PhyGRU: a Physics-Biased Variant for Gated Recurrent Unit

Girolamo Oddo

Abstract

The modeling of dynamical systems often takes place in regimes where data are limited and physical knowledge is only partially available. In such conditions, simplified physical models are often insufficient, while fully data-driven recurrent networks struggle to generalize. Moreover, many existing hybrid solutions require substantial architectural modifications or intervention, making them difficult to integrate into existing models or pipelines without major restructuring. To address this, PhyGRU is introduced, a physics-biased variant of the Gated Recurrent Unit in which the learned candidate state is replaced by an explicit time integration of a parametric physical model, optionally augmented by low-dimensional latent dynamics. The standard GRU gating mechanism is preserved, enabling interpolation between the previous state and a physics-informed candidate. PhyGRU targets scenarios with limited data and partial physical knowledge, where neither simplified physical models nor fully data-driven recurrent networks are sufficient. The approach requires only minimal architectural modifications relative to a standard GRU, at the cost of increased per-step computational time due to explicit state integration. Experiments, on three controlled dynamical systems of increasing complexity, under time-invariant and time-varying parameter regimes, indicate that PhyGRU can yield more physically coherent predicted trajectories when the assumed physical prior is reasonably aligned with the true dynamics, while performance degrade as the mismatch between the prior and the true system increases.

1 Introduction

In recent years, driven by the growing attention toward artificial intelligence, data-driven modeling has attracted significant interest from the scientific community [[2], [20]]. This has quickly highlighted the need to combine and exploit hybrid approaches that leverage the extensive available knowledge in terms of physical models together with the modeling and system identification capabilities provided by machine learning methodologies. To this end, several works have emerged, presenting different advantages and drawbacks as well as varying degrees of integration of physical information known a priori. As a result, the literature now offers a wide range of methods and models, spanning from purely data-driven to purely physics-based approaches.

This work positions itself in the middle of this spectrum, addressing a scenario characterized by a limited amount of data and a simplified physical model known a priori. In such a setting, the physical model alone would be insufficient due to its excessive simplifications, while the available data alone would not be enough to build an effective and reliable data-driven model. To address this, a variant of the well-known and widely adopted Gated Recurrent Unit (GRU) architecture [1], is presented introducing a minimal modification to its structure to allow the inclusion of an ODE within its internal state. This effectively enforces an inductive bias during training based on the known physical representation.

Compared to a standard GRU, this approach has shown to be an effective support across several configurations, even when the a priori physics was, in practice, inaccurate with respect to what the model observed during training, this still represents a bias that the model can exploit, even if only to a limited extent. Furthermore, in comparison with the various methodologies and models

available in the literature for tackling this problem, the proposed approach stands out for its very low practical adoption complexity, especially in systems already based on GRU cells. It therefore represents an initial and accessible pathway for introducing system knowledge without requiring disruptive changes to existing workflows, exploiting the well-known recurrent cells approach.

2 Related Works

Recurrent neural networks are widely used for data-driven modeling of dynamical systems, particularly when temporal dependencies and latent-state evolution are central. In recent years, several works have explored how prior knowledge of system dynamics can be incorporated into recurrent architectures to improve data efficiency, stability, and extrapolation. Existing approaches can be broadly organized according to how the dynamical prior interacts with the recurrent model.

2.1 Dynamical Priors as Inputs or Output Constraints

A common strategy consists in preserving the standard GRU or LSTM recurrence while injecting prior dynamical information at the input or output level. In this setting, predictions from a simplified dynamical model or simulator are concatenated with measured signals and processed by a recurrent network. For example, Jia *et al.* [2] proposed a physics-guided RNN in which outputs from a process-based dynamical model are provided as additional inputs to an LSTM, enabling the network to learn corrections to the nominal dynamics. Related formulations apply recurrent networks to estimate intermediate states, which are subsequently mapped to physically consistent outputs via known dynamical relationships [5]. This is one of the most common approaches and offers the advantage of maintaining full control over the dynamical model. However, it provides little support for the training of the neural component, as the learning procedure remains essentially standard, with only additional inputs or guardrails being introduced.

2.2 Hybrid and Parallel Dynamical Models

Another line of work combines recurrent networks with simplified dynamical models in a hybrid or parallel configuration. In such architectures, the recurrent network and the dynamical model operate as separate modules, with their predictions fused at a higher level. Zarzycki and Lin [3, 4] proposed variants where a nominal model-based component is combined with a GRU trained on residuals; similar fusion ideas appear in other hybrid proposals [20]. These approaches are effective when a prior dynamical model is available, but they typically treat the dynamical prior as an external correction rather than as part of the recurrent state evolution itself.

2.3 Dynamical Constraints through Training Objectives

Several methods incorporate prior knowledge of system dynamics through training-time constraints rather than architectural changes. Physics-informed recurrent neural networks enforce consistency with known differential equations by augmenting the loss function or by constraining the optimization process [6]. Bayesian recurrent models further encode dynamical assumptions through probabilistic priors, improving robustness and uncertainty quantification in long-term forecasting tasks [7].

2.4 Augmented Recurrent Architectures

Recent work has explored enhanced recurrent architectures tailored to specific dynamical modeling tasks, including attention-based mechanisms and adaptive weighting schemes. For instance, Al-Saleh

et al. [8] introduced an attention-augmented GRU trained with physics-informed objectives for battery health estimation. These models enhance expressivity or training robustness but generally preserve the standard formulation of the recurrent update.

2.5 Neural ODE and Integration-Inspired Recurrent Models

A substantial body of literature generalizes discrete RNNs to continuous-time dynamics or otherwise integrates numerical solvers into the recurrent computation:

- **Neural ODEs and latent ODEs.** The Neural ODE framework [9] and its extensions, including Latent ODEs and ODE-RNN models [10], describe the hidden-state evolution as the solution of a continuous-time ordinary differential equation. In these approaches, the governing vector field is learned from data and integrated numerically between observations, enabling flexible handling of irregular sampling but relying on a fully data-driven representation of the system dynamics.
- **Neural CDEs.** Neural Controlled Differential Equations extend Neural ODEs and latent ODEs idea to controlled dynamics driven by input paths [11], providing a principled continuous-time generalization of recurrent models for irregularly sampled time series.
- **ODE-based RNN variants.** Works such as Habiba, Pearlmutter ("ODE-GRU", "ODE-LSTM") & Hagge "ODE-RNN" explicitly redesign GRU/LSTM cells so the hidden or cell states evolve under an ODE solver between observations [12, 19].
- **Integration-inspired cell designs.** Several recent contributions embed specific numerical integration schemes directly into recurrent cells. For instance, RKGRU [13] integrates Runge–Kutta ideas into the GRU update; DynNet [14] and Newmark-inspired cells incorporate implicit solver structure into the recurrent cell for structural dynamics; Guo & Xu (Phy-RLK) embed a Newmark residual inside an LSTM via gating modifications [15]. Related works (RKRNN, RKN-like RNNs) also use Runge–Kutta or Newmark motifs for dynamical prediction [16].
- **Hybrid ODE–RNN fusions.** Other proposals merge Neural ODEs with gated architectures (e.g., SHGRU-DODE / SHGRU-DN [17]) or introduce multi-step integration-inspired attention in latent-space RNNs [18].

2.6 Positioning and Contribution

From the perspective of dynamical systems modeling, the proposed PhyGRU architecture can be interpreted as a gray-box recurrent model that lies between purely data-driven gated recurrent networks and fully physics-based numerical integration schemes. The guiding principle behind PhyGRU is to incorporate physical structure while introducing minimal modifications to the standard GRU architecture, thereby facilitating adoption in existing recurrent modeling pipelines.

In contrast to Neural ODE models and ODE-based RNN variants [12, 11], which describe the hidden-state evolution through learned continuous-time vector fields, PhyGRU operates entirely in discrete time. Neural ODE formulations define the latent dynamics as a continuous-time initial value problem and rely on numerical ODE solvers to propagate the hidden state. Training typically requires backpropagation through the solver, for instance, via adjoint-based methods, which increases architectural and implementation complexity. While this continuous-time formulation can offer greater modeling flexibility, it also departs substantially from conventional recurrent network designs.

By contrast, the novelty of the proposed PhyGRU does not lie in introducing a new continuous-time formulation or a redesigned integration scheme, but in how physical knowledge is incorporated within a standard discrete-time recurrent architecture. PhyGRU preserves the canonical GRU recurrence and fixed temporal discretization. No external ODE solver is introduced, and gradient propagation follows the conventional backpropagation-through-time procedure.

The key architectural choice is to embed a parametric physical state propagation exclusively within the candidate hidden-state computation. This propagation can be interpreted as a single-step numerical integration of a known or partially known dynamical model. Importantly, the GRU gating structure remains unchanged. As a result, the update gate naturally regulates the balance between physics-based evolution and data-driven correction without requiring additional coupling mechanisms or parallel dynamical branches.

Several recent works have incorporated physical priors into recurrent architectures by explicitly redesigning the state-transition mechanism through numerical integration schemes such as Runge–Kutta or Newmark methods [14, 16, 13, 15], or by coupling recurrent networks with physics-based modules via residual connections or auxiliary constraints [20]. In these approaches, the physical model typically alters the global state-transition structure or introduces parallel dynamical pathways.

In contrast, PhyGRU follows a deliberately minimalist design philosophy: the physical model is injected locally at the level of the candidate state, without modifying the gating equations or introducing additional solver dependencies. This results in a gray-box recurrent cell that retains the training stability and implementation simplicity of a classical GRU, while embedding an explicit inductive bias derived from prior knowledge of the system dynamics.

This inductive bias does not need to be perfectly accurate to be beneficial. Even an approximate physical propagation introduces structural information that the network can exploit during learning, especially in regimes with limited data. To the best of the author's knowledge, embedding a parametric dynamical propagation solely within the candidate hidden-state computation, while leaving the GRU gating mechanism fully intact, has not been explicitly reported in the existing literature. PhyGRU is therefore proposed as a minimal yet principled extension of the classical GRU architecture for scenarios where partial physical knowledge is available.

3 From GRU to PhyGRU

In this section, the mathematical structure of PhyGRU is presented, starting from the classical GRU architecture and describing, step by step, the introduced variations and their rationale.

3.1 Standard GRU Overview

A standard GRU [1] model is reported below:

$$\mathbf{z}_k = \sigma(W_z \mathbf{u}_k + U_z \mathbf{h}_{k-1} + \mathbf{b}_z), \tag{1}$$

$$\mathbf{r}_k = \sigma(W_r \mathbf{u}_k + U_r \mathbf{h}_{k-1} + \mathbf{b}_r), \tag{2}$$

$$\tilde{\mathbf{h}}_k = \tanh\big(W_h \mathbf{u}_k + U_h(\mathbf{r}_k \odot \mathbf{h}_{k-1}) + \mathbf{b}_h\big), \tag{3}$$

$$\mathbf{h}_k = (\mathbf{1} - \mathbf{z}_k) \odot \mathbf{h}_{k-1} + \mathbf{z}_k \odot \tilde{\mathbf{h}}_k, \tag{4}$$

$$\mathbf{x}_k = W_{\text{out}}\, \mathbf{h}_k + \mathbf{b}_{\text{out}}, \tag{5}$$

The last line (5) defines the model output at step k as a linear readout from the hidden state. In compact dynamical-system notation, the overall mapping may be written as follows.

$$\mathbf{x}_k = \text{GRU}(\mathbf{u}_k),$$

meaning that the sequence-to-sequence transformation produced by the recurrent cell yields an output $\mathbf{x}_k$ at each step, implicitly depending on past inputs through the hidden state.

Where d denotes the number of input features, n the number of hidden/output features and (L) the parameter matrices and vector which need to be learned during training:

Symbol	Description
$\mathbf{u}_k \in \mathbb{R}^d$	input vector at time step k
$\mathbf{h}_k \in \mathbb{R}^n$	hidden state vector at time step k
$\tilde{\mathbf{h}}_k \in \mathbb{R}^n$	candidate activation vector
$\mathbf{z}_k \in (0,1)^n$	update gate vector
$\mathbf{r}_k \in (0,1)^n$	reset gate vector
$\mathbf{x}_k \in \mathbb{R}^m$	model output at step k
$W_{\text{out}} \in \mathbb{R}^{m \times n}$	output weight matrix (L)
$\mathbf{b}_{\text{out}} \in \mathbb{R}^m$	output bias vector (L)
W_z, W_r, $W_h \in \mathbb{R}^{n \times d}$	input-to-hidden weight matrices (L)
U_z, U_r, $U_h \in \mathbb{R}^{n \times n}$	hidden-to-hidden weight matrices (L)
$\mathbf{b}_z$, $\mathbf{b}_r$, $\mathbf{b}_h \in \mathbb{R}^n$	bias vectors (L)
$\mathbf{1} \in \mathbb{R}^n$	vector of ones (used in $\mathbf{1} - \mathbf{z}_k$)
$\odot$	element-wise (Hadamard) product
$\sigma : \mathbb{R} \to (0,1)$	sigmoid / logistic activation
$\tanh : \mathbb{R} \to (-1,1)$	hyperbolic tangent activation

The candidate $\tilde{\mathbf{h}}_k$ is a fully learned nonlinear transformation. The GRU is expressive but purely data-driven. The proposed PhyGRU preserves the gated update structure of the standard GRU in Eq. (4), while replacing the learned candidate $\tilde{\mathbf{h}}_k$ with a physics-biased candidate obtained by time integration of physics prior provided.

