Received XX September 2021; revised ; accepted XX October 2021; Date of publication XX November 2021; date of current version XX November 2021. This paper was recommended by Associate Editor F. A. Author.

Digital Object Identifier 10.1109/OJCSYS.2021.Doi Number

ResDyNet: a Residual State-Space Neural Network for Nonlinear System Identification

MARCO LEDDA[1] (Student, IEEE), DIEGO DEPLANO[1] (Member, IEEE), ALESSANDRO GIUA[1] (Fellow, IEEE), MAURO FRANCESCHELLI[1] (Senior, IEEE)

[1]Department of Electrical and Electronic Engineering, University of Cagliari, Italy

CORRESPONDING AUTHOR: Mauro Franceschelli (e-mail: mauro.franceschelli@unica.it)

This work is partially supported by Fondazione di Sardegna with project “Artificial Intelligence for Smart Technologies (AISTech)”, call 2023, F23C25000310007 and by Fondazione di Sardegna with project ”Formal methods for Resiliency and Security in Networks of Systems (ReSNetS)” call 2025, F83C26000350007.

ABSTRACT This paper introduces ResDyNet, a structured neural state-space model for nonlinear system identification from input–output data. The architecture interprets residual learning from a dynamical-systems perspective by decomposing the encoder, state-transition map, and decoder into an explicit linear component and a nonlinear correction. In the state-transition map, the linear component retains the form of a latent state-space model, while the nonlinear correction is represented through a deep cascade of identity-residual blocks operating in an internal residual stream. This separates the system-level linear–nonlinear decomposition from the depth-wise residual mechanism used to model nonlinear effects.

The architecture is initialized from an identified linear model and its finite-memory state reconstructor, while additional latent coordinates are introduced as trainable degrees of freedom. All parameters are then trained through a weighted multi-step prediction objective aligned with recursive rollout.

We establish universal approximation of the proposed evolver. Numerical experiments show that the structured parametrization improves prediction accuracy over a comparable architecture without linear components, while the linear-informed initialization provides a more favorable optimization trajectory. Results on standard nonlinear system-identification benchmarks further demonstrate competitive prediction performance.

INDEX TERMS Learning in Control, Nonlinear Systems, System Identification

I. INTRODUCTION

Identifying accurate dynamical systems from input–output data is a central problem in system identification, control, forecasting, and monitoring [1], [2]. In many engineering applications, first-principles models are either unavailable, too costly to derive, or too simplified to capture the relevant nonlinear behavior. Data-driven identification methods address this issue by estimating predictive models directly from measured trajectories, with the final goal of supporting simulations, diagnosis, control design, and decision making.

Classical system identification has developed a broad set of parametric and nonparametric tools. Linear structures such as ARX, ARMAX, output-error, and state-space models remain widely used because they are easy to interpret, computationally efficient, and supported by a mature theory [1], [2]. Nonlinear extensions include block-oriented models, nonlinear autoregressive models, kernel-based methods, and Gaussian-process formulations [3]. These approaches provide strong modeling tools, but their practical performance often depends on the choice of regressors, basis functions, kernels, or model structure. Moreover, scalability can become challenging when the dataset is large, the input dimension is high, or long-horizon prediction is required. From a modeling perspective, an additional question is how to retain an explicit representation of linear dynamical behavior while introducing sufficient flexibility to capture the nonlinear effects present in the data.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

A. RELATED WORKS

Neural networks provide flexible function approximators for nonlinear system identification and have been used to parametrize state-transition maps, output maps, recurrent models, and encoder-based state estimators [4], [5], [6], [7], [8], [9]. Neural state-space models are particularly attractive because they preserve a dynamical-system representation suitable for simulation and control. Their identification from input–output data, however, remains challenging because the latent state must be inferred from measured histories and propagated consistently over time, while controlling model complexity, overfitting, and optimization difficulties [10], [11], [12].

Autoencoder-based approaches address the unmeasured-state problem by jointly learning a latent representation and nonlinear state-space maps. Masti and Bemporad [13], for instance, construct the latent state through an encoder acting on past input–output data and jointly learn a decoder and a nonlinear state-update map. Their training criterion combines reconstruction, state-consistency, and prediction objectives, and both fully connected and quasi-LPV model parametrizations are considered.

A different approach is followed by the subspace-encoder framework of Beintema et al. [14]. There, a nonlinear encoder approximates a reconstructability map and initializes short overlapping prediction subsequences, while the encoder, state-transition map, and output map are jointly estimated through a truncated prediction loss. Linear-informed initialization of neural state-space models has also been investigated in [15]. In that approach, the state-space matrices and the encoder of SUBNET are initialized directly from a Best Linear Approximation and its reconstructability map, so that the initial nonlinear model approximately reproduces the identified linear model.

The initialization proposed here differs structurally from this approach. Rather than requiring the identified linear model to directly parameterize the complete latent dynamics, we allow its order $\tilde{n}$ to be smaller than the latent dimension n_x and embed it into a designated subspace of the ResDyNet latent representation. The remaining latent coordinates and their couplings are initialized at zero and constitute additional trainable degrees of freedom during nonlinear training.

ResDyNet builds on these ideas but adopts a different structured parametrization. The explicit linear components are separated from the nonlinear corrections throughout the encoder, evolver, and decoder. Moreover, the nonlinear part of the evolver is implemented through a deep identity-residual stream connected to the state–input and latent spaces through dedicated input and output maps. Finally, the auxiliary linear model is embedded into a potentially higher-dimensional latent model, so that its order need not coincide with the latent dimension of ResDyNet. The decoder is directly conditioned on both the latent state and the current input, allowing direct input-dependent output maps.

The main contributions of this work are as follows:

Approach	State transition	Linear initialization / latent order
Masti, Bemporad [13]	Feedforward NN; quasi-LPV variant also considered.	No structured linear-informed initialization.
Beintema et al. [14]	Feedforward NN, possibly with a linear bypass.	No linear-informed initialization in the original SUBNET formulation.
Ramkannan et al. [15]	Explicit linear dynamics in parallel with a fully connected nonlinear NN.	BLA and reconstructability map initialize the corresponding model components; linear and nonlinear models use the same state dimension.
ResDyNet	Explicit linear dynamics plus a nonlinear correction implemented by a deep residual cascade.	A linear model of order $\tilde{n} \leq n_x$ is embedded into the latent model; the remaining coordinates and couplings provide additional trainable degrees of freedom.

TABLE 1. Structural comparison of ResDyNet with related neural state-space identification approaches.

- We introduce a structured neural state-space architecture in which the encoder, evolver, and decoder combine explicit linear mappings with nonlinear corrections. In the evolver, the system-level linear state-space component is separated from a deep nonlinear correction implemented through an internal residual stream.
- We introduce a structured linear-informed initialization that decouples the order $\tilde{n}$ of the identified linear realization from the latent dimension n_x of the neural model. For $\tilde{n} \leq n_x$, the identified linear dynamics and their finite-memory state reconstructor are embedded into a designated latent subspace, while the complementary latent coordinates and their couplings are initialized as inactive degrees of freedom and subsequently learned during the nonlinear training phase.
- We establish universal approximation of the proposed evolver
- We experimentally assess the role of the structured parametrization through a parameter-matched comparison with the same architecture without linear components, together with sensitivity analyses and standard nonlinear identification benchmarks.

Structure of the paper – The remainder of this paper is organized as follows. Section II introduces the notation and formulates the identification problem. Section III presents the proposed architecture and training procedure. Section IV establishes the universal approximation capability of the evolver. Section V reports the numerical experiments and the resulting hyperparameter-selection guidelines. Section VI evaluates the architecture's performance using benchmark datasets. Finally, Section VII concludes the paper.

II. NOTATION AND PROBLEM FORMULATION

A. NOTATION

The sets of real and integer numbers are denoted by $\mathbb{R}$ and $\mathbb{Z}$, respectively, while $\mathbb{R}_{\geq 0}$, $\mathbb{N}$, $\mathbb{R}_+$, and $\mathbb{N}_+$ denote their nonnegative and positive subsets. Matrices are denoted by

uppercase letters, vectors by bold lowercase letters, scalars by lowercase letters, and sets by calligraphic letters.

Given a vector signal $\boldsymbol{s}_k \in \mathbb{R}^n$, where $k \in \mathbb{N}$ denotes a generic discrete-time instant, and an integer $\delta \in \mathbb{N}_+$, we define

$$[\boldsymbol{s}]_k^\delta := \begin{bmatrix} \boldsymbol{s}_{k-\delta+1}^\top & \cdots & \boldsymbol{s}_k^\top \end{bmatrix}^\top \in \mathbb{R}^{n\delta}, \qquad k \geq \delta - 1. \quad (1)$$

Thus, $[\boldsymbol{s}]_k^\delta$ denotes the vector obtained by stacking the last δ samples of $\boldsymbol{s}$ up to time k.

B. DYNAMICAL SYSTEM IDENTIFICATION

We consider non-autonomous dynamical systems evolving in discrete-time $k \in \mathbb{N}$ within a Euclidean state space $\mathcal{X} \subseteq \mathbb{R}^{n_s}$ and denote the state of the system at a generic time k by $\boldsymbol{x}_k \in \mathcal{X}$. The system dynamics can be described by a state-space model as follows

$$\Sigma := \begin{cases} \boldsymbol{x}_{k+1} = f(\boldsymbol{x}_k, \boldsymbol{u}_k), \\ \boldsymbol{y}_k = g(\boldsymbol{x}_k, \boldsymbol{u}_k), \end{cases} \qquad \boldsymbol{x}_0 \in \mathcal{X}, \quad (2)$$

where $\boldsymbol{u}_k \in \mathbb{R}^{n_u}$ is the input, $\boldsymbol{y}_k \in \mathbb{R}^{n_y}$ is the output, $f : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_s}$ is the state-transition map, and $g : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is the output map.

The system identification problem consists in determining a state-space model $\hat{\Sigma}$ whose input-output behavior well approximates that of the system Σ over a finite prediction horizon $H \in \mathbb{N}_+$. The model $\hat{\Sigma}$ is parameterized by $\vartheta := (\vartheta^e, \vartheta^f, \vartheta^g)$, where ϑ^e, ϑ^f, and ϑ^g collect the trainable parameters of the encoder, state-transition map, and output map, respectively. The model dynamics is given by

$$\widehat{\Sigma}(\vartheta) := \begin{cases} \hat{\boldsymbol{x}}_{k+1} = \hat{f}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^f), \\ \hat{\boldsymbol{y}}_k = \hat{g}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^g), \\ \hat{\boldsymbol{x}}_i = e([\boldsymbol{y}]_{i-1}^{n_a}, [\boldsymbol{u}]_{i-1}^{n_b}; \vartheta^e). \end{cases} \quad (3)$$

where $i \geq \bar{n} := \max\{n_a, n_b\}$ is any time instant at which the prediction is initialized, so that the observation windows lie inside the available data. Here, $\hat{\boldsymbol{x}}_k \in \mathbb{R}^{n_x}$ denotes the latent state of the identified model $\widehat{\Sigma}(\vartheta)$, while $\hat{\boldsymbol{y}}_k \in \mathbb{R}^{n_y}$ denotes its predicted output. The input $\boldsymbol{u}_k$ is the same as that of the system (2) to be identified, n_a, $n_b \in \mathbb{N}_+$ are the lengths of observation windows where input/output data are collected to determine the initial state of the learned model through the map $e : \mathbb{R}^{n_a n_y} \times \mathbb{R}^{n_b n_u} \to \mathbb{R}^{n_x}$, $\hat{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ and $\hat{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ are the state-transition and output maps. This objective must be pursued under the following standing set of assumptions.

