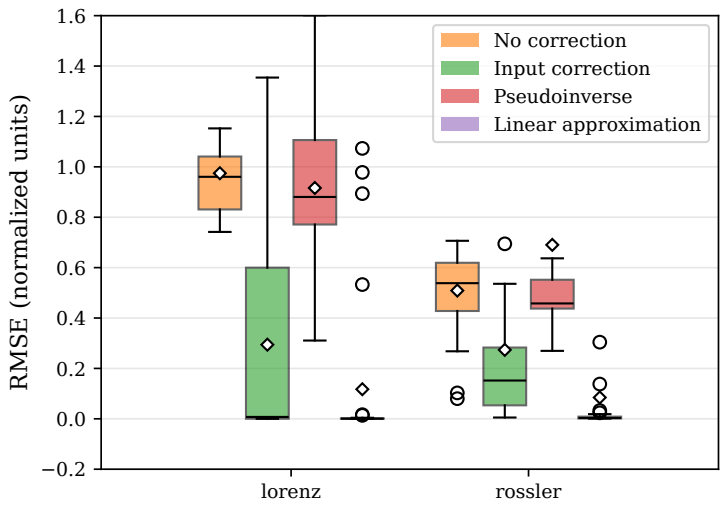

Figure 11: RMSE boxplot for different runs for the different correction implementations over the first 2000 steps of the forecasted trajectory.

is reset according to g at the corresponding endpoint. The index j records the number of jumps that have occurred.

More precisely, a solution is a pair (x, j) defined jointly over each interval $[t_j, t_{j+1}]$: for t in the interior of the interval, x evolves according to the flow map, $\dot{x}(t) = f(x(t), u(t))$, and the jump index j stays fixed. At the right endpoint $t = t_{j+1}$, if the state-input pair lies in the jump set, the state is reset instantaneously according to the jump map and the index increments, $j \mapsto j + 1$, marking the start of the next interval. Because the reset happens at a single instant of ordinary time t_{j+1}, the state is generally discontinuous there, and it becomes necessary to distinguish its value immediately before and immediately after the jump. Writing $x(t, j)$ for the state value at time t within the j-th interval, these two one-sided limits are exactly the pre- and post-jump states $x^-(t_{j+1}) = x(t_{j+1}, j)$ and $x^+(t_{j+1}) = x(t_{j+1}, j + 1)$ used throughout the paper: x^- is where the continuous flow of interval j was heading, and x^+ is where the jump map sends the state at the start of interval $j + 1$, i.e. $x^+(t_{j+1}) = g(x^-(t_{j+1}), u(t_{j+1}))$. This formulation excludes the possibility of multiple jumps occurring at the same instant of ordinary time, which is a possible behavior like Zeno solutions. For more in depth treatment of hybrid dynamical systems, we refer the reader to Goebel et al. (2012); Sanfelice (2021).