3.2 PhyGRU Overview

The complete PhyGRU update at step k can be written compactly as follow and a schematic comparison respect standard full gated GRU cell is reported in Figure 1:

$$\mathbf{z}_k = \sigma\left(W_z \begin{bmatrix} \mathbf{h}_{k-1} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_z\right), \tag{6}$$

$$\dot{\boldsymbol{\xi}}_{k-1} = f_{\text{phy}}(\boldsymbol{\xi}_{k-1}, \mathbf{u}_k; \boldsymbol{\theta}_{\text{phy}}), \tag{7}$$

$$\boldsymbol{\xi}_k^{\text{cand}} = \boldsymbol{\xi}_{k-1} + \Delta t\, \dot{\boldsymbol{\xi}}_{k-1}, \tag{8}$$

$$\dot{\boldsymbol{\ell}}_{k-1} = W_{\text{lat}} \begin{bmatrix} \mathbf{h}_{k-1} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_{\text{lat}}, \tag{9}$$

$$\boldsymbol{\ell}_k^{\text{cand}} = \boldsymbol{\ell}_{k-1} + \Delta t\, \dot{\boldsymbol{\ell}}_{k-1}, \tag{10}$$

$$\tilde{\mathbf{h}}_k = \begin{bmatrix} \boldsymbol{\xi}_k^{\text{cand}} \\ \boldsymbol{\ell}_k^{\text{cand}} \end{bmatrix}, \tag{11}$$

$$\mathbf{h}_k = (\mathbf{1} - \mathbf{z}_k) \odot \mathbf{h}_{k-1} + \mathbf{z}_k \odot \tilde{\mathbf{h}}_k, \tag{12}$$

$$\mathbf{x}_k = W_{\text{out}}\, \mathbf{h}_k + \mathbf{b}_{\text{out}}. \tag{13}$$

Analogously to the GRU case, the whole module may be viewed as a discrete-time dynamical system producing an output,

$$\mathbf{x}_k = \text{PhyGRU}(\mathbf{u}_k),$$

where the right-hand side indicates the recurrent transformation that depends on past inputs through the recurrent state.

Where d denotes the number of input features, n_ξ the dimension of the physical internal state obtained by the physics prior, n_ℓ the dimension of the latent state, $n = n_\xi + n_\ell$ the total recurrent-state dimension and (L) the parameter matrices and vector which need to be learned during training.

Symbol	Description
$\mathbf{u}_k \in \mathbb{R}^d$	input vector at step k
$\boldsymbol{\xi}_k \in \mathbb{R}^{n_\xi}$	physics prior state component of the recurrent state
$\boldsymbol{\ell}_k \in \mathbb{R}^{n_\ell}$	latent state component of the recurrent state
$\dot{\boldsymbol{\xi}}_{k-1} = f_{\text{phy}}(\boldsymbol{\xi}_{k-1}, \mathbf{u}_k; \boldsymbol{\theta}_{\text{phy}}) \in \mathbb{R}^{n_\xi}$	physics prior state time derivative
$\boldsymbol{\xi}_k^{\text{cand}} = \boldsymbol{\xi}_{k-1} + \Delta t\, \dot{\boldsymbol{\xi}}_{k-1} \in \mathbb{R}^{n_\xi}$	physics prior candidate obtained by time integration
$\dot{\boldsymbol{\ell}}_{k-1} = W_{\text{lat}} \begin{bmatrix} \mathbf{h}_{k-1} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_{\text{lat}} \in \mathbb{R}^{n_\ell}$	latent state time derivative (linear map)
$\boldsymbol{\ell}_k^{\text{cand}} = \boldsymbol{\ell}_{k-1} + \Delta t\, \dot{\boldsymbol{\ell}}_{k-1} \in \mathbb{R}^{n_\ell}$	latent candidate obtained by time integration
$\tilde{\mathbf{h}}_k = \begin{bmatrix} \boldsymbol{\xi}_k^{\text{cand}} \\ \boldsymbol{\ell}_k^{\text{cand}} \end{bmatrix} \in \mathbb{R}^n$	full candidate activation vector
$\mathbf{z}_k = \sigma(W_z[\mathbf{h}_{k-1}; \mathbf{u}_k] + \mathbf{b}_z) \in (0,1)^n$	update gate vector
$\mathbf{h}_k = (\mathbf{1} - \mathbf{z}_k) \odot \mathbf{h}_{k-1} + \mathbf{z}_k \odot \tilde{\mathbf{h}}_k \in \mathbb{R}^n$	hidden state vector at time step k
$\mathbf{x}_k \in \mathbb{R}^m$	model output at step k
$W_z \in \mathbb{R}^{n \times (n+d)}$	update-gate weight matrix (L)
$\mathbf{b}_z \in \mathbb{R}^n$	update-gate bias vector (L)
$W_{\text{lat}} \in \mathbb{R}^{n_\ell \times (n+d)}$	latent linear dynamics matrix (L)
$\mathbf{b}_{\text{lat}} \in \mathbb{R}^{n_\ell}$	latent dynamics bias vector (L)
$W_{\text{out}} \in \mathbb{R}^{m \times n}$	output weight matrix (L)
$\mathbf{b}_{\text{out}} \in \mathbb{R}^m$	output bias vector (L)

The physics prior $f_{\text{phy}}(\boldsymbol{\xi}, \mathbf{u}; \boldsymbol{\theta}_{\text{phy}})$ is assumed to be differentiable with respect to both the physical state and its parameters $\boldsymbol{\theta}_{\text{phy}}$, which define the parametric form of the known physical model. These parameters may be fixed or learned from data.

PhyGRU does not include a reset gate, resetting the physical state prior to integration would introduce unphysical discontinuities. The update gate alone suffices to adaptively blend the previous state with the physics-biased candidate while preserving physical consistency. For clarity and reproducibility, the PyTorch implementation is reported in Appendix.

Unlike ODE–GRU models like the ones proposed in [12], which reinterpret the GRU dynamics as a continuous–time system and evolve the hidden state by integrating a learned vector field, the proposed PhyGRU preserves the discrete GRU recurrence and embeds an explicit parametric physical integrator inside the candidate state update. Specifically, PhyGRU advances a physics–based state using a fixed time step, and combines it with memory through the standard GRU update gate. Consequently, ODE–GRU learns a fully neural continuous–time hidden dynamics via an ODE solver, whereas PhyGRU enforces explicit, interpretable physics while retaining discrete–time recurrence and gate semantics.

3.3 Physics Candidate

The physics prior f_{phy} defines the physical vector field

$$\dot{\boldsymbol{\xi}} = f_{\text{phy}}(\boldsymbol{\xi}, \mathbf{u}; \boldsymbol{\theta}_{\text{phy}}),$$

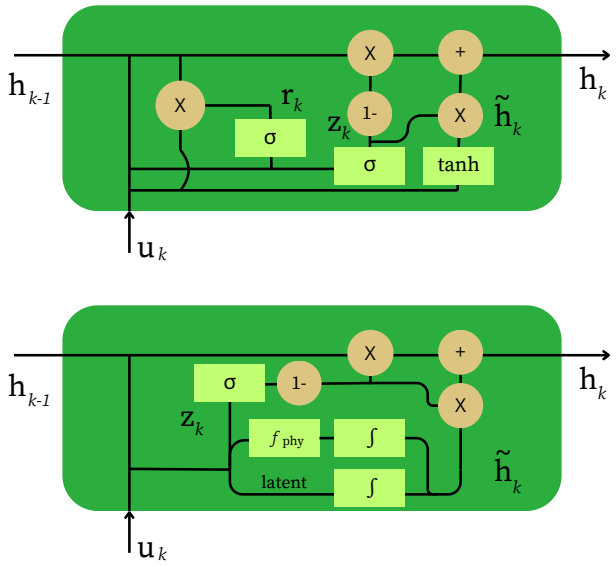

Figure 1: Top scheme standard full gated GRU cell, bottom scheme PhyGRU cell

with $\dot{\boldsymbol{\xi}} \in \mathbb{R}^{n_\xi}$. The physics part of the candidate is computed by one-step explicit Euler:

$$\boldsymbol{\xi}_k^{\text{cand}} = \boldsymbol{\xi}_{k-1} + \Delta t\, f_{\text{phy}}(\boldsymbol{\xi}_{k-1}, \mathbf{u}_k; \boldsymbol{\theta}_{\text{phy}}), \qquad \Delta t > 0.$$

Explicit Euler is adopted for two pragmatic reasons: (i) computational simplicity and low overhead (a single evaluation of the parametric vector field per step), and (ii) compatibility with the PhyGRU design, in which the physics integrator provides a lightweight, structurally meaningful candidate that is subsequently corrected by learned recurrent transformations. Explicit Euler is first-order accurate and, for sufficiently small Δt, yields a consistent forward propagation of the physics prior; however, it may suffer from stability or accuracy issues on stiff dynamics or when large time steps are used. The formulation intentionally leaves the integrator modular: users requiring higher numerical fidelity may replace explicit Euler with higher-order (e.g. RK4) or implicit schemes, at the expected cost of extra vector-field evaluations and implementation complexity. In such cases, however, the increased computational cost and architectural complexity should be carefully weighed against the expected benefits, given that the physical state is not directly exposed as the final model output.

3.4 Latent Dynamics

The latent channel is a simple linear integrator that corrects residual dynamics:

$$\dot{\boldsymbol{\ell}}_{k-1} = W_{\text{lat}} \begin{bmatrix} \mathbf{h}_{k-1} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_{\text{lat}}, \qquad \boldsymbol{\ell}_k^{\text{cand}} = \boldsymbol{\ell}_{k-1} + \Delta t\, \dot{\boldsymbol{\ell}}_{k-1},$$

with $W_{\text{lat}} \in \mathbb{R}^{n_\ell \times (n+d)}$ and $\mathbf{b}_{\text{lat}} \in \mathbb{R}^{n_\ell}$. Despite its simplicity, this linear latent integrator allows the model to account for residual dynamics not captured by the known physics, providing a flexible

correction pathway while keeping the physical state interpretable. The choice of linearity was made to remain consistent with minimal complexity; however, this is not a constraint for potential future applications, as it can easily be replaced with any nonlinearity. This design choice aims to limit internal model complexity and to prevent the latent channel from overshadowing the contribution of the physics component. By constraining the expressive power of the latent dynamics, the model encourages the physical prior to remain the dominant source of structured temporal evolution, while the latent state acts as a minimal corrective mechanism. From a practical standpoint, the framework does not preclude the use of more expressive latent dynamics, and users may adapt the latent structure to their specific system or application. Such extensions, however, should be considered carefully, as increasing latent expressivity may reduce the interpretability of the physics pathway and shift the model toward a predominantly data-driven regime.

3.5 Gated Reset

To preserve physical continuity, PhyGRU omits the GRU reset gate. An optional variant ($\text{PhyGRU}_{\text{rg}}$) applies a learned reset only to the latent subvector:

$$\mathbf{r}_k^\ell = \sigma\left(W_r^\ell \begin{bmatrix} \mathbf{h}_{k-1} \\ \mathbf{u}_k \end{bmatrix} + \mathbf{b}_r^\ell\right) \in (0,1)^{n_\ell}, \qquad \widetilde{\boldsymbol{\ell}}_{k-1} = \mathbf{r}_k^\ell \odot \boldsymbol{\ell}_{k-1},$$

with $W_r^\ell \in \mathbb{R}^{n_\ell \times (n+d)}$. This variant is reported only for ablation.

In support of the proposed choice to omit the reset gate in PhyGRU, several studies in the literature have evaluated GRU variants with pruned gates or components, as in [22, 23], some showing that in certain cases these variants outperform the standard GRU [21].

3.6 Time Complexity Evaluation

Consider a standard GRU with hidden-state dimension n and input dimension d. At each step, three affine transformations (update gate, reset gate, and candidate state) act on the concatenated vector $[\mathbf{h}_{k-1}; \mathbf{u}_k] \in \mathbb{R}^{n+d}$. Each transformation has cost $\mathcal{O}(n(n+d))$, yielding a per-step complexity

$$\mathcal{O}(3\,n(n+d))\,,$$

with elementwise nonlinearities contributing lower-order terms.

In PhyGRU, the reset gate is removed and a single update gate is evaluated, with cost $\mathcal{O}(n(n+d))$. The physical state $\boldsymbol{\xi}_k \in \mathbb{R}^{n_\xi}$ is propagated via explicit Euler integration, requiring one evaluation of the physics prior and $\mathcal{O}(n_\xi)$ arithmetic operations. The latent state $\boldsymbol{\ell}_k \in \mathbb{R}^{n_\ell}$ is advanced through a single linear map acting on $[\mathbf{h}_{k-1}; \mathbf{u}_k]$, with cost $\mathcal{O}(n_\ell(n+d))$. Elementwise blending through the update gate contributes an additional $\mathcal{O}(n)$ term.