Assumption 1. *The system (2) is such that:*

- *The state $\boldsymbol{x}_k$ cannot be measured;*
- *Functions f and g are continuous.*

In this work, maps $\hat{f}$, $\hat{g}$, and e will be modeled as neural networks presented in the next subsection.

C. NEURAL NETWORKS

A *neural network* with $m \in \mathbb{N}_+$ scalar trainable parameters maps an input $\boldsymbol{h}^0 \in \mathbb{R}^{n_0}$ into an output $\boldsymbol{h} \in \mathbb{R}^{n_L}$ through a map $\phi : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$, where ϑ collects its trainable parameters,

$$\boldsymbol{h} = \phi(\boldsymbol{h}^0; \vartheta).$$

A *feedforward neural network* with $L \in \mathbb{N}_+$ layers maps an input $\boldsymbol{h}^0 \in \mathbb{R}^{n_0}$ through the recursion

$$\boldsymbol{h}^\ell := \rho^\ell\left(W^\ell \boldsymbol{h}^{\ell-1} + \boldsymbol{b}^\ell\right), \quad \ell \in \{1, \ldots, L\}, \quad (4)$$

where $W^\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ is the weight matrix, $\boldsymbol{b}^\ell \in \mathbb{R}^{n_\ell}$ is the bias vector, and $\rho^\ell : \mathbb{R}^{n_\ell} \to \mathbb{R}^{n_\ell}$ is the activation map of the ℓ-th layer. Let $\phi_{\text{FF}} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ denote the feedforward map and $\vartheta_{\text{FF}} := (W^\ell, \boldsymbol{b}^\ell)_{\ell=1}^L$ the ordered collection of its weight matrices and bias vectors, containing $m_{\text{FF}} = \sum_{\ell=1}^L n_\ell(1 + n_{\ell-1})$ scalar trainable parameters. The feedforward output $\boldsymbol{h}^L \in \mathbb{R}^{n_L}$ is then given by

$$\boldsymbol{h}^L = \phi_{\text{FF}}(\boldsymbol{h}^0; \vartheta_{\text{FF}}).$$

Feedforward neural networks can be equipped with a skip connection, which combines the feedforward output $\boldsymbol{h}^L$ with the input $\boldsymbol{h}^0$ to obtain the final output $\boldsymbol{h}$ of the neural network. In particular, this paper limits the attention to linear skip connections:

- *Residual neural network*. When $n_0 = n_L$, the input is added to the feedforward output via an identity skip connection, i.e., $\boldsymbol{h} = \boldsymbol{h}^L + \boldsymbol{h}^0$.
- *Explicit linear path residual neural network*. A matrix $S \in \mathbb{R}^{n_L \times n_0}$ linearly projects the block input before the skip addition into the output space, i.e., $\boldsymbol{h} = \boldsymbol{h}^L + S\boldsymbol{h}^0$.

Since throughout the paper only the three above kinds of neural networks are employed, the following common notation is used,

$$\boldsymbol{h} = \phi(\boldsymbol{h}^0; \vartheta) = \phi_{\text{FF}}(\boldsymbol{h}^0; \vartheta_{\text{FF}}) + S\boldsymbol{h}^0, \quad (5)$$

where $S \in \mathbb{R}^{n_L \times n_0}$ is the skip matrix. If S is trainable, $\vartheta := (S, \vartheta_{\text{FF}})$ and $m = m_{\text{FF}} + n_0 n_L$; otherwise, $\vartheta = \vartheta_{\text{FF}}$ and $m = m_{\text{FF}}$. With this notation, the neural network in (5) is: feedforward if $S = \nvDash_{n_L \times n_0}$; residual if $n_0 = n_L$ and $S = I_{n_0}$; projected residual otherwise.

D. PROBLEM FORMULATION

We collect $N \in \mathbb{N}_+$ input/output pairs of the system (2) subject to an excitation sequence $\{\boldsymbol{u}_k\}_{k=0}^{N-1}$ from an unknown initial condition $\boldsymbol{x}_0$, yielding the output sequence $\{\boldsymbol{y}_k\}_{k=0}^{N-1}$. The resulting dataset is

$$\mathcal{D} := \{(\boldsymbol{u}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{u}_{N-1}, \boldsymbol{y}_{N-1})\}. \quad (6)$$

Let $\bar{n} := \max\{n_a, n_b\}$, then the dataset must be sufficiently rich so that $N \gg H + \bar{n}$, i.e., there are sufficient consecutive samples in the dataset to allow computing the initial state of the model (3) to be learned from the input/output observation windows and also to compare the predicted output with the real output over the prediction horizon. In other words, let $i \in \mathbb{N}$ be any time at which the prediction is made, then it must hold that $i \geq \bar{n}$ to have sufficient data to initialize the state of the learned model but also $i \leq N - H$ to have sufficient data to evaluate the predicted output of the learned

model against the real output, i.e., $N-H \geq \bar{n}$. Accordingly, define the set of admissible initial prediction times as

$$\mathcal{I} := \{\bar{n}, \bar{n}+1, \dots, N-H\}, \tag{7}$$

and the set of future prediction indices as

$$\mathcal{H} := \{0, 1, 2, \dots, H-1\}. \tag{8}$$

We also define $\mathcal{H}_+ := \mathcal{H} \setminus \{0\}$, i.e., the set obtained from $\mathcal{H}$ by excluding 0. Let us now use the subscript "$j|i$" to denote the prediction at time i of the quantity at time j with $j \geq i$. The quality of a model $\hat{\Sigma}$ is measured with the mean-squared error (MSE)

$$\frac{1}{|\mathcal{I}||\mathcal{H}|} \sum_{i\in\mathcal{I}} \sum_{h\in\mathcal{H}} \left\| \boldsymbol{y}_{i+h} - \hat{\boldsymbol{y}}_{i+h|i} \right\|_2^2, \tag{9}$$

which measures the mismatch between any output prediction $\hat{\boldsymbol{y}}_{i+h|i}$ and the real output $\boldsymbol{y}_{i+h}$, namely

$$\begin{aligned}
\hat{\boldsymbol{x}}_{i|i} &= e([\boldsymbol{y}]_{i-1}^{n_a}, [\boldsymbol{u}]_{i-1}^{n_b}; \vartheta^e), && \forall i \in \mathcal{I}, \\
\hat{\boldsymbol{x}}_{i+h|i} &= \hat{f}\left(\hat{\boldsymbol{x}}_{i+h-1|i}, \boldsymbol{u}_{i+h-1}; \vartheta^f\right), && \forall i \in \mathcal{I}, \forall h \in \mathcal{H}_+, \\
\hat{\boldsymbol{y}}_{i+h|i} &= \hat{g}\left(\hat{\boldsymbol{x}}_{i+h|i}, \boldsymbol{u}_{i+h}; \vartheta^g\right), && \forall i \in \mathcal{I}, \forall h \in \mathcal{H}.
\end{aligned}$$

Thus, the relevant objective is not merely to obtain accurate local input–output mappings, but to preserve prediction accuracy when the learned latent state is propagated recursively over multiple steps. Errors in the initial latent-state estimate and in the transition map are repeatedly fed back into the model and may therefore accumulate along the rollout.

The resulting design problem is to obtain a state-space parametrization that is sufficiently expressive to capture nonlinear dynamics while retaining a structure that can be effectively initialized and optimized under recursive multi-step training. In this work, we address this problem by separating explicit linear mappings from nonlinear corrections and by using residual learning inside the nonlinear state-transition model. This construction provides the basis for the architecture and the linear-informed training procedure introduced in the next section.

III. PROPOSED NN-BASED DYNAMIC MODEL AND TRAINING PROCEDURE

We address the identification objective described in Section D by introducing a NN-based dynamical model with the following structure

$$\widehat{\Sigma}(\vartheta) := \begin{cases} \hat{\boldsymbol{x}}_{k+1} = \hat{f}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^f), \\ \hat{\boldsymbol{y}}_k = \hat{g}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^g), \\ \hat{\boldsymbol{x}}_i = e([\boldsymbol{y}]_{i-1}^{n_a}, [\boldsymbol{u}]_{i-1}^{n_b}; \vartheta^e). \end{cases}$$

Since the true state $\boldsymbol{x}_k$ in (2) of the system to be identified cannot be measured, the latent state $\hat{\boldsymbol{x}}_k$ is initialized from finite input-output histories $[\boldsymbol{y}]_{i-1}^{n_a}, [\boldsymbol{u}]_{i-1}^{n_b}$ through a mapping $e : \mathbb{R}^{n_a n_y} \times \mathbb{R}^{n_b n_u} \to \mathbb{R}^{n_x}$, called the *encoder*. We model the encoder as a neural network:

$$e([\boldsymbol{y}]_{k-1}^{n_a}, [\boldsymbol{u}]_{k-1}^{n_b}; \vartheta^e) := \phi^e(\boldsymbol{r}_k; \vartheta^e) \quad \text{with} \quad \boldsymbol{r}_k = \begin{bmatrix} [\boldsymbol{y}]_{k-1}^{n_a} \\ [\boldsymbol{u}]_{k-1}^{n_b} \end{bmatrix} \tag{10}$$

where $\boldsymbol{r}_k$ is the regression vector at time k of dimension $n_r := n_a n_y + n_b n_u$. The encoder has a projected residual design with trainable matrix $S_e \in \mathbb{R}^{n_x \times n_r}$ and parameters $\vartheta^e := (S_e, \vartheta^e_{\mathrm{FF}})$, where $\vartheta^e_{\mathrm{FF}}$ collects the weight matrices and bias vectors of its feedforward branch, namely,

$$\phi^e(\boldsymbol{r}_k; \vartheta^e) := \phi^e_{\mathrm{FF}}(\boldsymbol{r}_k; \vartheta^e_{\mathrm{FF}}) + S_e \boldsymbol{r}_k.$$