The resulting per-step time complexity of PhyGRU is therefore

$$\mathcal{O}(n(n+d) + n_\ell(n+d) + n_\xi)\,.$$

Both GRU and PhyGRU exhibit comparable asymptotic scaling in the state and input dimensions. PhyGRU incurs a slightly larger constant factor due to explicit physical state propagation.

3.7 Incremental Stability Evaluation

Recall the PhyGRU state decomposition

$$\mathbf{h}_k = \begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\ell}_k \end{bmatrix}, \qquad \boldsymbol{\xi}_k \in \mathbb{R}^{n_\xi}, \quad \boldsymbol{\ell}_k \in \mathbb{R}^{n_\ell},$$

and the discrete-time update

$$\mathbf{h}_{k+1} = (\mathbf{1} - \mathbf{z}_k(\mathbf{h}_k, \mathbf{u}_k)) \odot \mathbf{h}_k + \mathbf{z}_k(\mathbf{h}_k, \mathbf{u}_k) \odot \tilde{\mathbf{h}}_k, \quad \tilde{\mathbf{h}}_k = \begin{bmatrix} \boldsymbol{\xi}_k + \Delta t\, f_{\text{phy}}(\boldsymbol{\xi}_k, \mathbf{u}_k; \boldsymbol{\theta}_{\text{phy}}) \\ \boldsymbol{\ell}_k + \Delta t\big(W_{\text{lat}}[\mathbf{h}_k; \mathbf{u}_k] + \mathbf{b}_{\text{lat}}\big) \end{bmatrix}, \tag{14}$$

where $\Delta t > 0$ is the integration step. Let $\mathbf{h}_0, \mathbf{h}'_0$ be two initial states evolving under the same input sequence $\{\mathbf{u}_k\}$. Assume that, for all admissible states and inputs,

1. the physics prior is Lipschitz continuous, uniformly in the parameters,

$$\|f_{\text{phy}}(\boldsymbol{\xi}, \mathbf{u}; \boldsymbol{\theta}_{\text{phy}}) - f_{\text{phy}}(\boldsymbol{\eta}, \mathbf{u}; \boldsymbol{\theta}_{\text{phy}})\| \le L_p \|\boldsymbol{\xi} - \boldsymbol{\eta}\|;$$

2. the latent linear dynamics satisfy

$$\|W_{\text{lat}}[\mathbf{h}; \mathbf{u}] - W_{\text{lat}}[\mathbf{h}'; \mathbf{u}]\| \le L_\ell \|\mathbf{h} - \mathbf{h}'\|;$$

3. the update gate is componentwise bounded and Lipschitz in the state,

$$\mathbf{z}(\mathbf{h}, \mathbf{u}) \in [\underline{z}, \overline{z}]^n, \qquad 0 < \underline{z} \le \overline{z} < 1,$$

   and there exists $L_z \ge 0$ such that for all admissible $\mathbf{h}, \mathbf{h}', \mathbf{u}$

$$\|\mathbf{z}(\mathbf{h}, \mathbf{u}) - \mathbf{z}(\mathbf{h}', \mathbf{u})\| \le L_z \|\mathbf{h} - \mathbf{h}'\|.$$

4. the candidate increment is uniformly bounded: there exists $B \ge 0$ with

$$\|\tilde{\mathbf{h}} - \mathbf{h}\| \le \Delta t\, B \qquad \text{for all admissible } \mathbf{h}, \mathbf{u}.$$

Proposition 3.1. *For two PhyGRU trajectories $\mathbf{h}_k, \mathbf{h}'_k$ driven by the same inputs, the one-step difference satisfies*

$$\|\mathbf{h}_{k+1} - \mathbf{h}'_{k+1}\| \le \gamma\, \|\mathbf{h}_k - \mathbf{h}'_k\|, \qquad \gamma = (1 - \underline{z}) + \overline{z}\big(1 + \Delta t \max(L_p, L_\ell)\big) + L_z \Delta t\, B.$$

Hence the evolution map is globally Lipschitz in the initial condition (uniformly in the input) and the perturbation grows at most geometrically with factor γ. In particular, if $\gamma < 1$ the PhyGRU dynamics are incrementally exponentially stable.

Proof. Subtracting the two updates in (14) and rearranging terms yields

$$\begin{aligned} \mathbf{h}_{k+1} - \mathbf{h}'_{k+1} = (\mathbf{1} - \mathbf{z}_k) \odot (\mathbf{h}_k - \mathbf{h}'_k) + \mathbf{z}_k \odot (\tilde{\mathbf{h}}_k - \tilde{\mathbf{h}}'_k) \\ + (\mathbf{z}_k - \mathbf{z}'_k) \odot (\tilde{\mathbf{h}}'_k - \mathbf{h}'_k), \end{aligned}$$

where $\mathbf{z}_k = \mathbf{z}(\mathbf{h}_k, \mathbf{u}_k)$ and $\mathbf{z}'_k = \mathbf{z}(\mathbf{h}'_k, \mathbf{u}_k)$. Taking norms and applying the triangle inequality gives

$$\begin{aligned} \|\mathbf{h}_{k+1} - \mathbf{h}'_{k+1}\| \le \|(\mathbf{1} - \mathbf{z}_k) \odot (\mathbf{h}_k - \mathbf{h}'_k)\| + \|\mathbf{z}_k \odot (\tilde{\mathbf{h}}_k - \tilde{\mathbf{h}}'_k)\| \\ + \|\mathbf{z}_k - \mathbf{z}'_k\|\, \|\tilde{\mathbf{h}}'_k - \mathbf{h}'_k\|. \end{aligned}$$

Using the componentwise bounds on the gates,

$$\|(\mathbf{1} - \mathbf{z}_k) \odot (\mathbf{h}_k - \mathbf{h}'_k)\| \le (1 - \underline{z}) \|\mathbf{h}_k - \mathbf{h}'_k\|,$$

and

$$\|\mathbf{z}_k \odot (\tilde{\mathbf{h}}_k - \tilde{\mathbf{h}}'_k)\| \le \overline{z}\|\tilde{\mathbf{h}}_k - \tilde{\mathbf{h}}'_k\|.$$

From the Lipschitz assumptions on f_{phy} and W_{lat} we get

$$\|\tilde{\mathbf{h}}_k - \tilde{\mathbf{h}}'_k\| \le \big(1 + \Delta t \max(L_p, L_\ell)\big)\|\mathbf{h}_k - \mathbf{h}'_k\|.$$

By the Lipschitz property of the gate and the boundedness of the candidate increment,

$$\|\mathbf{z}_k - \mathbf{z}'_k\|\,\|\tilde{\mathbf{h}}'_k - \mathbf{h}'_k\| \le L_z\|\mathbf{h}_k - \mathbf{h}'_k\| \cdot (\Delta t\, B).$$

Combining the three estimates yields the stated bound with

$$\gamma = (1 - \underline{z}) + \overline{z}(1 + \Delta t \max(L_p, L_\ell)) + L_z \Delta t B$$

. □

Although γ may satisfy $\gamma \ge 1$ under general Lipschitz assumptions, contraction ($\gamma < 1$) can arise under additional dissipativity conditions on the candidate increment. In particular, suppose that the physical and latent increments are jointly incrementally contractive, i.e.,

$$\|\tilde{\mathbf{h}}(\mathbf{h}, \mathbf{u}) - \tilde{\mathbf{h}}(\mathbf{h}', \mathbf{u})\| \le (1 - \beta)\|\mathbf{h} - \mathbf{h}'\|, \qquad \beta > 0,$$

uniformly in $\mathbf{u}$. Then the update satisfies

$$\|\mathbf{h}_{k+1} - \mathbf{h}'_{k+1}\| \le \big[(1 - \underline{z}) + \overline{z}(1 - \beta)\big]\|\mathbf{h}_k - \mathbf{h}'_k\|.$$

Hence, if $\beta > 0$ is sufficiently large and $0 < \underline{z} \le \overline{z} < 1$, the factor

$$\gamma = (1 - \underline{z}) + \overline{z}(1 - \beta)$$

is strictly smaller than one, yielding incremental exponential stability. Such contraction may occur, for instance, when the physical prior is incrementally dissipative (e.g., damped mechanical dynamics) and the latent linear block is stable, possibly enforced through spectral constraints or sufficiently small integration step Δt.

The above proposition provides a guarantee of well-posedness and continuous dependence of the trajectories on the initial conditions, for a fixed input sequence: perturbations in the initial state grow at most geometrically with factor γ. It is important to note that, under the general assumptions stated above, γ may satisfy $\gamma \ge 1$, and therefore contraction is not automatically ensured. Knowing these aspects is especially useful in applications like trajectory tracking, nonlinear control, and observer design, where consistent performance across varying operating conditions is critical.

4 Methodology

In this section, it is clarified how the tests were conducted, which models were compared, and the metrics used to evaluate the results, in order to provide the reader with a clear overview for a proper interpretation of the findings.

4.1 Data Generation

Three synthetic controlled dynamical systems are generated via numerical integration of ordinary differential equations with time step $\Delta t = 0.01$ and trajectory length $T = 6000$. For each system, two data-generation settings are considered: one with time-invariant parameters and one with smoothly time-varying parameters. This is done in order to simulate a limiting use case in which only a small amount of data is available, just sufficient to perform training, validation, and testing, while a reasonable level of knowledge of the target physical system is available. This is a fairly common scenario, and it is precisely in this context that the effectiveness of the modification introduced in PhyGRU is demonstrated.

For each system, one long trajectory is generated for training, validation, and testing. The excitation signals are bounded, non-stationary inputs obtained by combining amplitude- and frequency-modulated sinusoids followed by a $\tanh(\cdot)$ nonlinearity. All inputs share the same parametric structure, while different parameter sets are used for the three splits. States and inputs are normalized using the absolute maximum values computed on the training trajectory only, and the same scaling is applied to validation and test data. This procedure reflects a limited-data scenario typical in practical applications, where only one representative trajectory per split may be available.

$$u_t = \tanh\Big((a_0 + a_1 t)\big[(b_0 + b_1 t)\sin((\omega_{10} + \omega_{11} t)t) + (c_0 + c_1 t)\sin((\omega_{20} + \omega_{21} t)t)\big]\Big).$$

The numerical values of the parameters used to generate the training, validation, and test input signals are summarized in Table 1.

Parameter	Train	Validation	Test
a_0	0.30	0.40	0.50
a_1	$-5{\cdot}10^{-5}$	$-5{\cdot}10^{-5}$	$-5{\cdot}10^{-5}$
b_0	0.25	0.35	0.30
b_1	$-1{\cdot}10^{-3}$	$-3{\cdot}10^{-3}$	$-2{\cdot}10^{-3}$
c_0	0.10	0.15	0.15
c_1	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$1{\cdot}10^{-3}$
ω_{10}	$7{\cdot}10^{-5}$	$9{\cdot}10^{-5}$	$5{\cdot}10^{-4}$
ω_{11}	$1{\cdot}10^{-7}$	$2{\cdot}10^{-7}$	$5{\cdot}10^{-7}$
ω_{20}	$1{\cdot}10^{-6}$	$3{\cdot}10^{-6}$	$4{\cdot}10^{-6}$
ω_{21}	$-1{\cdot}10^{-6}$	$-1{\cdot}10^{-6}$	$-1{\cdot}10^{-6}$

Table 1: Input signal parameters for the three dataset splits.

All systems are written in the general form $\dot{\mathbf{s}} = f(\mathbf{s}, u; \boldsymbol{\theta}(t))$. In the time-varying setting, the coefficients $\boldsymbol{\theta}(t)$ evolve smoothly over time according to polynomial or affine functions, ensuring gradual parameter drift. In particular the systems are defined as follow: System 1 is a second-order linear system,

$$\ddot{x} = u - b(t)\dot{x} - c(t)x,$$

with linearly and quadratically varying damping and stiffness. System 2 extends this model by adding a nonlinear term,

$$\ddot{x} = u - b(t)\dot{x} - c(t)x + d(t)\tanh(x\dot{x}),$$

while System 3 is a third-order nonlinear system,

$$\dddot{x} = u - a(t)\ddot{x} - b(t)\tanh(\dot{x}) - c(t)\tanh(x).$$

The three systems were defined so as to evaluate both the case in which the model provided to PhyGRU is correct and the cases in which the provided model lacks complexities that instead emerge from the data, thereby highlighting more or less favorable operating conditions.

For each system, a time-invariant (TI) version is obtained by fixing the coefficients to constant values, while a time-varying (TV) version is obtained by allowing the same coefficients to evolve smoothly over time. The constant parameters used in the TI configuration are reported in Table 2, while the corresponding time-dependent parameterizations for the TV configuration are summarized in Table 3. For clarity, in Figures 2 and 3 the systems, in their two respective configurations, are shown as a function of time.

System	a	b	c	d
S1	–	0.5	0.2	–
S2	–	0.5	0.2	1.0
S3	3.0	2.0	0.1	–

Table 2: Time-invariant (TI) system parameters.

System	$a(t)$	$b(t)$	$c(t)$	$d(t)$
S1	–	$0.5 + 9{\cdot}10^{-5}t$	$0.2 + 1{\cdot}10^{-7}t^2$	–
S2	–	$0.5 + 9{\cdot}10^{-5}t$	$0.2 + 1{\cdot}10^{-7}t^2$	$1 + 9{\cdot}10^{-5}t$
S3	$3 - 2{\cdot}10^{-4}t$	$2 + 9{\cdot}10^{-4}t$	$0.1 + 3{\cdot}10^{-7}t^2$	–

Table 3: Time-Varying (TV) system parameters.

The six resulting systems (three structures, each in TI and TV configurations) are designed to progressively stress the models under increasing mismatch between the assumed physical prior and the true data-generating dynamics, with particular emphasis on evaluating the robustness of the proposed PhyGRU architecture under misdefined physics priors.

4.2 Models Compared

Three model classes are evaluated. A standard GRU is trained in a sequence-to-sequence setting using only the input signal $u(t)$, with hidden sizes in $\{1, 2, 4, 8, 32\}$. The GRU_{obs} model uses the same architecture but augments the input with observed past states and finite-difference derivatives, passed through $\tanh(\cdot)$ for numerical stability. Finally, a PhyGRU model embeds a second-order prior and optionally augments the physical state with a low-dimensional latent vector, ranging in $\{0, 1, 2, 3\}$. The physical prior used by PhyGRU follows:

$$\ddot{\xi} = \frac{u - b\dot{\xi} - c\xi}{a},$$

with learnable parameters initialized as $a = 0.5$, $b = 0.6$, and $c = 0.7$. In addition to these architectures, a further exploratory test is conducted to investigate, at an early stage, the behavior of stacked recurrent models combining standard GRU and PhyGRU blocks. In particular, two sequential configurations are evaluated: a GRU–GRU cascade and a PhyGRU–GRU cascade, where the output of the first recurrent model is fed as input to a second GRU. This experiment is intended to provide preliminary insight into the potential benefits of introducing physics-informed recurrence within deeper recurrent stacks, while preserving the original training protocol and evaluation setup.

The main focus of this study is to evaluate the effect of introducing a physics-informed ODE prior directly within a standard GRU architecture. As such, the benchmarking effort is performed relative to classical GRU baselines (with and without observed states) to isolate the impact of the proposed architectural modification. Direct comparisons against Neural ODEs, ODE-RNNs, or other continuous-time recurrent models are not performed, as these constitute fully distinct architectures with different training protocols and objectives. The aim is to demonstrate that a minimal GRU modification can incorporate physical priors effectively, without requiring a complete redesign of the recurrent network.

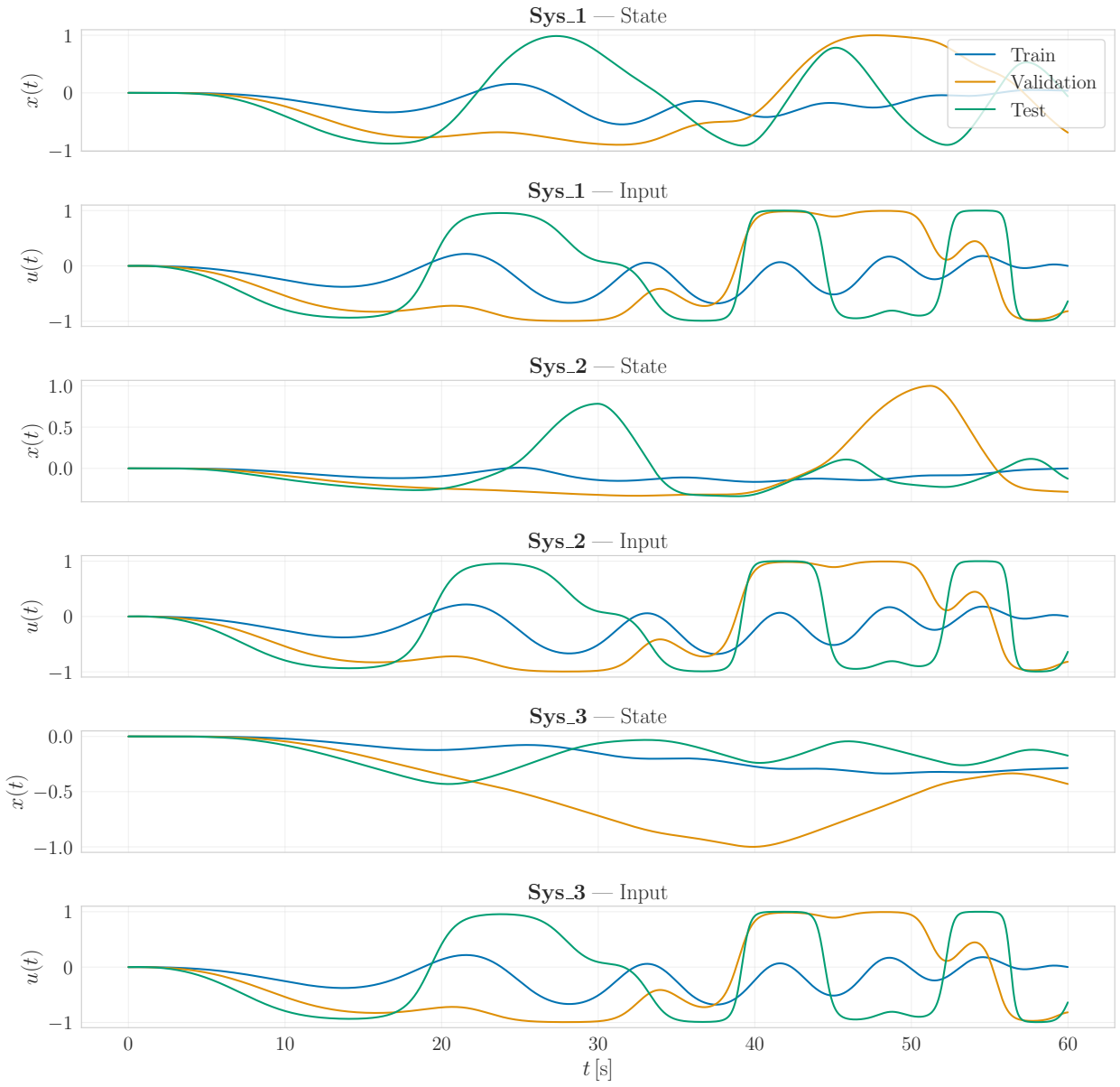

Figure 2: Dataset trajectories and inputs over the considered systems, in time invariant parameters

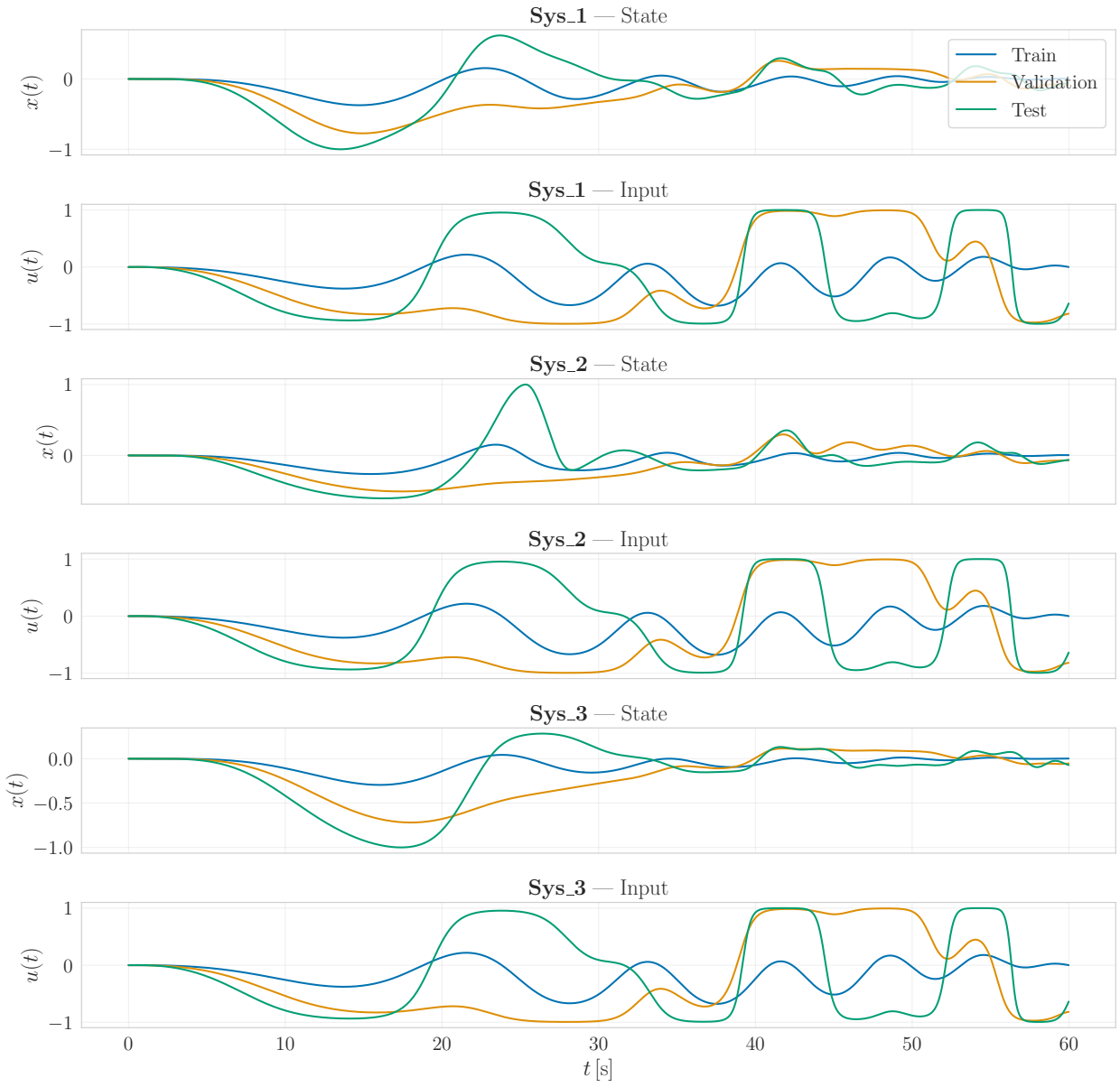

Figure 3: Dataset trajectories and inputs over the considered systems, in time varying parameters

4.3 Training Protocol

All models are trained for 150 epochs using the Adam optimizer with learning rate 5×10^{-3} and mean squared error loss. Training is performed on single trajectories, in order to follow the case proposed with low data availability and limited physical knowledge of the system. Validation and testing are always carried out in closed-loop, and the model achieving the lowest validation error is selected for final evaluation. For GRU_{obs}, training combines a standard teacher-forced one-step prediction loss with an explicit closed-loop rollout loss. In details, for GRU_{obs}, the training loss combines two components. First, a teacher-forced one-step prediction loss is computed, in which the model is provided with ground-truth past states to predict the next state. Second, a self-fed closed-loop loss is computed by rolling out the model predictions over the entire trajectory: at each

step, the model's previous predictions for x, $\dot{x}$, and $\ddot{x}$ are used to construct the augmented input for the next time step. The predicted trajectory is then compared to the ground truth using mean squared error. The total training loss is obtained as the simple average of the teacher-forced and closed-loop terms, which encourages accurate one-step predictions while reducing exposure bias and improving long-horizon stability.

During validation and testing, the model is always evaluated in a fully self-fed, closed-loop manner, mimicking the deployment scenario. Derivatives of the predicted states are computed at each step via finite differences and fed back as inputs for the subsequent prediction. The model achieving the lowest validation error is selected for final evaluation. This combination of augmented inputs and dual loss terms allows GRU_{obs} to effectively leverage past state information while remaining stable during long-horizon prediction.

Furthermore, in the tests conducted under the TI setup, the $\text{PhyGRU}_{\text{rg}}$ architecture was also evaluated, which is essentially PhyGRU with a reset gate applied only to the latent state, following the same approach used for the PhyGRU and GRU models.

Summarizing the main hyperparameters are as follows: $\Delta t = 0.01$, $T = 6000$, 150 training epochs, batch size= 1 Adam optimizer with learning rate 5×10^{-3}, MSE loss, GRU hidden sizes $\{1, 2, 4, 8, 32\}$, PhyGRU latent dimensions $\{0, 1, 2, 3\}$, and fixed random seed 0.

4.4 Evaluation Metrics

Performance is assessed using Mean Squared Error (MSE) on validation and test sets, Spearman rank correlation between predicted and true trajectories and inference time per time step, to have a clear comparison plots are also reported for best validation run and test set.