Remark 1. The term $S_e \boldsymbol{r}_k$ provides an explicit linear finite-memory state estimator, while $\phi^e_{\mathrm{FF}}(\boldsymbol{r}_k; \vartheta^e_{\mathrm{FF}})$ provides an additional nonlinear correction. The projection is necessary because the regressor dimension $n_a n_y + n_b n_u$ generally differs from the latent dimension n_x; hence, an identity shortcut would not be admissible in general. ⋄

The predicted output $\hat{\boldsymbol{y}}_k$ is instead obtained through an output mapping $\hat{g} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$, called the *decoder*. We model the decoder as a neural network:

$$\hat{g}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^g) := \phi^g(\boldsymbol{h}_k; \vartheta^g) \quad \text{with} \quad \boldsymbol{h}_k = \begin{bmatrix} \hat{\boldsymbol{x}}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{11}$$

The decoder has a projected residual design with trainable matrix $S_g \in \mathbb{R}^{n_y \times (n_x + n_u)}$ and parameters $\vartheta^g := (S_g, \vartheta^g_{\mathrm{FF}})$, where $\vartheta^g_{\mathrm{FF}}$ collects the weight matrices and bias vectors of its feedforward branch, namely,

$$\phi^g(\boldsymbol{h}_k; \vartheta^g) := \phi^g_{\mathrm{FF}}(\boldsymbol{h}_k; \vartheta^g_{\mathrm{FF}}) + S_g \boldsymbol{h}_k.$$

Remark 2. The proposed decoder inherits the explicit linear path structure already defined for the encoder network. It also differs from output maps employed in [13], [14] since it is explicitly conditioned on both the latent state $\hat{\boldsymbol{x}}_k$ and the current input $\boldsymbol{u}_k$. ⋄

The latent state is propagated forward in time by the map $\hat{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$, called the *evolver*. We model the evolver as a neural network:

$$\hat{f}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k; \vartheta^f) := \phi^f(\boldsymbol{h}_k; \vartheta^f) \quad \text{with} \quad \boldsymbol{h}_k = \begin{bmatrix} \hat{\boldsymbol{x}}_k \\ \boldsymbol{u}_k \end{bmatrix}. \tag{12}$$

Let $q := n_x + n_u$ denote the dimension of $\boldsymbol{h}_k$. Since the evolver maps $\mathbb{R}^q$ into $\mathbb{R}^{n_x}$ and is therefore non-square, its nonlinear branch is organized as a cascade of $B \in \mathbb{N}_+$ identity-residual blocks acting on a *residual stream* of dimension $n_h \geq q$, accessed through a linear input map $W_i \in \mathbb{R}^{n_h \times q}$ and a linear output map $W_o \in \mathbb{R}^{n_x \times n_h}$. The design is depicted in Fig. 1, with trainable matrix $S_f \in \mathbb{R}^{n_x \times q}$ and parameters $\vartheta^f := (S_f, W_i, W_o, \vartheta^{f,1}_{\mathrm{FF}}, \dots, \vartheta^{f,B}_{\mathrm{FF}})$, where $\vartheta^{f,b}_{\mathrm{FF}}$ collects the weights and biases of block b, namely

$$\begin{aligned}
\phi^f(\boldsymbol{h}_k; \vartheta^f) &:= S_f \boldsymbol{h}_k + \Xi(\boldsymbol{h}_k; \vartheta^f), \\
\Xi(\boldsymbol{h}_k; \vartheta^f) &:= W_o \left(\boldsymbol{h}_k^B - \boldsymbol{h}_k^0 \right), \\
\boldsymbol{h}_k^b &:= \boldsymbol{h}_k^{b-1} + \phi^{f,b}_{\mathrm{FF}}(\boldsymbol{h}_k^{b-1}; \vartheta^{f,b}_{\mathrm{FF}}), \quad b \in \{1, \dots, B\}, \\
\boldsymbol{h}_k^0 &:= W_i \boldsymbol{h}_k,
\end{aligned} \tag{13}$$

where each feedforward branch $\phi^{f,b}_{\mathrm{FF}} : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}$ is square, so that the identity shortcuts inside the cascade are well defined, whereas the linear component $S_f \in \mathbb{R}^{n_x \times q}$ is

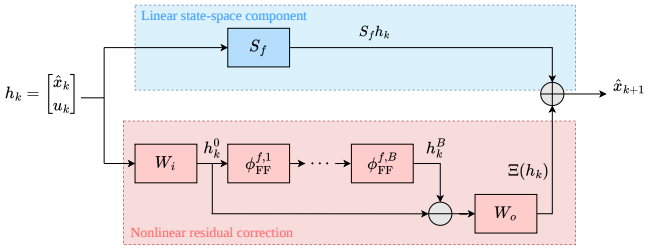


FIGURE 1. **Evolver $\hat{f}$ of Fig. 2. The input $h_k = [\hat{x}_k^\top \; u_k^\top]^\top$ is mapped by W_i into a residual stream of dimension n_h, processed by a cascade of B identity-residual blocks, and mapped to the latent state space by W_o. The linear backbone $S_f h_k$ is then added to obtain the next latent state $\hat{x}_{k+1}$.**

projected. For later use, let $T_b(\boldsymbol{h}) := \boldsymbol{h} + \phi_{\mathrm{FF}}^{f,b}(\boldsymbol{h}; \vartheta_{\mathrm{FF}}^{f,b})$ denote the map of the b-th block and define the residual cascade

$$\Phi_B := T_B \circ T_{B-1} \circ \cdots \circ T_1 : \mathbb{R}^{n_h} \to \mathbb{R}^{n_h}, \tag{14}$$

so that $\Phi_B(\boldsymbol{h}_k^0) = \boldsymbol{h}_k^B$ and the nonlinear correction reads $\Xi(\boldsymbol{h}_k; \vartheta^f) = W_o\big(\Phi_B(W_i \boldsymbol{h}_k) - W_i \boldsymbol{h}_k\big)$. Unfolding the cascade gives

$$\Phi_B(\boldsymbol{h}_k^0) - \boldsymbol{h}_k^0 = \sum_{b=1}^{B} \phi_{\mathrm{FF}}^{f,b}(\boldsymbol{h}_k^{b-1}; \vartheta_{\mathrm{FF}}^{f,b}), \tag{15}$$

i.e., the correction is the sum of the individual block corrections.

Remark 3. The proposed evolver contains two residual mechanisms operating in different spaces. The identity shortcuts act within the square residual stream $\mathbb{R}^{n_h}$, whereas the external projected term $S_f \boldsymbol{h}_k$ acts from $\mathbb{R}^q$ to $\mathbb{R}^{n_x}$ and can be written as an explicit linear state-space model in the latent coordinates,

$$S_f \boldsymbol{h}_k = \begin{bmatrix} A_f & B_f \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{x}}_k \\ \boldsymbol{u}_k \end{bmatrix}.$$

The nonlinear branch provides an additional correction to this linear contribution and is represented internally by a cascade of identity-residual blocks. The two mechanisms therefore play different roles: S_f exposes a linear latent state-transition structure that can be initialized from linear system identification, whereas the internal identity shortcuts provide direct paths across the depth of the nonlinear transition network and facilitate the optimization of deeper evolvers. $\diamond$

All feedforward branches ϕ_{FF}^{e}, ϕ_{FF}^{g}, $\phi_{\mathrm{FF}}^{f,1}, \ldots, \phi_{\mathrm{FF}}^{f,B}$ are composed of $L \geq 2$ layers, where L may differ across branches. The first $L-1$ layers are hidden layers and are followed by nonlinear activation functions ρ^ℓ, $\ell = 1, \ldots, L-1$. The last layers are the output layers and are taken to be linear, i.e., no nonlinear activation is applied after them, so that ρ^L is the identity map.

A. PROPOSED TRAINING PROCEDURE

This subsection describes the training procedure adopted to identify the parameters ϑ of the proposed NN-based dynamical model. The procedure is composed of two stages. First, a linear state-space approximation and an associated finite-memory state reconstructor are identified from the training data and embedded into the explicit linear components of ResDyNet. Second, starting from this linear-informed initialization, all linear and nonlinear parameters are trained by minimizing a weighted multi-step prediction loss.

1) LINEAR-INFORMED INITIALIZATION

Using only the training portion of the dataset $\mathcal{D}$, we first identify a linear state-space model

$$\begin{aligned} \tilde{\boldsymbol{x}}_{k+1} &= \tilde{A}\tilde{\boldsymbol{x}}_k + \tilde{B}\boldsymbol{u}_k, \\ \tilde{\boldsymbol{y}}_k &= \tilde{C}\tilde{\boldsymbol{x}}_k + \tilde{D}\boldsymbol{u}_k, \end{aligned} \tag{16}$$

of order $\tilde{n} \leq n_x$, where $\tilde{A} \in \mathbb{R}^{\tilde{n} \times \tilde{n}}$, $\tilde{B} \in \mathbb{R}^{\tilde{n} \times n_u}$, $\tilde{C} \in \mathbb{R}^{n_y \times \tilde{n}}$, and $\tilde{D} \in \mathbb{R}^{n_y \times n_u}$. The proposed framework does not prescribe a specific linear identification algorithm, provided that it returns a state-space realization; this stage can therefore be carried out using a standard linear approximation and state-space identification procedure.

The order $\tilde{n}$ is kept distinct from the latent dimension n_x, since the latter is a design parameter of the neural model. Choosing a linear realization of unnecessarily high order may introduce weakly observable or redundant modes, whereas the proposed construction allows a lower-order linear model to be embedded into a larger latent space.

Assumption 2 (Linear identification stage)**.** *The pair $(\tilde{C}, \tilde{A})$ of the realization in eq. (16) is observable. The observation windows satisfy $n_a = n_b =: n \geq \nu$, where $\nu \leq \tilde{n}$ denotes the observability index of $(\tilde{C}, \tilde{A})$.*

To initialize the encoder consistently with (16), we construct a finite-memory reconstructability map following standard linear systems theory. Define the finite-horizon observability matrix

$$\mathcal{O} := \begin{bmatrix} \tilde{C}^\top & (\tilde{C}\tilde{A})^\top & \cdots & (\tilde{C}\tilde{A}^{n-1})^\top \end{bmatrix}^\top \in \mathbb{R}^{nn_y \times \tilde{n}}. \tag{17}$$

Under Assumption 2 one has $\mathrm{rank}(\mathcal{O}) = \tilde{n}$, which requires in particular $nn_y \geq \tilde{n}$, and the Moore–Penrose pseudoinverse $\mathcal{O}^\dagger = (\mathcal{O}^\top \mathcal{O})^{-1} \mathcal{O}^\top$ is a left inverse of $\mathcal{O}$.

For the linear model (16), the stacked outputs over the interval $k-n, \ldots, k-1$ satisfy

$$[\tilde{\boldsymbol{y}}]_{k-1}^{n} = \mathcal{O}\tilde{\boldsymbol{x}}_{k-n} + \mathcal{T}[\boldsymbol{u}]_{k-1}^{n}, \tag{18}$$

where

$$\mathcal{T} := \begin{bmatrix} \tilde{D} & 0 & \cdots & 0 \\ \tilde{C}\tilde{B} & \tilde{D} & \ddots & \vdots \\ \tilde{C}\tilde{A}\tilde{B} & \tilde{C}\tilde{B} & \ddots & 0 \\ \vdots & \vdots & \ddots & \tilde{D} \\ \tilde{C}\tilde{A}^{n-2}\tilde{B} & \tilde{C}\tilde{A}^{n-3}\tilde{B} & \cdots & \tilde{D} \end{bmatrix} \in \mathbb{R}^{nn_y \times nn_u}. \tag{19}$$

Moreover, propagating the state over the same interval gives

$$\tilde{\boldsymbol{x}}_k = \tilde{A}^n \tilde{\boldsymbol{x}}_{k-n} + \mathcal{C}[\boldsymbol{u}]_{k-1}^{n}, \tag{20}$$

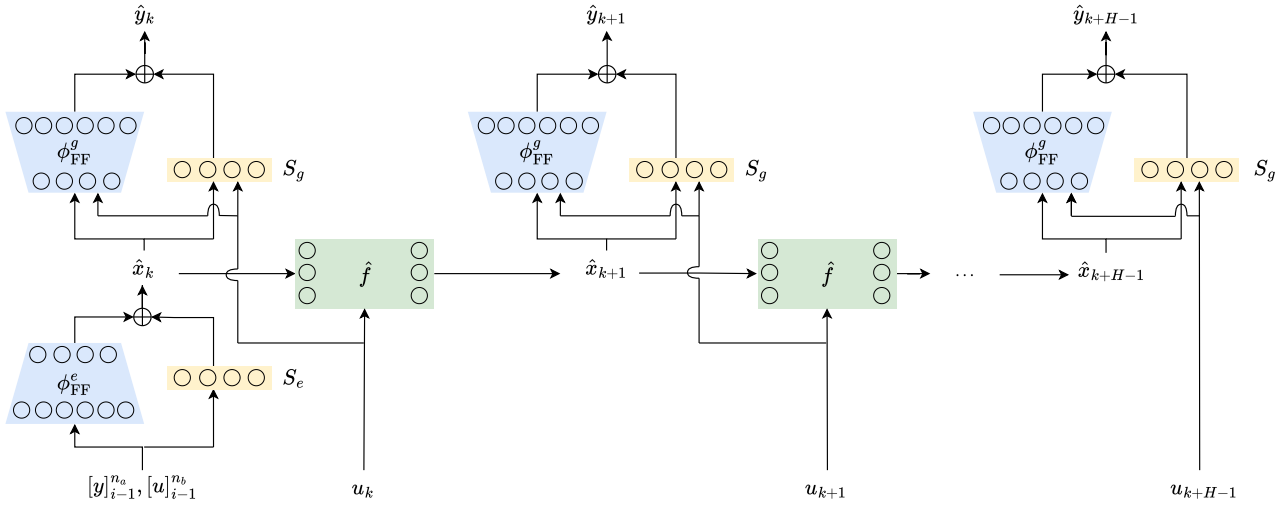


FIGURE 2. **Schematic representation of ResDyNet. Yellow blocks represent the linear component while the blue blocks are the nonlinear correction. At the first time step, the state $\hat{x}_k$ is initialized by the encoder, and then the predictions are rolled out by the evolver and the decoder, without updating through the encoder.**

with $\mathcal{C} := \begin{bmatrix}\tilde{A}^{n-1}\tilde{B} & \cdots & \tilde{B}\end{bmatrix} \in \mathbb{R}^{\tilde{n}\times n n_u}$. Recovering $\tilde{\boldsymbol{x}}_{k-n}$ from (18) through $\mathcal{O}^\dagger$ and substituting into (20) yields

$$\tilde{\boldsymbol{x}}_k = \tilde{A}^n\mathcal{O}^\dagger[\tilde{\boldsymbol{y}}]^n_{k-1} + \left(\mathcal{C} - \tilde{A}^n\mathcal{O}^\dagger\mathcal{T}\right)[\boldsymbol{u}]^n_{k-1}. \tag{21}$$

Relation (21) involves the outputs $\tilde{\boldsymbol{y}}$ generated by (16), which do not coincide with the measured outputs $\boldsymbol{y}$ collected in the regressor $\boldsymbol{r}_k$ of (10). Replacing $[\tilde{\boldsymbol{y}}]^n_{k-1}$ by the measured window $[\boldsymbol{y}]^n_{k-1}$ therefore does not reproduce the state of (16), but defines a finite-memory *estimate* of it,

$$\tilde{\boldsymbol{x}}_k := R\boldsymbol{r}_k, \quad R := \begin{bmatrix}\tilde{A}^n\mathcal{O}^\dagger & \mathcal{C} - \tilde{A}^n\mathcal{O}^\dagger\mathcal{T}\end{bmatrix}, \tag{22}$$

where $R \in \mathbb{R}^{\tilde{n}\times n_r}$ with $n_r = n(n_y + n_u)$. Equation (22) is thus a deadbeat finite-memory reconstructor rather than a minimum-variance estimator; it is used here only as an initialization, and all its entries are subsequently trained.

The linear-informed initialization of ResDyNet is then defined by

$$S_e := \left[\begin{array}{c} R \\ \hline \mathbf{0} \end{array}\right], \tag{23}$$
$$S_f := \left[\begin{array}{c|c|c} \tilde{A} & \mathbf{0} & \tilde{B} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array}\right],$$
$$S_g := \left[\begin{array}{c|c|c} \tilde{C} & \mathbf{0} & \tilde{D} \end{array}\right],$$

where $\mathbf{0}$ denotes zero blocks of conformable dimensions and the block partitions are conformable with $n_x = \tilde{n} + (n_x - \tilde{n})$ and with $\boldsymbol{h}_k = [\hat{\boldsymbol{x}}_k^\top \ \boldsymbol{u}_k^\top]^\top$, so that the $n_x - \tilde{n}$ additional latent coordinates are initially inactive and are populated only during the training phase. The initialization is completed by zeroing the output layers of all nonlinear branches: denoting by $(W^{\bullet,L}, \boldsymbol{b}^{\bullet,L})$ the weights and biases of the L-th (linear) layer of the branch $\phi^\bullet_{\mathrm{FF}}$, we set

$$\begin{aligned} W^{e,L} &= 0, \quad & \boldsymbol{b}^{e,L} &= 0, \\ W^{g,L} &= 0, \quad & \boldsymbol{b}^{g,L} &= 0, \\ W^{f,b,L} &= 0, \quad & \boldsymbol{b}^{f,b,L} &= 0, \quad \forall b \in \{1,\dots,B\}, \end{aligned} \tag{24}$$

while all remaining parameters, including W_i and W_o, are initialized randomly. By (24) one has $\phi^e_{\mathrm{FF}} \equiv 0$ and $\phi^g_{\mathrm{FF}} \equiv 0$, and $\Xi \equiv 0$.

2) MULTI-STEP TRAINING

Starting from (23)–(24), all parameters of the encoder, evolver, and decoder are jointly trained, including S_e, S_f, and S_g. Hence, the linear identification stage provides an initialization and does not constrain the final linear components of the learned model.

Let $\boldsymbol{\gamma} := [\gamma_0 \ \cdots \ \gamma_{H-1}]^\top \in \mathbb{R}^{|\mathcal{H}|}_{\geq 0}$ be a vector of nonnegative prediction weights, where γ_h weights the contribution of the prediction error at the h-th step of the rollout. Uniform weights recover the standard multi-step MSE, whereas nonuniform weights can be used to emphasize different portions of the prediction horizon. In the implementation, the weights may be normalized as

$$\frac{1}{|\mathcal{H}|}\sum_{h\in\mathcal{H}}\gamma_h = 1, \tag{25}$$

so that the scale of the training objective remains comparable for different values of H.

As depicted in Fig. 2, training is performed by considering finite rollout windows extracted from the measured input–output trajectory. For each admissible initial prediction time $i \in \mathcal{I}$, the encoder is used only once to initialize the latent state from the available past measurements. The future latent states are then obtained by recursively applying the evolver

$\hat{f}$, and the corresponding output predictions are computed through the decoder $\hat{g}$. Thus, the encoder is not repeatedly used along the prediction horizon. Instead, the model is trained so that the initialized latent state can be propagated recursively without further access to measured outputs.

The training phase is formulated as

$$\begin{aligned}\min_{\vartheta^e,\vartheta^f,\vartheta^g}\quad & \frac{1}{|\mathcal{I}||\mathcal{H}|}\sum_{i\in\mathcal{I}}\sum_{h\in\mathcal{H}}\gamma_h\left\|\boldsymbol{y}_{i+h}-\hat{\boldsymbol{y}}_{i+h|i}\right\|_2^2, && (26)\\ \text{s.t.}\quad & \hat{\boldsymbol{x}}_{i|i}=e\left([\boldsymbol{y}]_{i-1}^{n_a},[\boldsymbol{u}]_{i-1}^{n_b};\vartheta^e\right), && \forall i\in\mathcal{I},\\ & \hat{\boldsymbol{x}}_{i+h|i}=\hat{f}\left(\hat{\boldsymbol{x}}_{i+h-1|i},\boldsymbol{u}_{i+h-1};\vartheta^f\right), && \forall i\in\mathcal{I},\forall h\in\mathcal{H}_+,\\ & \hat{\boldsymbol{y}}_{i+h|i}=\hat{g}\left(\hat{\boldsymbol{x}}_{i+h|i},\boldsymbol{u}_{i+h};\vartheta^g\right), && \forall i\in\mathcal{I},\forall h\in\mathcal{H}.\end{aligned}$$

Note that standard linear identification methods rely on criteria, such as one-step prediction error, output error, or subspace-based fitting, that generally differ from the multi-step objective (26); the linear stage is therefore used as a consistent starting point and not as a partial solution of (26). The effect of this initialization on the optimization trajectory is assessed in Section V by comparison with random initialization under the same architecture and training budget.

IV. UNIVERSAL APPROXIMATION OF THE EVOLVER

Universal approximation is not the main contribution of this work. The purpose of this section is to verify that the explicit linear branch $S_f\boldsymbol{h}$ in (13) does not restrict the class of maps representable by the evolver. The argument relies on the control-theoretic approximation result for deep residual networks of [16] and on its correction [17], which raises the sufficient width from $q+1$ to $2q+1$.

Consistently with Section C, let $\varrho:\mathbb{R}\to\mathbb{R}$ be a scalar activation function, let $\rho(\boldsymbol{\zeta}):=[\varrho(\zeta_1)\ \cdots\ \varrho(\zeta_m)]^\top$ for $\boldsymbol{\zeta}\in\mathbb{R}^m$, and let $\|F\|_{L^\infty(\mathcal{K})}:=\sup_{\boldsymbol{p}\in\mathcal{K}}\|F(\boldsymbol{p})\|_\infty$ for a map F and a compact set $\mathcal{K}$.