5 Results

Results extracted from the experiments are summarized in Figures and in the Tables below. Each table reports: Best validation MSE, test MSE, number of parameters, inference time per step (only for the first system on Intel(R) Xeon(R) CPU @ 2.20 GHz), and Spearman rank correlation on validation and test trajectories. The section is divided in three macro blocks where the first one report the results with time invariant parameters, the second one the time varying versions of the systems and the last report the results obtained from the stacked architecture over the different sizes on the time varying parameters systems. To better assess the proposed architecture and provide the reader with a clear and comprehensive perspective, the first section includes additional complementary studies. Specifically, inference-time analyses are reported, comparing PhyGRU with a standard GRU reimplemented, without using framework available version, in order to ensure a fair comparison against the same architectural formulation, rather than against highly optimized implementations available in common machine learning frameworks.

Further investigations are presented by introducing a reset gate into PhyGRU, enabling a direct performance comparison with the reset-free version. In addition, specific analyses of the model internals are conducted to gain insight into the mechanisms driving the predictions. These include the temporal evolution of the update gate activation, the contributions of the different prediction candidates, and an evaluation of whether the model is able to accurately identify the parameters of the provided physical prior.

5.1 Results with Time Invariant Parameters

The results obtained for the considered models on the three different systems under time-invariant conditions are reported below. In particular, Tables 4, 7 and 8 present the numerical results, while Figures 5, 6, show the corresponding time-series plots, with the reference trajectory indicated by a dashed line.

In support of these tests, the models were also compared in terms of inference time per sample. To this end, although all other experiments in this work were conducted using the standard implementation available in PyTorch 2.9.0, For this specific comparison, a different approach was adopted to ensure fairness. In particular, to avoid comparing a highly optimized library implementation against PhyGRU, which is implemented with a research-oriented focus rather than performance optimization, the standard GRU was reimplemented, thereby making the inference-time comparison more meaningful. To obtain robust measurements, the models were evaluated over 90,000 samples, and the results of this analysis are reported in Table 5. Although not reported in the table to avoid potential confusion, for the sake of transparency it is noted that the results obtained using the PyTorch implementation range on average from 0.035 to 0.061 ms per sample on the same hardware and under the same system conditions.

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	1.007e+0/1.020e+0	0.802/0.261
GRU	hidden=2	33	7.892e−1/9.022e−1	0.893/0.481
GRU	hidden=4	89	8.073e−1/9.524e−1	0.896/0.483
GRU	hidden=8	273	4.756e−1/4.801e−1	0.904/0.907
GRU	hidden=32	3393	3.543e−1/3.506e−1	0.901/0.918
GRU_{obs}	hidden=1	23	8.945e−1/1.028e+0	0.783/0.282
GRU_{obs}	hidden=2	51	1.058e+0/1.023e+0	0.764/0.273
GRU_{obs}	hidden=4	125	4.925e−1/1.172e+0	0.732/0.266
GRU_{obs}	hidden=8	345	6.263e−1/1.033e+0	0.755/0.293
GRU_{obs}	hidden=32	3681	4.132e−1/1.048e+0	0.775/0.341
PhyGRU	latent=0	11	6.402e−2/1.700e−1	0.971/0.954
PhyGRU	latent=1	23	2.421e−2/2.554e−1	0.982/0.921
PhyGRU	latent=2	39	1.079e−2/4.370e−2	0.995/0.982
PhyGRU	latent=3	59	5.995e−1/2.432e−1	0.858/0.869
$PhyGRU_{rg}$	latent=1	23	2.159e−2/4.368e−2	0.991/0.985
$PhyGRU_{rg}$	latent=2	39	2.928e−2/1.917e−1	0.966/0.958
$PhyGRU_{rg}$	latent=3	59	1.210e−1/3.101e−1	0.964/0.886

Table 4: Results for the compared architectures, over System 1 in TI condition.

Model	Hidden / Latent	Parameters	Mean	Std
GRU	1	14	0.1463	0.0262
GRU	2	33	0.1441	0.0236
GRU	4	89	0.1413	0.0201
GRU	8	273	0.1407	0.0202
GRU	32	3393	0.1401	0.0201
PhyGRU	0	11	0.1507	0.0206
PhyGRU	1	23	0.2016	0.0335
PhyGRU	2	39	0.1969	0.0321
PhyGRU	3	59	0.2028	0.0307
$\text{PhyGRU}_{\text{rg}}$	0	11	0.1528	0.0216
$\text{PhyGRU}_{\text{rg}}$	1	28	0.2425	0.0269
$\text{PhyGRU}_{\text{rg}}$	2	51	0.2476	0.0532
$\text{PhyGRU}_{\text{rg}}$	3	80	0.2499	0.0343

Table 5: Inference time comparison between GRU, PhyGRU, and PhyGRU with reset gate ($\text{PhyGRU}_{\text{rg}}$). Mean and standard deviation are reported per time step, implemented manually. Mean and Std are expressed in ms.

To study the effect of the integration timestep on PhyGRU, all trajectories (training, validation, and test) are generated at a fixed resolution of 0.01 s for System 1 TI. These trajectories serve as the ground truth for training and evaluation. During training, two configurations of the PhyGRU model are explored with internal integration of the system dynamics using [0.01, 0.9] as timesteps and latent set to one, which may differ from the trajectory timestep. After training, the models are evaluated on test trajectories sampled at the same fixed resolution, while the model's internal timestep is set to [0.005, 0.01, 0.9]. This approach deliberately introduces a mismatch between the data sampling rate and the model's internal integration rate. By doing so, a systematic investigation can be conducted (1.) How the choice of training timestep affects learning, (2.) How errors propagate when the model timestep differs from the data timestep, (3.) The generalization of the learned model to different integration rates.

The setup used ensures that all observed effects are solely due to the timestep mismatch, without altering the original trajectory resolution. Numeric results are resumed in Table 6, trajectory plots over test set are shown in Figure 4. The value of 0.9 has been chosen based on the following study on physics prior provided to PhyGRU:

$$\dot{x}_1 = x_2, \qquad \dot{x}_2 = -\frac{b}{a}x_2 - \frac{c}{a}x_1,$$

with positive parameters $a, b, c > 0$. In the oscillatory regime, i.e. when

$$\frac{4c}{a} > \left(\frac{b}{a}\right)^2,$$

the system has a pair of complex conjugate eigenvalues with negative real part. For the scalar test equation $y' = \lambda y$, the explicit Euler method is stable if and only if $|1 + \Delta t\, \lambda| < 1$. Applying this condition to the dominant eigenvalue of the system yields a maximum admissible timestep for stability. In the oscillatory case, this critical timestep simplifies to

$$\Delta t_{\text{crit}} = \frac{b}{c}.$$

Using the prior parameter values $a = 0.5$, $b = 0.6$, and $c = 0.7$, the stability limit becomes

$$\Delta t_{\text{crit}} \approx 0.857 \text{ s}.$$

Hence, explicit Euler integration of the linearized system is stable for timesteps smaller than this value, while in practice significantly smaller timesteps are typically chosen to ensure adequate accuracy. The chosen value of 0.9 was deliberately selected to be slightly off the theoretical value, thereby placing PhyGRU under a stress condition. For the timestep sensitivity analysis, all experiments are conducted using 32-bit floating-point precision (float32). This choice is intentionally adopted to accentuate numerical truncation effects. The latter is responsible for the mismatch in the results with respect to those reported in Table 4, for the $\Delta t = 0.01$ in training and testing configuration, resulting in an unexpected slight performance improvement, which may be attributed to the implicit regularization induced in the model.

Train Δt	Test Δt	Test MSE	Spearman test
0.01	0.005	6.597e−1	0.653
0.90	0.005	1.210e−1	0.974
0.01	0.01	1.093e−2	0.993
0.90	0.01	3.281e−1	0.856
0.01	0.90	1.815e+0	0.275
0.90	0.90	1.692e+0	0.249

Table 6: Cross-evaluation of PhyGRU models trained with different internal integration timesteps.

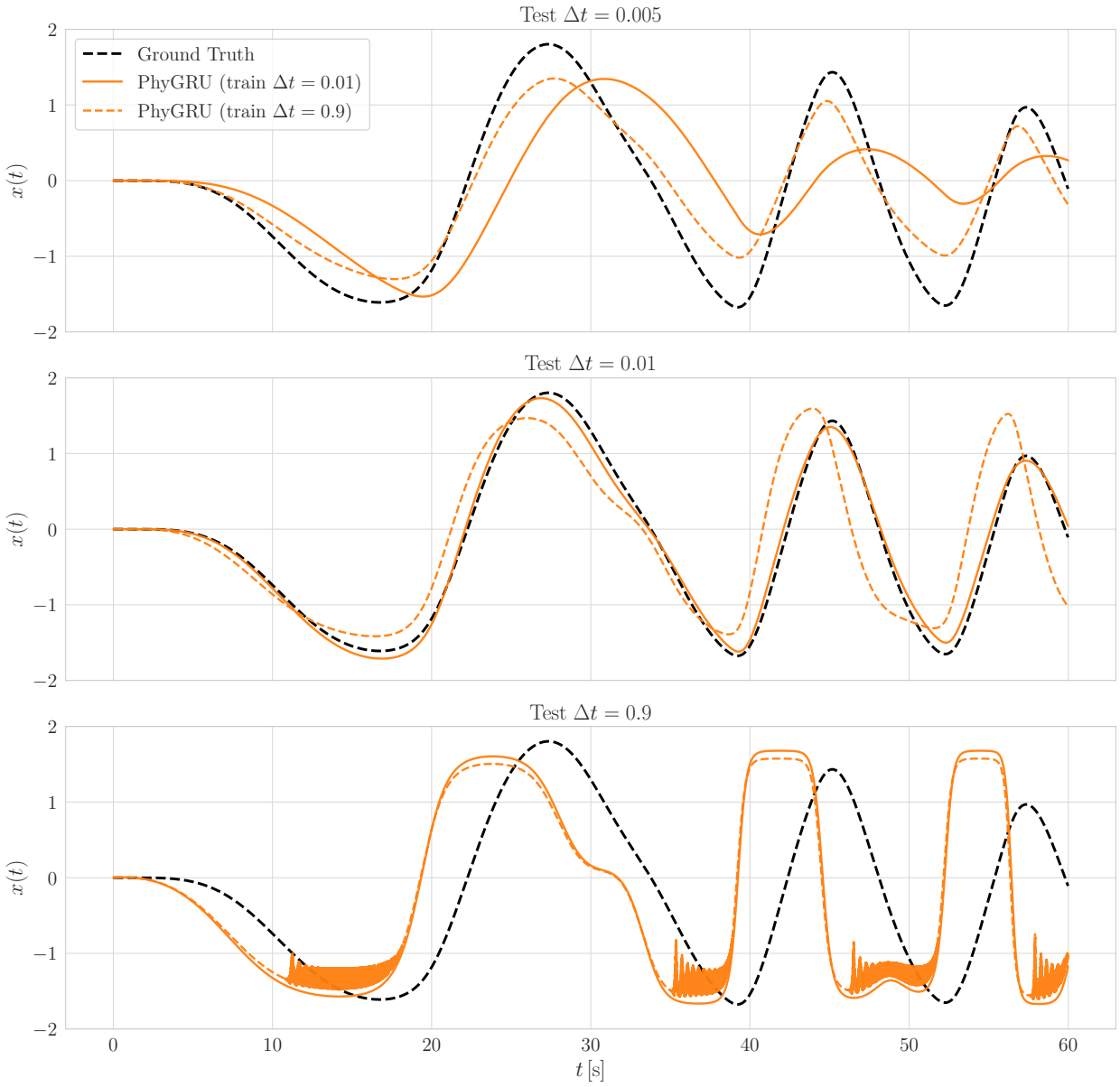

Figure 4: Timestep sensitivity on two PhyGRU_l1 over System 1 (TI) test set using different PhyGRU internal timestep in training and testing

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	4.413e+0/2.592e+0	0.685/0.221
GRU	hidden=2	33	4.130e+0/2.467e+0	0.804/0.512
GRU	hidden=4	89	3.294e+0/2.071e+0	0.909/0.558
GRU	hidden=8	273	3.477e+0/2.616e+0	0.688/0.229
GRU	hidden=32	3393	2.380e+0/1.471e+0	0.960/0.855
GRU_{obs}	hidden=1	23	4.474e+0/2.497e+0	0.705/0.316
GRU_{obs}	hidden=2	51	4.429e+0/2.709e+0	0.685/0.240
GRU_{obs}	hidden=4	125	4.225e+0/2.514e+0	0.666/0.233
GRU_{obs}	hidden=8	345	3.481e+0/2.749e+0	0.571/0.261
GRU_{obs}	hidden=32	3681	3.127e+0/2.425e+0	0.679/0.292
PhyGRU	latent=0	11	1.327e+0/1.344e+0	0.924/0.893
PhyGRU	latent=1	23	2.714e+0/1.200e+0	0.890/0.830
PhyGRU	latent=2	39	3.256e+0/4.682e+0	0.639/0.372
PhyGRU	latent=3	59	5.751e+0/2.153e+0	0.639/0.653
$PhyGRU_{rg}$	latent=1	23	1.776e+0/1.028e+0	0.878/0.835
$PhyGRU_{rg}$	latent=2	39	5.998e+0/3.427e+0	0.615/0.414
$PhyGRU_{rg}$	latent=3	59	1.702e+0/1.187e+0	0.919/0.863

Table 7: Results for the compared architectures, over System 2 in TI condition.