Definition 1 (Activation class [16, Def. 3.1])**.** *The function ϱ belongs to $\mathcal{A}_{\text{quad}}$ if: (i) there exist $j\in\mathbb{N}$ and $a_0,a_1,a_2\in\mathbb{R}$, $a_2\neq 0$, such that $\xi:=D^j\varrho$ satisfies $D\xi=a_0+a_1\xi+a_2\xi^2$; (ii) ϱ is Lipschitz continuous; (iii) $D\varrho\geq 0$; (iv) ξ is injective.*

The result we invoke is [17, Cor. 3.2]: for $q>1$, $\varrho\in\mathcal{A}_{\text{quad}}$, every continuous $\psi:\mathbb{R}^q\to\mathbb{R}^q$, every compact $\mathcal{K}\subset\mathbb{R}^q$ and every $\varepsilon>0$, there exist $\tau>0$, an injection $\alpha:\mathbb{R}^q\to\mathbb{R}^{2q+1}$, a projection $\beta:\mathbb{R}^{2q+1}\to\mathbb{R}^q$ and a piecewise-constant input $(s,W,\boldsymbol{b}):[0,\tau]\to\mathbb{R}\times\mathbb{R}^{(2q+1)\times(2q+1)}\times\mathbb{R}^{2q+1}$ such that the time-τ flow φ^τ of

$$\dot{\boldsymbol{\zeta}}(t)=s(t)\,\rho(W(t)\boldsymbol{\zeta}(t)+\boldsymbol{b}(t)) \tag{27}$$

satisfies $\|\psi-\beta\circ\varphi^\tau\circ\alpha\|_{L^\infty(\mathcal{K})}\leq\varepsilon$. Since the architecture (13) accesses the residual stream through the matrices W_i and W_o, we require the lifting maps to be linear.

The explicit construction in the proof of [16, Cor. 5.2], as corrected in [17, Cor. 3.2], yields linear lifting maps: for $\boldsymbol{p}\in\mathbb{R}^q$ and $(\boldsymbol{x},\boldsymbol{y},z)\in\mathbb{R}^q\times\mathbb{R}^q\times\mathbb{R}$,

$$\alpha(\boldsymbol{p})=(\boldsymbol{p},\boldsymbol{p},\mathbf{1}^\top\boldsymbol{p}),\qquad \beta(\boldsymbol{x},\boldsymbol{y},z)=\boldsymbol{x}-\kappa\mathbf{1}z, \tag{28}$$

where $\kappa\geq 0$ is a constant determined by the construction. Both maps are linear, α is injective and β is surjective, so they can be represented by the linear input and output maps of the proposed evolver.

Theorem 1 (Universal approximation of the evolver)**.** *Let $\varrho\in\mathcal{A}_{\text{quad}}$, let $q:=n_x+n_u>1$, and let $\mathcal{K}\subset\mathbb{R}^q$ be compact. Then, for every continuous $f^\star:\mathbb{R}^q\to\mathbb{R}^{n_x}$ and every $\varepsilon>0$, there exist a residual-stream dimension $n_h=2q+1$, a depth $B\in\mathbb{N}_+$, matrices $S_f\in\mathbb{R}^{n_x\times q}$, $W_i\in\mathbb{R}^{n_h\times q}$, $W_o\in\mathbb{R}^{n_x\times n_h}$ and block parameters $\vartheta_{\text{FF}}^{f,1},\dots,\vartheta_{\text{FF}}^{f,B}$ such that the evolver (13) satisfies*

$$\left\|f^\star-\phi^f\right\|_{L^\infty(\mathcal{K})}\leq\varepsilon. \tag{29}$$

Each residual block can be chosen with one ϱ-activated hidden layer of width n_h and one linear output layer.

Proof:
Let $\iota:\mathbb{R}^{n_x}\to\mathbb{R}^q$, $\iota(\boldsymbol{p}):=[\boldsymbol{p}^\top\ \mathbb{1}_{n_u}^\top]^\top$, let $\pi_{n_x}:\mathbb{R}^q\to\mathbb{R}^{n_x}$ be the projection onto the first n_x coordinates, and set $\psi:=\iota\circ f^\star:\mathbb{R}^q\to\mathbb{R}^q$, which is continuous and square. Applying [17, Cor. 3.2] to ψ with accuracy $\varepsilon/2$ yields τ, α, β and $(s,W,\boldsymbol{b})$ with

$$\left\|\psi-\beta\circ\varphi^\tau\circ\alpha\right\|_{L^\infty(\mathcal{K})}\leq\varepsilon/2. \tag{30}$$

Set $n_h:=2q+1$ and, since α and β in (28) are linear,

$$W_i:=\alpha,\qquad W_o:=\pi_{n_x}\circ\beta,\qquad S_f:=W_oW_i, \tag{31}$$

which have the dimensions required in (13).

By [16, Sec. II-A], the flow of (27) is approximated by the explicit Euler recursion

$$\boldsymbol{\zeta}^b=\boldsymbol{\zeta}^{b-1}+T\,s(t_{b-1})\,\rho\Big(W(t_{b-1})\boldsymbol{\zeta}^{b-1}+\boldsymbol{b}(t_{b-1})\Big), \tag{32}$$

with $T:=\tau/B$ and $t_b:=bT$. Each step (32) is a residual block of (13): taking $\phi_{\text{FF}}^{f,b}(\boldsymbol{\zeta})=W_2^b\rho(W_1^b\boldsymbol{\zeta}+\boldsymbol{b}_1^b)$ with one hidden layer and a linear output layer with zero bias, and setting $W_1^b=W(t_{b-1})$, $\boldsymbol{b}_1^b=\boldsymbol{b}(t_{b-1})$, $W_2^b=Ts(t_{b-1})I_{n_h}$, the block update coincides with (32). Moreover, ρ is Lipschitz and $(s,W,\boldsymbol{b})$ is bounded, so the solutions of (27) issuing from the compact set $\alpha(\mathcal{K})$ remain, for $t\in[0,\tau]$, in a compact set on which the vector field is Lipschitz. By the forward-Euler approximation argument of [16, Sec. II-A], the approximation error is uniform on $\alpha(\mathcal{K})$ and vanishes as $T=\tau/B\to 0$. Hence B can be chosen so that

$$\sup_{\boldsymbol{\zeta}\in\alpha(\mathcal{K})}\left\|\varphi^\tau(\boldsymbol{\zeta})-\Phi_B(\boldsymbol{\zeta})\right\|_\infty\leq\frac{\varepsilon}{2\max\{1,\|\beta\|_\infty\}}, \tag{33}$$

where $\|\beta\|_\infty$ is the induced infinity norm of β. Combining (30) and (33),

$$\left\|\psi-\beta\circ\Phi_B\circ\alpha\right\|_{L^\infty(\mathcal{K})}\leq\varepsilon. \tag{34}$$

Finally, by (31) the linear branch cancels the shift of the residual cascade exactly, since for every $\boldsymbol{h}\in\mathbb{R}^q$

$$\begin{aligned}\phi^f(\boldsymbol{h})&=S_f\boldsymbol{h}+W_o\big(\Phi_B(W_i\boldsymbol{h})-W_i\boldsymbol{h}\big)\\&=W_o\Phi_B(W_i\boldsymbol{h})=\pi_{n_x}\circ\beta\circ\Phi_B\circ\alpha(\boldsymbol{h}).\end{aligned}$$

Since $f^\star=\pi_{n_x}\circ\psi$ and $\|\pi_{n_x}\|_\infty=1$, applying π_{n_x} to (34) gives (29). ∎

V. NUMERICAL SIMULATIONS

In this section, we evaluate the proposed identification framework through an extensive simulation study. We first validate the main claim of the paper by comparing ResDyNet with an otherwise identical architecture without linear components, under matched parameter counts and training budgets. Next, we analyze the impact of the main architectural and training hyperparameters on identification performance. We also investigate the relationship between the learned latent-state representation and the underlying physical state, and derive practical guidelines for hyperparameter selection.

We assess predictive accuracy using the root mean square error (RMSE) and the normalized root mean square error (NRMSE). Let $\boldsymbol{y}_k$ denote the measured output sequence and let $\hat{\boldsymbol{y}}_k$ be the corresponding model predictions for a test sequence of length N, $k = \{1, \dots, N\}$. The RMSE is defined as

$$\mathrm{RMSE} := \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\|_2^2}. \tag{35}$$

This quantity measures the prediction error in the same physical unit as the output. To account for the variability of the output signal, we also report the NRMSE, defined as

$$\mathrm{NRMSE} := \frac{\mathrm{RMSE}}{\sigma_y}, \tag{36}$$

where σ_y denotes the standard deviation of the measured output,

$$\sigma_y := \sqrt{\frac{1}{N}\sum_{k=1}^{N} \|\boldsymbol{y}_k - \bar{\boldsymbol{y}}\|_2^2}, \qquad \bar{\boldsymbol{y}} := \frac{1}{N}\sum_{k=1}^{N} \boldsymbol{y}_k. \tag{37}$$

For ease of interpretation, we also report the percentage form

$$\mathrm{NRMSE}[\%] := 100\,\frac{\mathrm{RMSE}}{\sigma_y}. \tag{38}$$

A. EFFECT OF THE LINEAR COMPONENTS OF RESDYNET

A central design choice of ResDyNet is the explicit decomposition of the encoder, evolver, and decoder into a linear contribution and a nonlinear correction.

To assess the effect of this structured parametrization, we compare ResDyNet with a plain neural state-space architecture having the same organization, depth, and hidden-layer widths, but without the explicit linear paths. Moreover, the identity shortcuts within the evolver are removed, so that its hidden transformations are composed sequentially as standard feedforward layers. The comparison therefore contrasts the proposed linear–nonlinear residual parametrization with a fully nonlinear network realization.

The experiment is carried out on the Cascaded Tanks system, which describes the water-level dynamics of two interconnected tanks. Neglecting overflow effects, the process can be represented as

$$\Sigma_T := \begin{cases} \dot{x}_1(t) = -k_1\sqrt{x_1(t)} + k_4 u(t) + w_1(t), \\ \dot{x}_2(t) = k_2\sqrt{x_1(t)} - k_3\sqrt{x_2(t)} + w_2(t), \\ y(t) = x_2(t) + v(t), \end{cases} \tag{39}$$

where $u(t)$ denotes the pump input, $x_1(t)$ and $x_2(t)$ are the water levels in the upper and lower tanks, respectively, $w_1(t)$ and $w_2(t)$ represent process disturbances, and $v(t)$ is measurement noise. The benchmark provides 1024 training samples and 1024 test samples [18]. The experimental process also exhibits overflow effects, making the identification problem particularly challenging despite the limited number of available samples.

For both architectures, the latent dimension is set to $n_x = 6$, the input–output history lengths to $n_a = n_b = 10$, and the prediction horizon to $H = 10$. Encoder and decoder have two hidden layers of width 64; the evolver uses $B = 4$ residual blocks with one hidden layer of width 64 acting on a residual stream of dimension $n_h = 2q + 1 = 15$, i.e., the width prescribed by Theorem 1. The same hidden dimensions, optimizer settings, training data, and random seeds are used for the two models. In particular, the comparison is repeated over 5 paired random initializations, such that the corresponding models are trained under the same stochastic conditions.

Figure 3 reports the validation NRMSE % along the training epochs. The solid curves denote the average over the independent runs, while the shaded regions indicate one standard deviation. ResDyNet exhibits a faster decrease of the prediction error and converges to consistently lower validation NRMSE values than the plain architecture. Moreover, the reduced dispersion across initializations indicates that

TABLE 2. Comparison between ResDyNet with a complete architecture and without the linear components on the Cascaded Tanks.

ResDyNet	Params	Layers	Neurons	NRMSE [%]
w/o linear	18,822	2	64	27.60 ± 3.65
Complete	18,991	2	64	$\mathbf{14.14 \pm 2.56}$

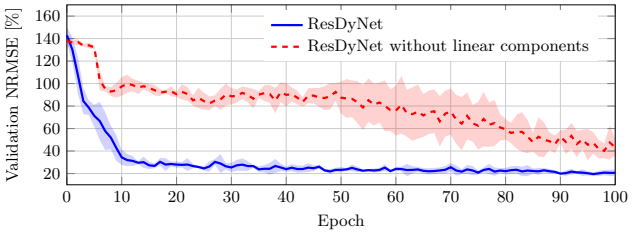


FIGURE 3. Validation NRMSE on the Cascaded Tanks benchmark for ResDyNet and the counterpart without linear components. Curves report the mean over 5 paired random initializations and shaded regions denote one standard deviation.

the structured parametrization leads to a more reproducible optimization behavior.

The performances on the test set are summarized in Table 2. The reported NRMSE values are obtained from the best checkpoint on the validation set. The parameter counts are also included to make explicit the complexity of the two parametrizations. ResDyNet introduces only a modest increase in model complexity, with 0.90% more parameters than its counterpart. The 169 additional parameters are exactly the entries of S_e, S_f, and S_g, since the identity shortcuts are parameter-free. Despite the limited parameter overhead, ResDyNet achieves improved performance, indicating that the observed gains cannot be attributed solely to a substantial increase in model size.

B. EFFECT OF THE PROPOSED INITIALIZATION

We assess the effect of the proposed linear-informed initialization introduced in Section A on the Cascaded Tanks benchmark. Random and linear-informed initialization are compared using the same training data, architecture, optimizer settings, and training budget.

In the random case, all parameters follow the standard network initialization. In the linear-informed case, the identified linear model and its finite-memory state reconstructor are embedded according to (23), while the complementary latent coordinates and nonlinear corrections are initialized as in (24). All parameters, including those initialized at zero, remain trainable during multi-step training.

The comparison is repeated over 5 paired random seeds for 60 epochs. Figure 4 reports the mean validation NRMSE %, with shaded regions denoting one standard deviation.

After the initial transient, the linear-informed initialization consistently achieves lower validation error. Its NRMSE rapidly decreases and stabilizes around 30%, whereas the randomly initialized model converges more slowly and remains close to 40% after 60 epochs. The variability across runs is also generally smaller, especially during the early stages of training.

These results indicate that the identified linear model provides a more informative starting point for nonlinear training. Since all parameters remain trainable, the linear model acts only as an initial backbone and does not constrain the final nonlinear representation.

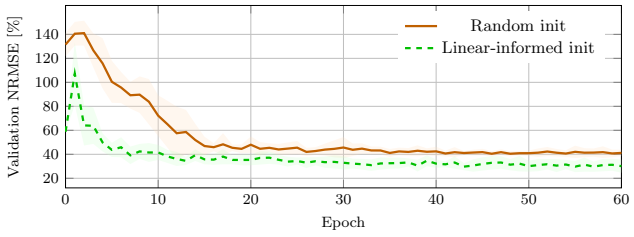


FIGURE 4. Validation NRMSE on the Cascaded Tanks system using random and linear-informed initialization. Curves report the mean over 5 paired runs and shaded regions denote one standard deviation.

C. SENSITIVITY TO ARCHITECTURAL AND TRAINING PARAMETERS

The sensitivity analyses are carried out on a nonlinear Duffing oscillator discretized with sampling time $T_s = 0.01\,\mathrm{s}$, described by

$$\Sigma_D := \begin{cases} x_{1,k+1} = x_{1,k} + T_s x_{2,k}, \\ x_{2,k+1} = x_{2,k} + T_s \left(-\delta x_{2,k} - \alpha x_{1,k} - \beta x_{1,k}^3 + \gamma u_k \right), \\ y_k = x_{1,k} + 0.05 x_{1,k}^2 + v_k, \end{cases} \tag{40}$$

where $\alpha = \beta = \gamma = 1$ and $\delta = 0.2$. To excite the system over different dynamical regimes, the input sequence is constructed by concatenating four types of excitation: random piecewise-constant steps, random-phase multisine signals, a chirp signal, and a smooth stochastic input generated through a first-order autoregressive process. All input samples are bounded such that $|u_k| \leq 2$. The measurement noise v_k is Gaussian and its variance is selected to obtain a signal-to-noise ratio of 40 dB. The resulting dataset is split into training, validation, and test sets of length 90,000, 10,000, and 20,000, respectively.

Unless otherwise stated, the analyses in the following subsections vary one design choice at a time while keeping the remaining settings fixed. The baseline configuration is $H = 10$, $n_a = n_b = 50$, and $n_x = 6$. The encoder e and decoder $\hat{g}$ consist of two hidden layers with 128 neurons each. The evolver $\hat{f}$ contains $B = 2$ residual blocks, each composed of two hidden layers with 128 neurons. All models are trained using the Adam optimizer with learning rate 10^{-4} and batch size 256.

1) SENSITIVITY WITH RESPECT TO THE OBSERVATION-WINDOW LENGTH

We start investigating the influence of the observation-window lengths n_a and n_b, which determine the amount of past output and input information supplied to the encoder for latent-state initialization. In this experiment, the two memory lengths are varied by setting $n_a = n_b \in \{10, 30, 50\}$ while keeping the remaining architecture and training hyperparameters unchanged.

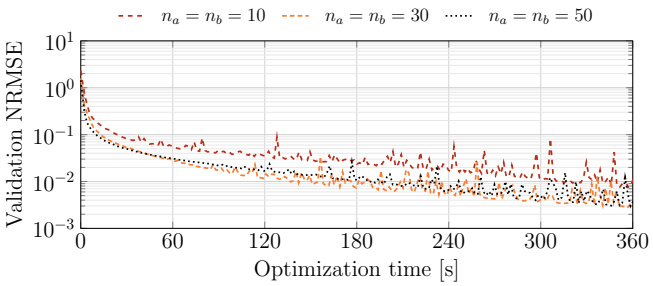


FIGURE 5. Effect of the observation-window length n_a, n_b on the NRMSE of the validation set during optimization.

Figure 5 reports the validation NRMSE over the optimization time. Among the tested configurations, $n_a = n_b = 30$ achieves the lowest validation NRMSE, whereas $n_a = n_b = 10$ yields the largest error over the considered interval. However, performance does not improve monotonically with the window length, as increasing the memory beyond 30 samples does not systematically reduce the prediction error. This behavior reflects an information–complexity trade-off. Short windows may not provide enough temporal information for the encoder to reconstruct a latent state consistent with the unmeasured physical state. Conversely, longer windows increase the encoder input dimension n_r and the number of parameters in its first layer. Older samples may also provide limited additional information while introducing redundancy and greater exposure to measurement noise, making optimization harder without a corresponding gain in generalization. The curve for $n_a = n_b = 50$ suggest that the useful information approximately saturates before the largest tested window lengths.

2) SENSITIVITY WITH RESPECT TO THE LATENT DIMENSION

The latent dimension n_x plays the role of the model order in this architecture. Figure 6 reports the validation set NRMSE during optimization for $n_x \in \{2, 6, 12\}$. For each choice of n_x, the model is trained from five independent random initializations, and the figure shows the mean trajectory together with the corresponding standard deviation.

The results show that the smallest latent dimension, $n_x = 2$, which matches the state dimension of the data-generating system Σ_D, yields the worst performance. In this case, $\tilde{n} = n_x$, so the identified linear model occupies the entire latent space and no additional latent coordinates are available to represent nonlinear components beyond the linear subspace. Increasing the latent dimension to $n_x = 6$ introduces additional degrees of freedom and substantially improves prediction accuracy, providing the best trade-off between expressiveness and generalization. A further increase to $n_x = 12$ yields no clear benefit and may introduce redundant degrees of freedom, increasing model complexity and the risk of overfitting.

Overall, these results suggest that a moderate latent dimension is preferable: it should provide sufficient additional capacity to capture nonlinear dynamics without introducing unnecessary latent degrees of freedom.

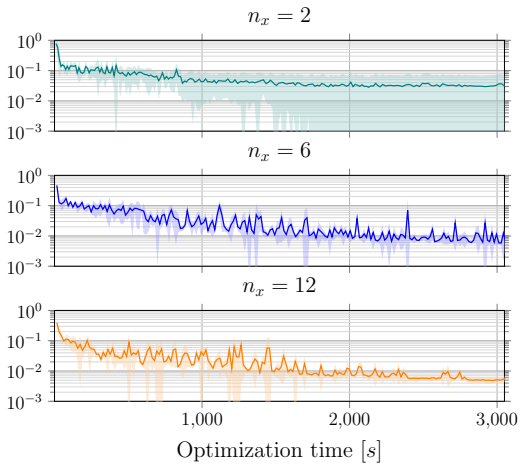


FIGURE 6. **Effect of the latent dimension n_x on the validation NRMSE during training. The solid lines represent the mean over five runs with different random initialization. The shaded region denotes the standard deviation.**

3) SENSITIVITY WITH RESPECT TO THE RESIDUAL DEPTH

We analyze the sensitivity of the evolver $\hat{f}$ to the number of residual blocks B and to the prediction horizon H used during training. Figure 7 reports the validation NRMSE % for all combinations of $B \in \{1, 2, 4, 6\}$ and $H \in \{1, 5, 10, 20\}$.

The largest errors are observed for $H = 1$ across all values of B. In this case, the training loss contains only the prediction at the initial rollout step and therefore does not depend on any state propagated through the evolver. Hence, the evolver is not trained through recursive predictions. Increasing H beyond one step consistently reduces the validation error, with diminishing improvements beyond $H = 10$.