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	1.198e+0/2.784e−1	0.147/0.279
GRU	hidden=2	33	1.704e+0/1.176e−1	0.136/0.239
GRU	hidden=4	89	1.613e+0/9.887e−2	0.308/0.468
GRU	hidden=8	273	1.664e+0/1.888e−1	−0.145/ − 0.276
GRU	hidden=32	3393	6.989e−1/3.549e−1	0.813/0.614
GRU_{obs}	hidden=1	23	1.089e+0/3.767e−1	0.185/0.293
GRU_{obs}	hidden=2	51	1.079e+0/4.480e−1	0.077/0.140
GRU_{obs}	hidden=4	125	2.078e+0/1.595e−1	0.196/0.295
GRU_{obs}	hidden=8	345	1.311e+0/2.560e−1	0.101/0.202
GRU_{obs}	hidden=32	3681	1.127e+0/1.693e−1	0.667/0.597
PhyGRU	latent=0	11	2.549e−1/7.362e−1	0.919/0.805
PhyGRU	latent=1	23	2.885e−1/1.513e+0	0.943/0.575
PhyGRU	latent=2	39	2.330e−1/1.200e−1	0.961/0.732
PhyGRU	latent=3	59	3.613e−1/9.759e−1	0.930/0.630
$PhyGRU_{rg}$	latent=1	23	1.773e−1/4.006e−1	0.980/0.856
$PhyGRU_{rg}$	latent=2	39	5.866e+0/5.697e+0	0.022/0.146
$PhyGRU_{rg}$	latent=3	59	5.233e−1/3.686e+0	0.687/0.261

Table 8: Results for the compared architectures, over System 3 in TI condition.

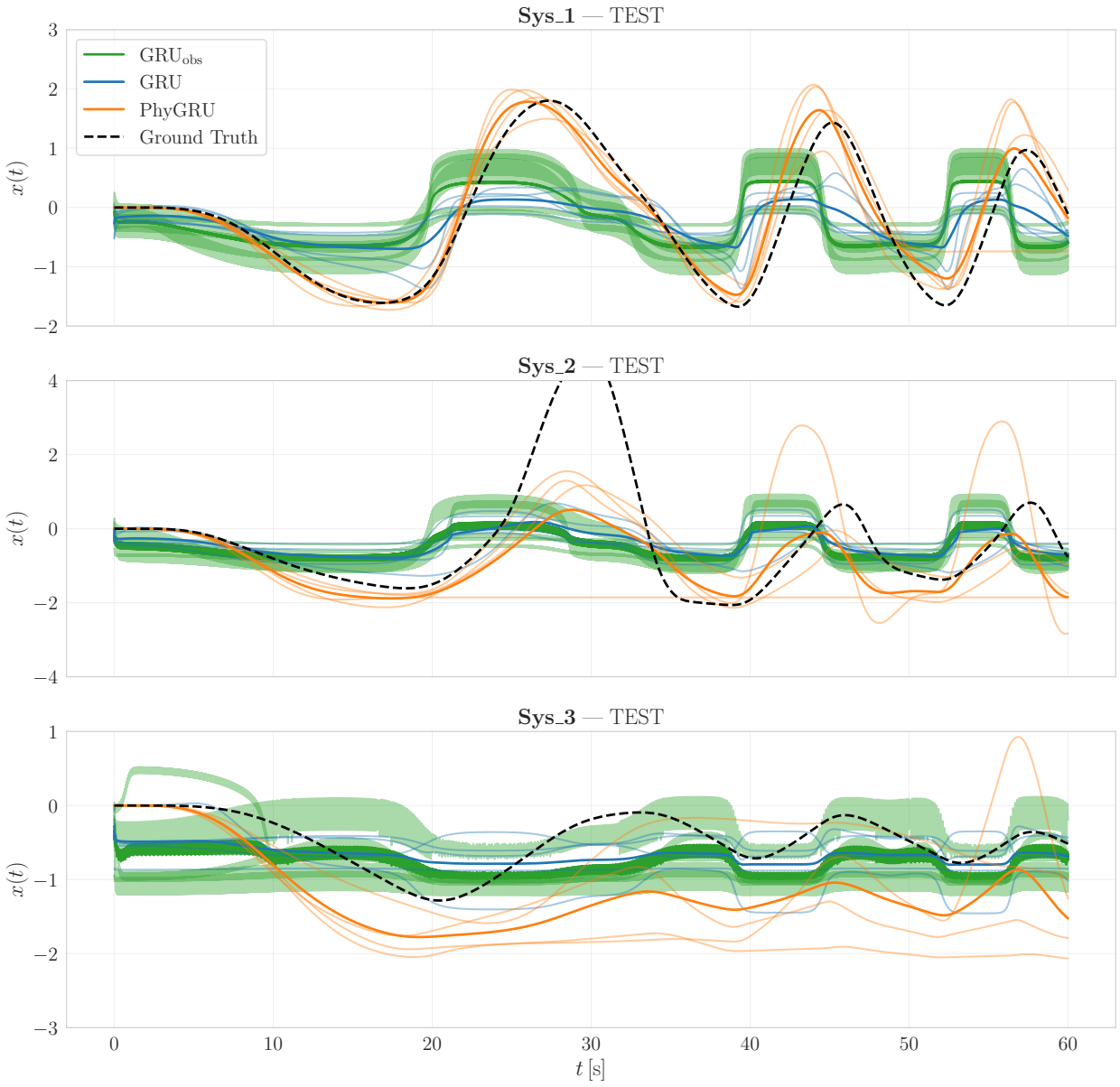

Figure 5: Test results trajectories over the considered systems, in time invariant parameters

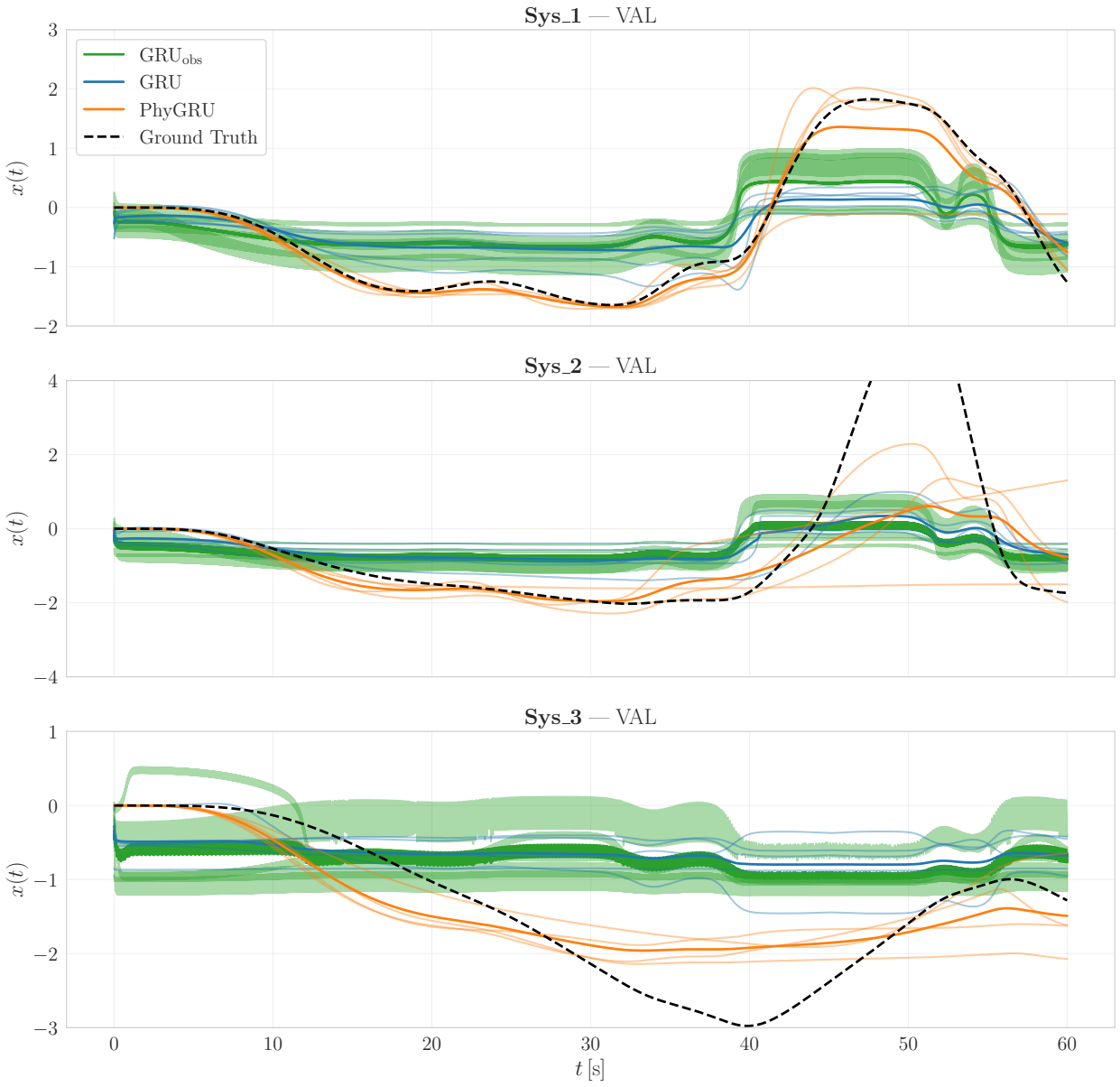

Figure 6: Validation results trajectories over the considered systems, in time invariant parameters

To assess the parameter identification performance on System 1, Table 9 reports the learned values after training, alongside the ground truth (GT) and the initial guess (IG) used at the start of training.

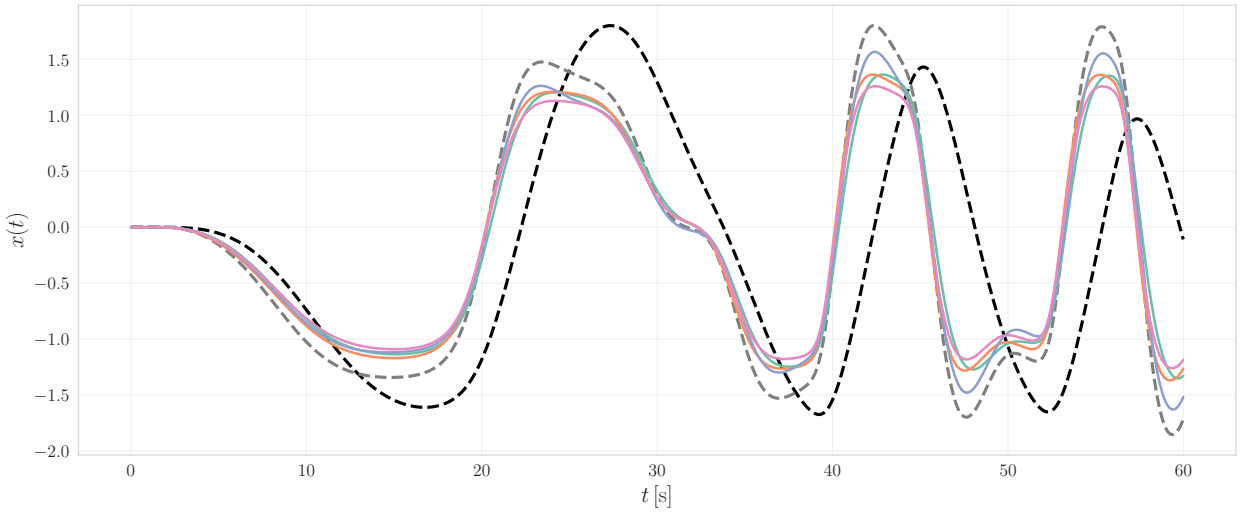

Figure 7: Identified systems using only the a, b and c parameters respect to ground truth (black) and initial guess (gray) trajectories for System 1 in time invariant configuration under test excitation

Model	a	b	c
GT	1.000	0.500	0.200
IG	0.500	0.600	0.700
l0	0.631	0.915	0.826
l1	0.382	0.735	0.800
l2	0.658	0.698	0.841
l3	0.436	0.832	0.858

Table 9: System 1 idenified parameters (a, b, c)

For better understanding, these identified systems are plotted in Figure 7 with respect to the target system. They differ from the PhyGRU prediction not only due to the effect of the latent, which directly influences the prediction in cases where a latent is present, but also in the case without a latent, given the presence of the update gate.

For the PhyGRU case applied at system one and three with a latent size equal to one and two, Figures 8, 9 are provided to clarify, through an application example, how the physics candidate, latent candidate, and gate interact during operation.