For $H \geq 5$, increasing B improves prediction accuracy, with the best result obtained at $B = 6$ and $H = 10$, where the validation NRMSE reaches 1.45%. However, the gain from $B = 2$ to $B = 6$ is modest compared with that from $B = 1$ to $B = 2$, indicating diminishing returns as residual depth increases, together with higher parameter count and optimization complexity.

4) EFFECT OF THE PREDICTION-HORIZON WEIGHTS

To assess the role of the weighting coefficients γ_h in the multi-step loss, we compare three profiles while keeping the model architecture and all remaining training settings unchanged. In particular, we consider uniform weighting, exponentially decaying weights that emphasize the first prediction steps, and exponentially increasing weights that emphasize the final portion of the rollout. In all cases, the weights are normalized according to (25).

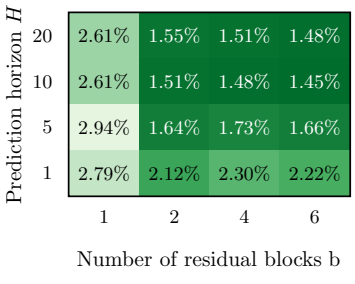


FIGURE 7. **Validation NRMSE % obtained by training the proposed model for different numbers of residual blocks B and prediction horizons H. Each cell reports the error for one trained model.**

Figure 8 shows that the horizon weights mainly redistribute prediction accuracy along the rollout. Decaying weights retain performance near the beginning but deteriorate toward the end, whereas increasing weights sacrifice short-term accuracy in favor of the final prediction steps. Uniform weighting provides the most balanced behavior across the complete horizon.

D. HYPERPARAMETER CHOICE AND GUIDELINES

The numerical simulation on the Duffing oscillator gives us some guidelines on the choice of the main hyperparameters:

- *Observation window lengths* n_a, n_b. These parameters determine the information available to the encoder. Too short histories may lead to poor latent state initialization, while unnecessarily long histories increase the regressor dimension and may make training more sensitive to noise.
- *Latent dimension* n_x. The latent dimension plays the role of a model order. It should be large enough to represent the relevant dynamics, but excessive values may introduce redundant degrees of freedom and degrade generalization.
- *Prediction horizon* H. Larger horizons promote long-term consistency, but increase the length of the computational graph and may make optimization harder. The horizon should match the intended use of the model.
- *Horizon weights* γ_h. Uniform weights provide a balanced objective. Increasing weights emphasize long-horizon accuracy, whereas decreasing weights favor short-term prediction. The normalization in (25) keeps the loss scale comparable across different horizons.

E. ANALYSIS OF THE LEARNED LATENT STATE

We investigate whether the learned latent state admits a state-like interpretation by testing its relation with the physical state. This analysis is deliberately carried out on a linear system. For minimal LTI realizations, equivalent state representations are related by an invertible linear change of coordinates; here, an additional offset is allowed to account for the biases of the neural model. For nonlinear systems, instead, equivalent state representations may in general be related through nonlinear coordinate transformations, making an affine matching test unnecessarily restrictive.

We consider an RLC circuit described by

$$\Sigma_{\text{LTI}} := \begin{cases} \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -R/L & -1/L \\ 1/C & 0 \end{bmatrix} \boldsymbol{x}(t) + \begin{bmatrix} 1/L \\ 0 \end{bmatrix} u(t), \\ y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} \boldsymbol{x}(t), \end{cases} \tag{41}$$

where $\boldsymbol{x}(t) = [\, i_L(t) \quad v_C(t) \,]^\top$, $R = 2.0$, $L = 0.05$, and $C = 0.0002$. The model is discretized under zero-order hold with sampling time $T_s = 0.005$ s, yielding

$$\boldsymbol{x}_{k+1} = A_d \boldsymbol{x}_k + B_d u_k, \qquad y_k = C \boldsymbol{x}_k. \tag{42}$$

Since the physical state dimension is known, we set $n_x = n_s = 2$ and test whether the learned coordinates satisfy

$$\hat{\boldsymbol{x}}_k \approx P \boldsymbol{x}_k + \boldsymbol{c}, \tag{43}$$

for some $P \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{c} \in \mathbb{R}^{n_x}$. Let X and $\hat{X}$ collect the physical and latent trajectories, respectively. The affine map is estimated through

$$\min_{P,\boldsymbol{c}} \left\| \hat{X} - \left(X P^\top + \mathbf{1} \boldsymbol{c}^\top \right) \right\|_F^2. \tag{44}$$

A small residual therefore indicates that the learned latent coordinates are approximately affinely related to the physical state.

We evaluate this relation in two operating modes: (i) the latent state is reconstructed independently at each time step through the encoder, and (ii) the encoder is used only for initialization, after which the state is recursively propagated through the evolver $\hat{f}$. The corresponding relative affine matching errors are reported in Figs. 9 and 10. In both cases, the excitation switches from a sinusoidal input to a chirp at $k = 800$, thereby testing the correspondence under a broader excitation range. To avoid imposing the alignment by construction, all parameters are randomly initialized in this experiment, i.e., the linear-informed initialization of Section A is not used.

When the encoder is repeatedly applied, the affine matching error remains bounded but exhibits larger oscillations and a transient around the change in excitation. When the state is instead propagated through the evolver, the error is smaller

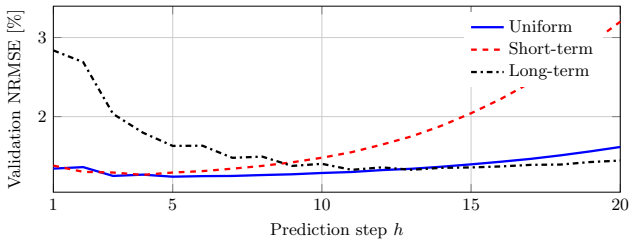


FIGURE 8. Validation NRMSE% as a function of the prediction step for uniform, short-term, and long-term horizon-weight profiles. The weighting coefficients γ_h are normalized according to eq. (25)

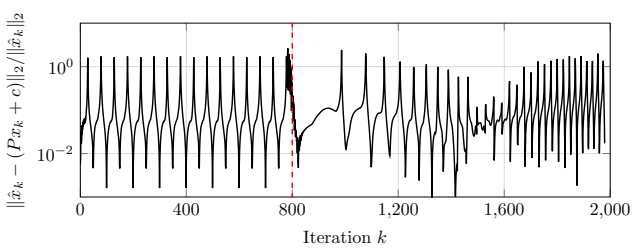


FIGURE 9. Relative affine matching error $\|\hat{\boldsymbol{x}}_k - (P\boldsymbol{x}_k + \boldsymbol{c})\|_2 / \|\hat{\boldsymbol{x}}_k\|_2$ when the latent state is reconstructed at each time step by the encoder. The input is sinusoidal for $k < 800$ and chirp-shaped for $k \geq 800$.

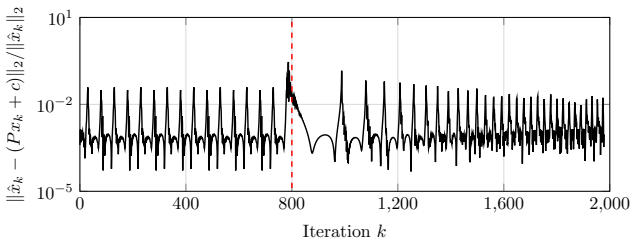


FIGURE 10. **Relative affine matching error when the latent state is initialized by the encoder and subsequently propagated through the evolver.**

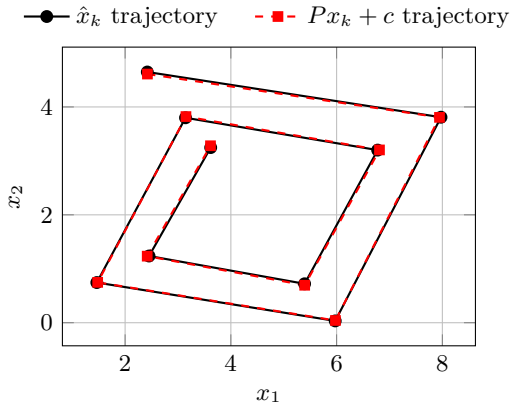


FIGURE 11. **Comparison between $\hat{x}_k$ and $P\boldsymbol{x}_k + \boldsymbol{c}$ under a constant input over $N = 10$ time steps.**

and more regular, indicating a more consistent alignment between latent and physical coordinates.

Finally, Fig. 11 compares $\hat{\boldsymbol{x}}_k$ with $P\boldsymbol{x}_k + \boldsymbol{c}$ under a constant input. Their close agreement further supports an approximately affine relation between the learned latent state and the physical state.

VI. BENCHMARK SIMULATIONS

We finally assess ResDyNet on two standard nonlinear system-identification benchmarks: the Wiener–Hammerstein system and the Cascaded Tanks process. The latter has already been introduced in the previous section and is reconsidered here only for comparison with results reported in the literature.

The Wiener–Hammerstein benchmark consists of a static nonlinearity sandwiched between two linear dynamic blocks, as illustrated in Fig. 12. Its input–output behavior can be written schematically as

$$y(t) = g_2\big(f(g_1(u(t)))\big),$$

where g_1 and g_2 denote linear dynamic systems and f is a static nonlinear map. The dataset contains 80,000 training samples, 20,000 validation samples, and a separate test sequence of 88,000 samples.

TABLE 3. **Comparison with literature results on the Wiener–Hammerstein and Cascaded Tanks benchmarks.**

Wiener–Hammerstein benchmark			
Identification method	**RMSE** [mV]	**NRMSE** [%]	**Time [h]**
Proposed method	0.237	0.0983	< 34
Encoder Subspace [14]	0.241	0.0987	> 200
DynoNet [19]	1.75	0.708	< 1
LS-SVM [20]	4.07	1.64	n.r.
Masti & Bemporad [13]	12.010	4.9070	< 1
SVM [21]	47.400	19.3730	n.r.
BLA [22]	56.200	22.9690	n.r.

Cascaded Tanks benchmark			
Identification method	**RMSE** [V]	**NRMSE** [%]	**Time [sec]**
Proposed method	0.168	7.99	60
Beintema et al. [23]	0.306	14.60	n.r.
Forgione & Piga [4]	0.330	15.70	n.r.
DynoNet [19]	0.421	20.10	60
Champneys et al. [24]	0.697	33.20	n.r.

For the Wiener–Hammerstein benchmark, input–output histories of length $n_a = n_b = 50$ are used, with latent dimension $n_x = 14$ and prediction horizon $H = 80$.

For the Cascaded Tanks benchmark, the evolver consists of $B = 2$ residual blocks with 64 neurons per layer and $\tanh$ activation. The latent dimension is set to $n_x = 8$, the observation windows to $n_a = n_b = 10$, and the prediction horizon to $H = 10$.

For both benchmarks, the models are trained using Adam with learning rate 10^{-3} and early stopping based on the validation loss.