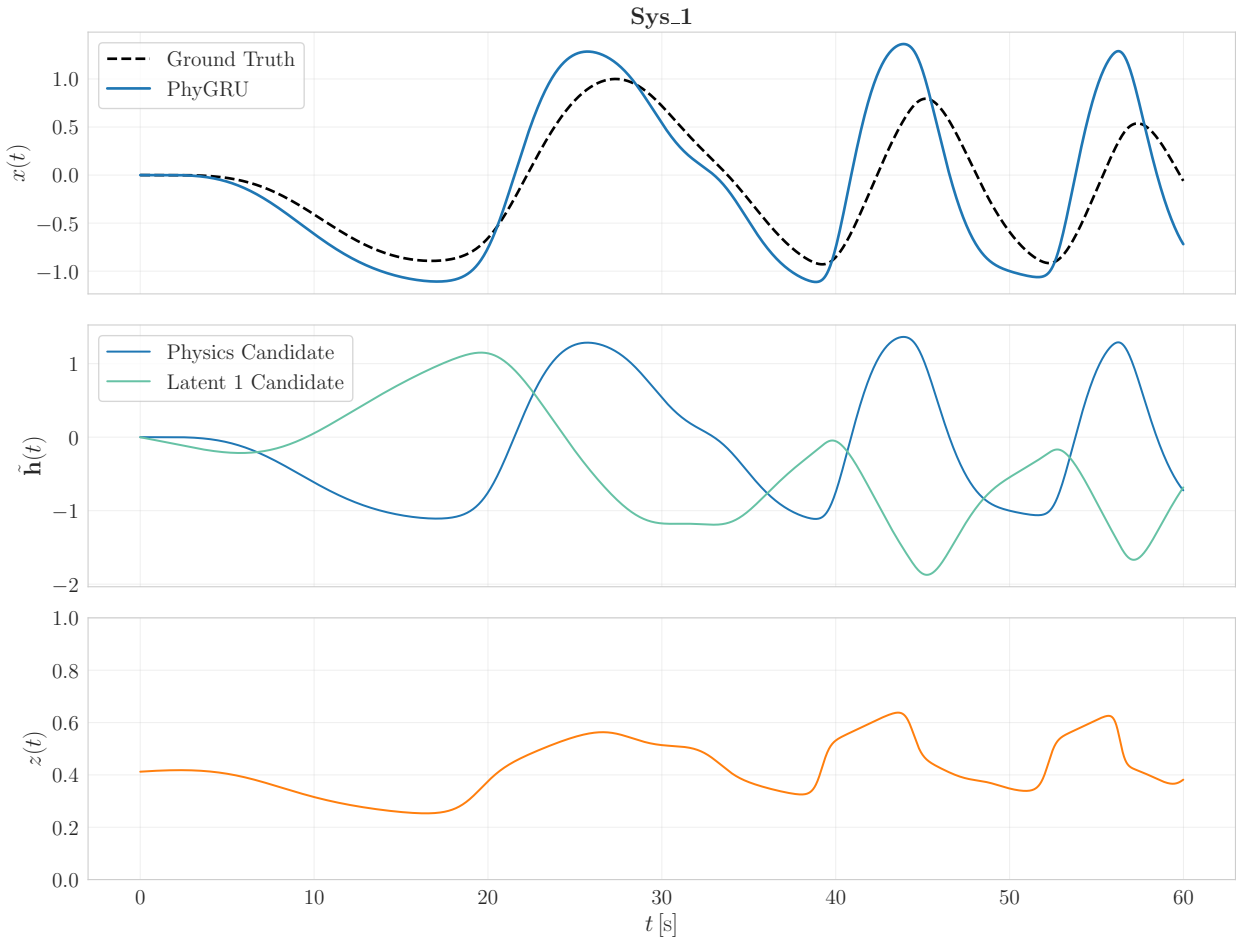

Figure 8: Physics and latent candidates values over time and gate activation, specific for ξ component, for the case with latent size equal to one, for the System 1 in time invariant parameters

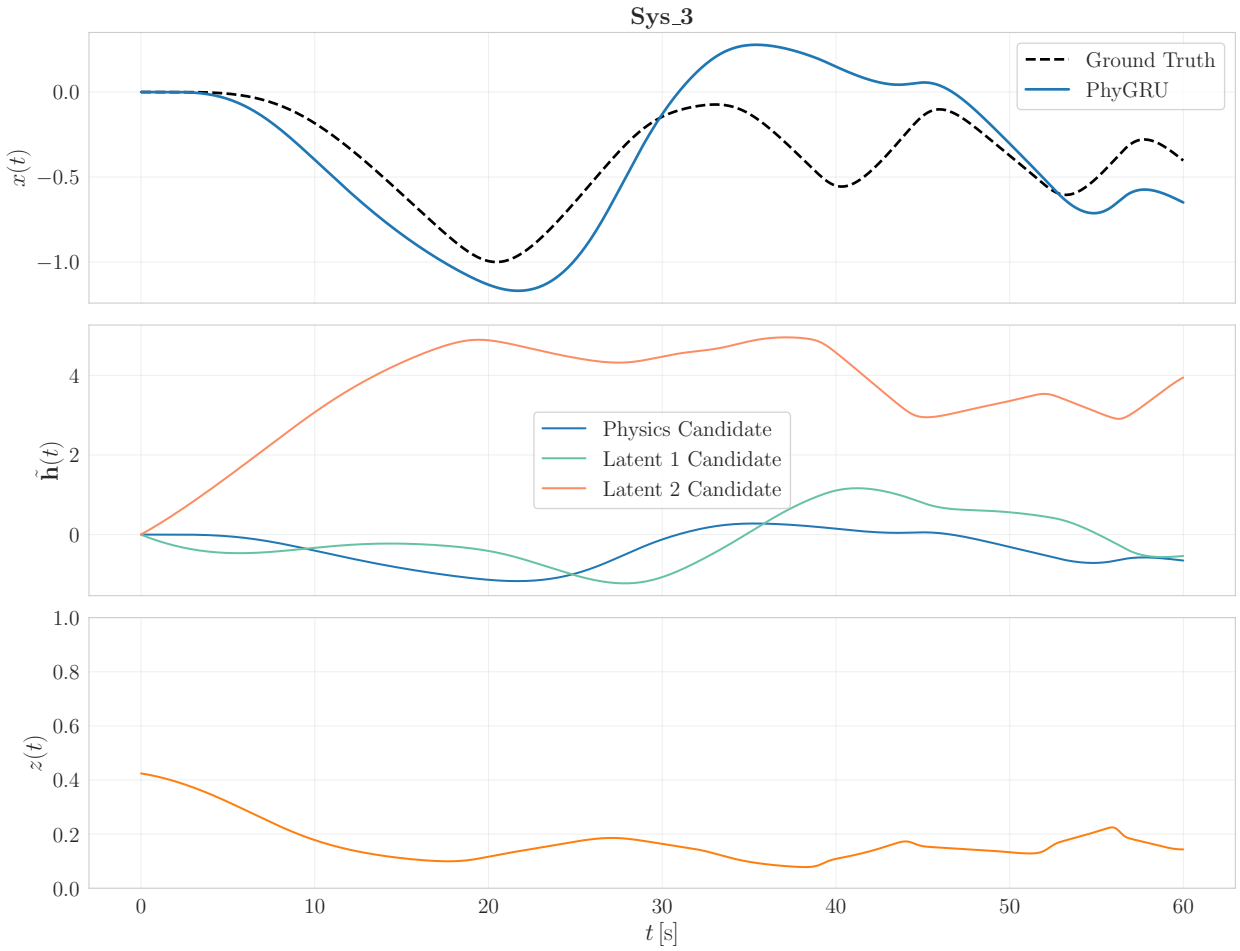

Figure 9: Physics and latent candidates values over time and gate activation, specific for ξ component, for the case with latent size equal to two, for the System 3 in time invariant parameters

To experimentally support the analysis of incremental stability, a test was conducted using System 1 with time-invariant parameters, simulating PhyGRU with latent size equal to one and the standard GRU with 32 hidden units, over 200 different initial values randomly sampled in the range [-2.5, 2.5] according to a uniform distribution, thereby broadly perturbing the initial condition. The evolution of the deviation from the unperturbed reference is then tracked over time. The results are shown in Figure 10, confirming expectations from the analysis, and illustrating how the standard GRU practically produces good results but without the structural insight that can instead be derived for PhyGRU.

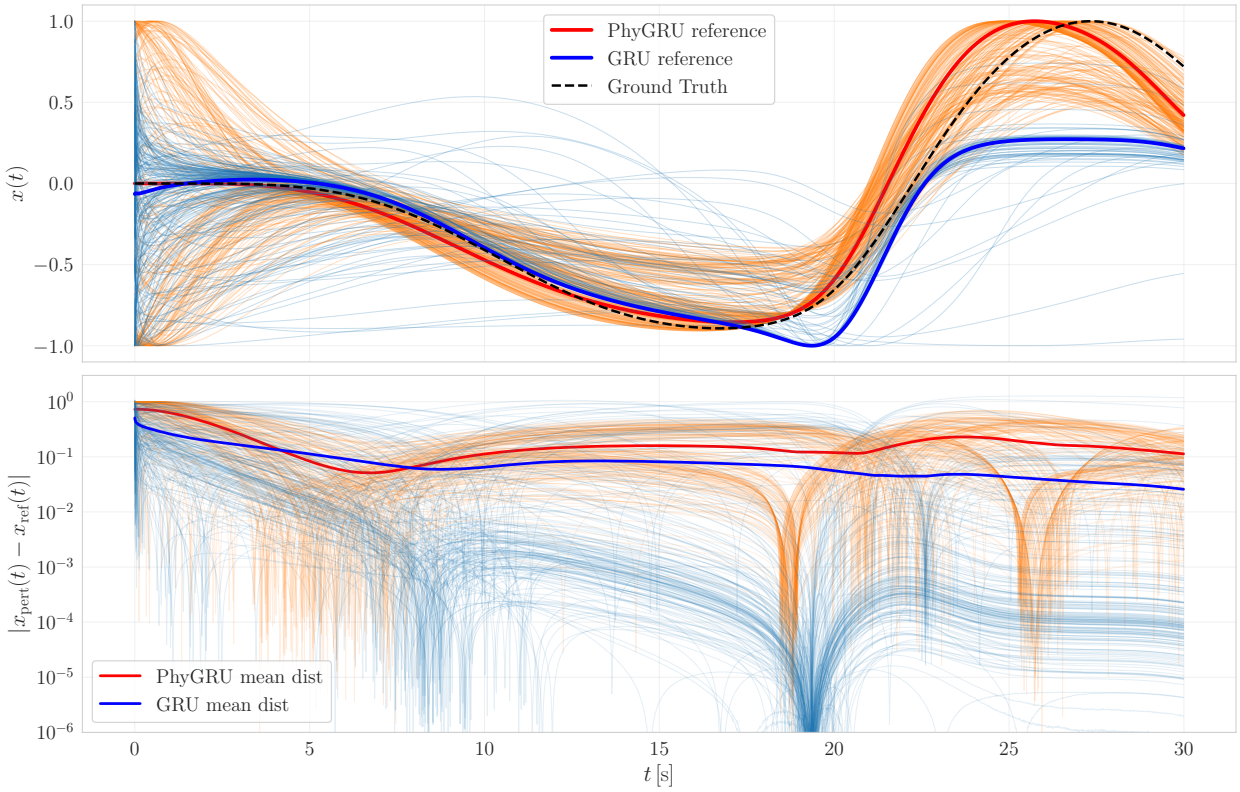

Figure 10: Incremental stability study using PhyGRU_l1 and GRU_h32 on System 1, in TI case, varying initial conditions

5.2 Results with Time Varying Parameters

In this section, similarly to the previous one, the results of the models on the three systems under time-varying conditions are reported, with the aim of identifying the failure modes of PhyGRU. To this end, the model is challenged not only with systems that differ from the physical prior, as done in the time-invariant cases, but also with systems whose dynamics evolve over time. This set of experiments is considered particularly relevant for the reader, as it is conceptually close to what can be expected in practical operational applications of the model. The numerical results are reported in Tables 10, 11 and 12 while the corresponding test and validation set trajectories are shown in Figures 11 and 12.

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	2.620e−1/5.772e−1	0.811/0.820
GRU	hidden=2	33	2.112e−1/5.417e−1	0.808/0.805
GRU	hidden=4	89	2.587e−1/5.197e−1	0.819/0.841
GRU	hidden=8	273	2.305e−1/5.129e−1	0.808/0.806
GRU	hidden=32	3393	1.828e−1/3.516e−1	0.869/0.935
GRU_{obs}	hidden=1	23	2.575e−1/5.517e−1	0.880/0.866
GRU_{obs}	hidden=2	51	2.786e−1/5.970e−1	0.807/0.815
GRU_{obs}	hidden=4	125	3.020e−1/5.423e−1	0.801/0.806
GRU_{obs}	hidden=8	345	2.106e−1/4.561e−1	0.878/0.890
GRU_{obs}	hidden=32	3681	2.161e−1/4.874e−1	0.800/0.821
PhyGRU	latent=0	11	3.789e−1/5.419e−1	0.846/0.884
PhyGRU	latent=1	23	3.379e−1/4.267e−1	0.827/0.892
PhyGRU	latent=2	39	1.829e−1/3.055e+0	0.826/0.337
PhyGRU	latent=3	59	7.668e−1/1.027e+2	0.615/0.514

Table 10: Results for the compared architectures, over System 1 in TV condition.