The benchmark results are summarized in Table 3. On the Wiener–Hammerstein dataset, ResDyNet attains an RMSE of 0.237 mV and an NRMSE of 0.0983%, which are comparable to the best results reported in the literature and slightly below those reported for Encoder Subspace in [14]. The corresponding training time remains substantially lower than the reported optimization time of that method.

On the Cascaded Tanks benchmark, ResDyNet achieves an RMSE of 0.168 V and an NRMSE of 7.99%. This result further confirms that the proposed structured latent dynamics can provide accurate multi-step predictions also on a comparatively small experimental dataset.

Unlike the controlled comparison of Section A, where both architectures share a common, non-optimized config-

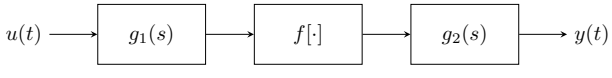


FIGURE 12. **Wiener–Hammerstein benchmark: cascade of a linear dynamic block g_1, a static nonlinear map f, and a linear dynamic block g_2.**

uration, the hyperparameters used here were selected on the validation set to maximize performance.

VII. CONCLUSIONS

This paper introduced ResDyNet, a structured neural state-space model that combines explicit linear components with nonlinear corrections in the encoder, state-transition map, and decoder. The nonlinear correction of the evolver is represented through a deep residual cascade, while an auxiliary linear model and its finite-memory state reconstructor provide a structured initialization of the complete architecture. The resulting model is subsequently trained through a weighted multi-step prediction objective.

We showed that the proposed evolver retains universal approximation capability under the stated assumptions. A controlled ablation against a comparable architecture without linear components showed a clear improvement in multi-step prediction accuracy with limited parameter overhead, while the benchmark experiments yielded competitive results on the Wiener–Hammerstein and Cascaded Tanks datasets. Future work will investigate the proposed initialization and structured parametrization on larger-scale and control-oriented identification problems.

REFERENCES

[1] L. Ljung, *System Identification*. Boston, MA: Birkhäuser Boston, 1998, pp. 163–173.

[2] J. Schoukens and L. Ljung, "Nonlinear system identification: A user-oriented road map," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 28–99, 2019.

[3] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[4] M. Forgione and D. Piga, "Continuous-time system identification with neural networks: Model structures and fitting criteria," *European Journal of Control*, vol. 59, pp. 69–81, 2021.

[5] G. Pillonetto, A. Aravkin, D. Gedon, L. Ljung, A. H. Ribeiro, and T. B. Schön, "Deep networks for system identification: A survey," *Automatica*, vol. 171, p. 111907, 2025.

[6] M. Forgione, A. Muni, D. Piga, and M. Gallieri, "On the adaptation of recurrent neural networks for system identification," *Automatica*, vol. 155, p. 111092, 2023.

[7] J. Xie, F. Bonassi, and R. Scattolini, "Learning control affine neural narx models for internal model control design," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 8137–8149, 2025.

[8] M. Ledda, A. Giua, and M. Franceschelli, "Evaluating performance of neural network models for nonlinear dynamical systems via adversarial perturbations," in *2026 IEEE 22nd Conference on Automation Science and Engineering (CASE)*, 2026. To appear.

[9] M. Ledda, D. Deplano, A. Giua, and M. Franceschelli, "On the computation of relu-based rnns equivalent to cpwa models of dynamical systems and vice versa," in *2026 23rd IFAC World Congress*, 2026. To appear.

[10] H. Zhou, C. Ibrahim, W. X. Zheng, and W. Pan, "Sparse bayesian deep learning for dynamic system identification," *Automatica*, vol. 144, p. 110489, 2022.

[11] M. Ledda, D. Deplano, A. Giua, and M. Franceschelli, "Certification of autoencoder-based models for dynamical systems," in *2025 IEEE 64th Conference on Decision and Control (CDC)*, 2025, pp. 401–406.

[12] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021, 19th IFAC Symposium on System Identification SYSID 2021.

[13] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, p. 109666, 2021.

[14] G. I. Beintema, M. Schoukens, and R. Tóth, "Deep subspace encoders for nonlinear system identification," *Automatica*, vol. 156, p. 111210, 2023.

[15] R. Ramkannan, G. I. Beintema, R. Tóth, and M. Schoukens, "Initialization approach for nonlinear state-space identification via the subspace encoder approach," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 5146–5151, 2023, 22nd IFAC World Congress.

[16] P. Tabuada and B. Gharesifard, "Universal approximation power of deep residual neural networks through the lens of control," *IEEE Transactions on Automatic Control*, vol. 68, no. 5, pp. 2715–2728, 2023.

[17] ——, "Correction to "universal approximation power of deep residual neural networks through the lens of control"," *IEEE Transactions on Automatic Control*, vol. 69, no. 7, pp. 4891–4892, 2024.

[18] M. Schoukens and J. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, pp. 446–451, 07 2017.

[19] M. Forgione and D. Piga, "dynonet: A neural network architecture for learning dynamical systems," *International Journal of Adaptive Control and Signal Processing*, vol. 35, 01 2021.

[20] T. Falck, K. Pelckmans, J. A. Suykens, and B. De Moor, "Identification of wiener-hammerstein systems using ls-svms," *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 820–825, 2009, 15th IFAC Symposium on System Identification.

[21] A. Marconato and J. Schoukens, "Identification of wiener-hammerstein benchmark data by means of support vector machines," *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 816–819, 2009, 15th IFAC Symposium on System Identification.

[22] L. Lauwers, R. Pintelon, and J. Schoukens, "Modelling of wiener-hammerstein systems via the best linear approximation," *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 1098–1103, 2009, 15th IFAC Symposium on System Identification.

[23] G. I. Beintema, M. Schoukens, and R. Tóth, "Continuous-time identification of dynamic state-space models by deep subspace encoding," 2023.

[24] M. D. Champneys, G. I. Beintema, R. Tóth, M. Schoukens, and T. J. Rogers, "Baseline results for selected nonlinear system identification benchmarks," 2024.



Marco Ledda (Student, IEEE) is a Ph.D. candidate in Systems and Control at the University of Cagliari, Italy. He earned his Bachelor's degree in Electrical, Electronic, and Computer Engineering from the University of Cagliari in 2022, followed by a Master's degree in Computer Engineering, Cybersecurity, and Artificial Intelligence, graduating with honors in 2024 from the same institution. His research focuses on the certification and safety of learning-enabled autonomous systems, with particular emphasis on learning-based control, neural-network-based system modeling, and robustness to adversarial perturbations.



Diego Deplano (Member, IEEE) is Assistant Professor (RTD-A) at the Department of Electrical and Electronic Engineering, University of Cagliari, Italy. He received the B.S., M.S., and PhD degrees in Electronic Engineering "cum laude" from the University of Cagliari, Italy, respectively in '15, '17, and '21. He was awarded the best PhD Thesis defended in the area of Systems and Control Engineering at an Italian University by SIDRA. He spent visiting periods at the Nanyang Technological University (NTU), Singapore, at the Centre National de la Recherche Scientifique (CNRS), Grenoble, France, and at the University of Toronto (UofT), Toronto, Canada. He has lectured seminars at KTH in Stockholm, Sweden, and at Peking University in Beijing, China. He is a member of the Conference Editorial Board (CEB) for the IEEE Control Systems Society (CSS) and serves as Associate Editor for the IEEE Conference on Decision

and Control (CDC) and the IEEE American Control Conference (ACC) since 2025. He also serves as an Associate Editor for the IEEE Conference on Automation Science and Engineering (CASE) since 2024. His research interests include open multi-agent systems, distributed optimization and learning, dynamic consensus problems, monotone dynamical systems, contraction theory.



Alessandro Giua (Fellow, IEEE) received the Laurea degree in electrical engineering from the University of Cagliari, Cagliari, Italy, in 1988 and the master's and Ph.D. degrees in computer and systems engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990 and 1992, respectively. He is currently a Professor of automatic control with the Department of Electrical and Electronic Engineering, University of Cagliari. He has also held faculty or visiting positions in several institutions worldwide, including Aix-Marseille University, Marseille, France, and Xidian University, Xi'an, China. On the topics of his research work, he has authored or coauthored two books, more than 300 papers in international journals and conferences and has often been invited to give plenary talks or keynotes. His research interests include discrete event systems, hybrid systems, networked control systems, Petri nets, and failure diagnosis. Dr. Giua has been Principal Investigator and Group Leader for several projects, financed by the European Union, the Italian state, and other organizations. He is a Member of the IEEE Control Systems Society, where he was a Vice President for Conference Activities (2020–2021), General Chair of the 55th Conference on Decision and Control, in 2016, and a Member of the Board of Governors (2013–2015). He is an affiliate of the International Federation of Automatic Control (IFAC) where he has served as the Chair of the IFAC Technical Committee 1.3 on Discrete Event and Hybrid Systems (2008–2014). He has served as Editor-in-Chief of the IFAC journal Nonlinear Analysis: Hybrid Systems (2014–2022), Senior Editor of IEEE TRANSACTIONS ON AUTOMATIC CONTROL (2017–2022), and Department Editor of the Journal Discrete Event Dynamic Systems (2015–2020). He is a Fellow of the IFAC for contributions to Discrete Event and Hybrid Systems. He was the recipient of the IFAC Outstanding Service Award and the People's Republic of China Friendship Award in 2017



Mauro Franceschelli (Senior, IEEE) is an Associate Professor at the Department of Electrical and Electronic Engineering, University of Cagliari, Italy. He received the Laurea degree in Electronic Engineering "cum laude" in 2007 and the PhD degree in Electronic and Computer Engineering in 2011 from the University of Cagliari. He spent visiting periods at the Georgia Institute of Technology (GaTech), and the University of California at Santa Barbara (UCSB), USA. In 2013 he received a fellowship from the National Natural Science Foundation of China (NSFC), at Xidian University, Xi'an, China. In 2015 he was awarded a position of Assistant Professor (RTD-A) funded by the Italian Ministry of Education, University and Research (MIUR) under the 2014 call "Scientific Independence of Young Researchers" (SIR). He is currently an Associate Editor for the IEEE Transactions on Automatic Control (TAC) since 2025 and IFAC Nonlinear Analysis: Hybrid Systems (NAHS) since 2023. He is a member of the Conference Editorial Board (CEB) for the IEEE Control Systems Society (CSS) since 2019. He serves as Associate Editor for the IEEE Conference on Automation Science and Engineering (CASE) since 2015. He was an Associate Editor for the IEEE Transactions on Automation Science and Engineering (T-ASE) 2021-2023 and Guest Associate Editor for the IEEE Open Journal of Control Systems (OJ-CSYS) in 2024. He is Officer (vice-Chair) for the IEEE Italy Section Chapter of the IEEE Control Systems Society since 2025. His research interests include consensus problems, open multi-agent systems, multi-robot systems, networks of energy systems, resilient networks, distributed optimization and learning.