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	6.782e−1/1.189e+0	0.815/0.688
GRU	hidden=2	33	3.535e−1/9.584e−1	0.857/0.831
GRU	hidden=4	89	2.338e−1/8.113e−1	0.850/0.840
GRU	hidden=8	273	2.824e−1/8.163e−1	0.867/0.784
GRU	hidden=32	3393	1.451e−1/6.866e−1	0.882/0.852
GRU_{obs}	hidden=1	23	7.170e−1/1.371e+0	0.704/0.648
GRU_{obs}	hidden=2	51	3.802e−1/1.100e+0	0.819/0.698
GRU_{obs}	hidden=4	125	3.362e−1/9.858e−1	0.777/0.666
GRU_{obs}	hidden=8	345	3.480e−1/9.172e−1	0.763/0.656
GRU_{obs}	hidden=32	3681	2.484e−1/8.775e−1	0.818/0.678
PhyGRU	latent=0	11	3.173e−1/6.807e−1	0.846/0.798
PhyGRU	latent=1	23	4.449e−1/9.735e−1	0.851/0.812
PhyGRU	latent=2	39	6.731e−1/3.602e+0	0.632/0.272
PhyGRU	latent=3	59	4.503e−1/7.931e−1	0.652/0.787

Table 11: Results for the compared architectures, over System 2 in TV condition.

Model	latent/hidden	Params	MSE (val / test)	Spearman (val / test)
GRU	hidden=1	14	7.410e−1/1.344e+0	0.808/0.625
GRU	hidden=2	33	4.293e−1/1.053e+0	0.800/0.810
GRU	hidden=4	89	5.294e−1/1.109e+0	0.893/0.778
GRU	hidden=8	273	4.286e−1/1.053e+0	0.877/0.812
GRU	hidden=32	3393	3.220e−1/9.020e−1	0.820/0.800
GRU_{obs}	hidden=1	23	9.030e−1/1.563e+0	$-0.749/-0.582$
GRU_{obs}	hidden=2	51	4.049e−1/8.879e−1	0.787/0.761
GRU_{obs}	hidden=4	125	6.783e−1/1.258e+0	0.807/0.677
GRU_{obs}	hidden=8	345	7.618e−1/1.374e+0	0.754/0.598
GRU_{obs}	hidden=32	3681	4.553e−1/1.194e+0	0.781/0.617
PhyGRU	latent=0	11	5.247e−1/9.236e−1	0.864/0.850
PhyGRU	latent=1	23	7.327e−1/1.509e+0	$-0.458/-0.154$
PhyGRU	latent=2	39	6.415e−1/8.269e−1	0.761/0.801
PhyGRU	latent=3	59	4.115e−1/1.241e+0	0.718/0.713

Table 12: Results for the compared architectures, over System 3 in TV condition.

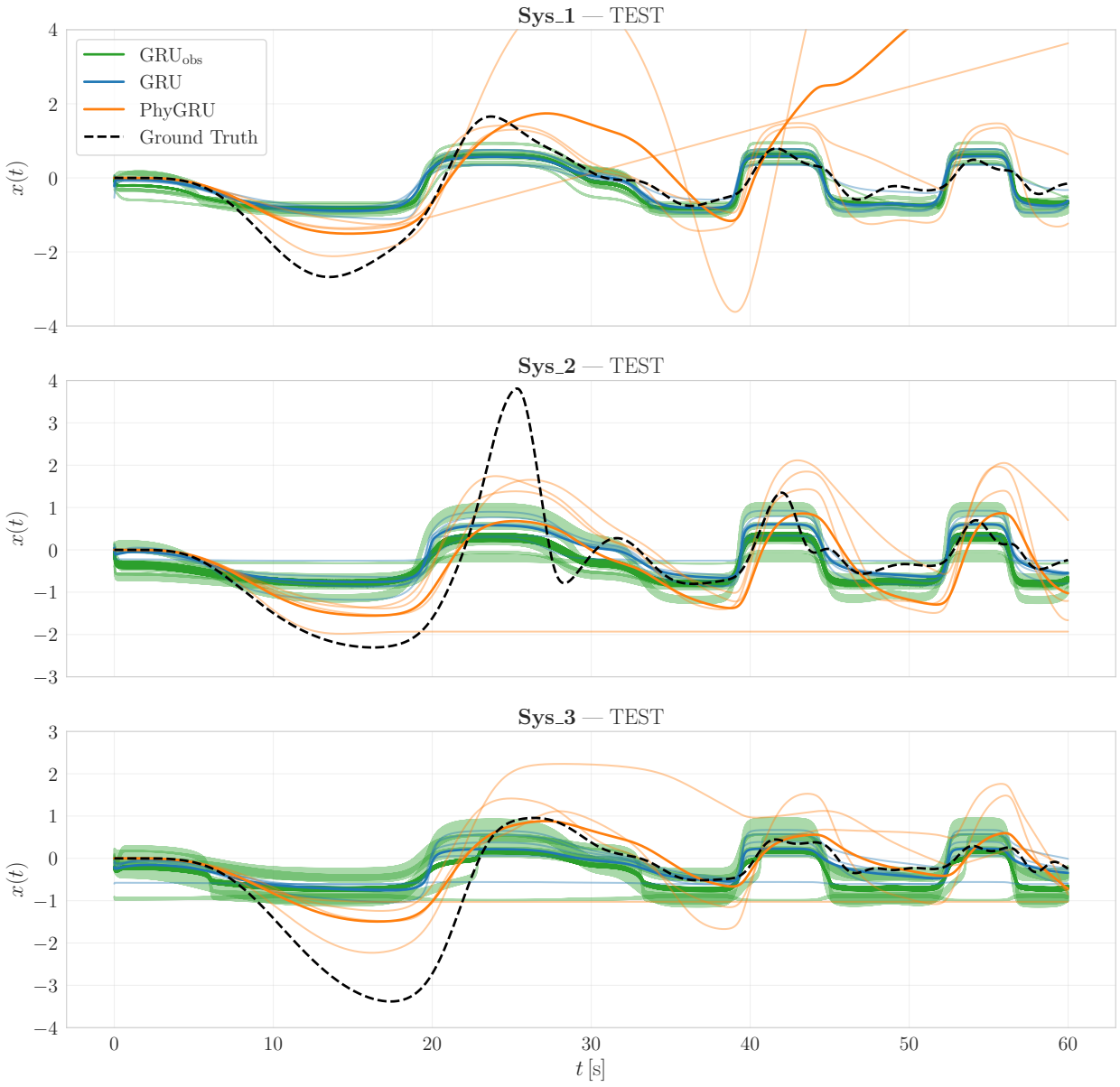

Figure 11: Test results trajectories over the considered systems, in time varying parameters

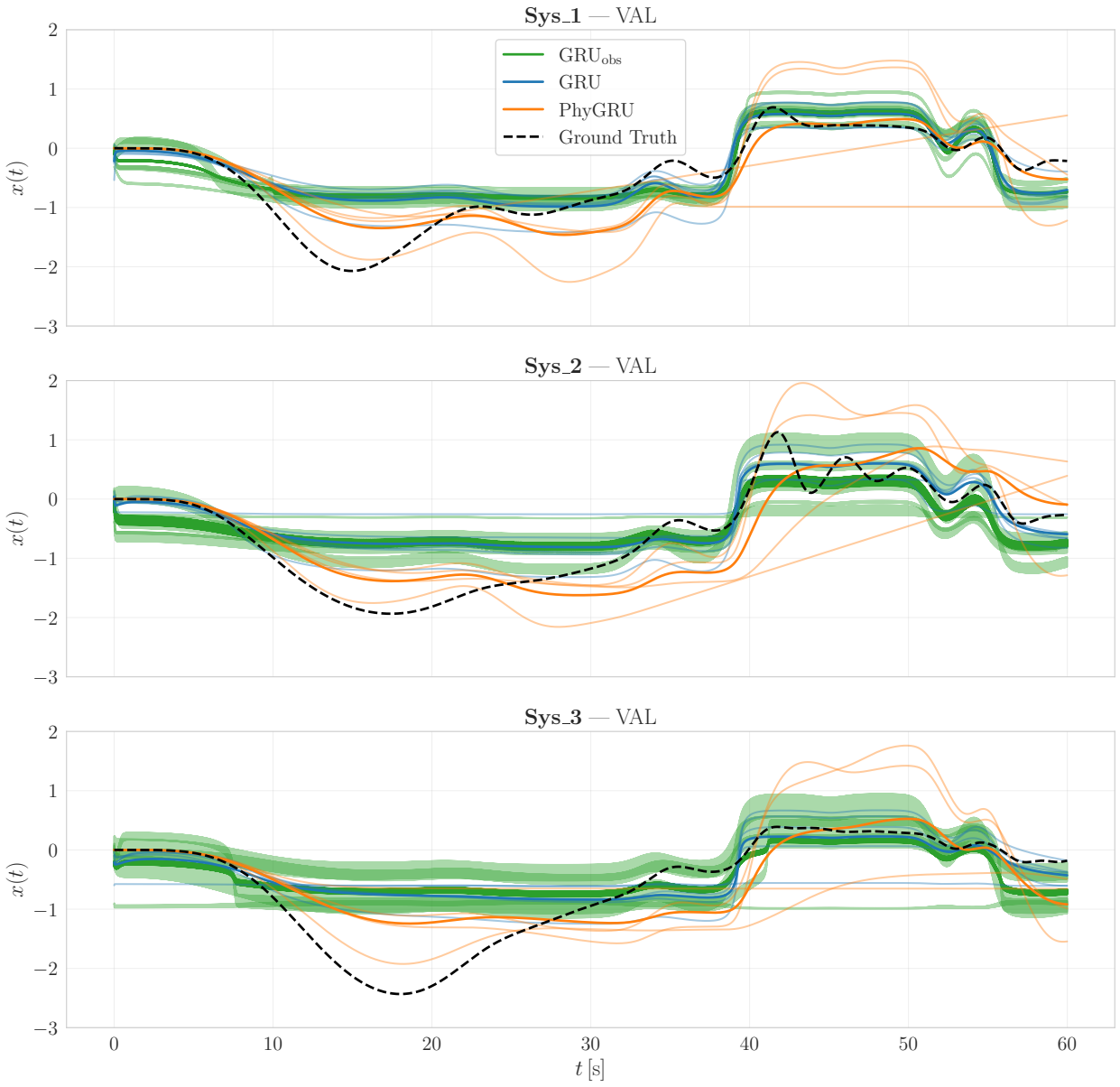

Figure 12: Validation results trajectories over the considered systems, in time varying parameters

5.3 Results with Stacked PhyGRU-GRU Time Varying Parameters

In this final results section, the investigation begins on whether using PhyGRU in a stacked configuration with standard GRU layers can provide tangible benefits to the overall system. These analyses were therefore conducted to simulate its application in contexts where GRU cells were already present, by replacing them in the simplest and most reasonable manner. Specifically, the GRU layer closest to the input is substituted with a PhyGRU layer, so as to provide an informed representation to the subsequent standard GRU layer. As in the previous sections, the numerical results are reported in Table 13, while the corresponding validation and test set trajectories are shown in Figures 13 and 14. For these final tests, only the test-set results are reported; however, the corresponding validation trajectories plots are still provided.

Model	L-H	System 1		System 2		System 3	
		Test MSE	Spearman	Test MSE	Spearman	Test MSE	Spearman
GRU-GRU	—8	5.139e−1	0.934	9.029e−1	0.850	8.954e−1	0.811
GRU-GRU	—16	4.309e−1	0.934	7.409e−1	0.866	8.590e−1	0.840
GRU-GRU	—32	4.378e−1	0.936	7.231e−1	0.854	9.157e−1	0.849
PhyGRU-GRU	0-8	4.658e−1	0.854	8.051e−1	0.707	8.710e−1	0.642
PhyGRU-GRU	0-16	6.272e−1	0.806	1.225e+0	0.654	8.413e−1	0.680
PhyGRU-GRU	0-32	6.041e−1	0.696	9.258e−1	0.594	8.287e−1	0.707
PhyGRU-GRU	1-8	7.659e−1	0.732	8.021e−1	0.745	9.516e−1	0.709
PhyGRU-GRU	1-16	1.027e+0	0.079	8.027e−1	0.755	8.559e−1	0.700
PhyGRU-GRU	1-32	5.081e−1	0.840	7.338e−1	0.769	8.593e−1	0.629
PhyGRU-GRU	2-8	9.532e−1	0.248	1.699e+0	0.233	1.160e+0	0.418
PhyGRU-GRU	2-16	4.754e−1	0.843	8.822e−1	0.597	9.770e−1	0.766
PhyGRU-GRU	2-32	1.109e+0	−0.061	1.825e+0	0.028	1.200e+0	0.422

Table 13: Test MSE and Spearman correlation for stacked architectures across Systems 1–3 in TV condition, Latent and Hidden dimension are reported as L-H.

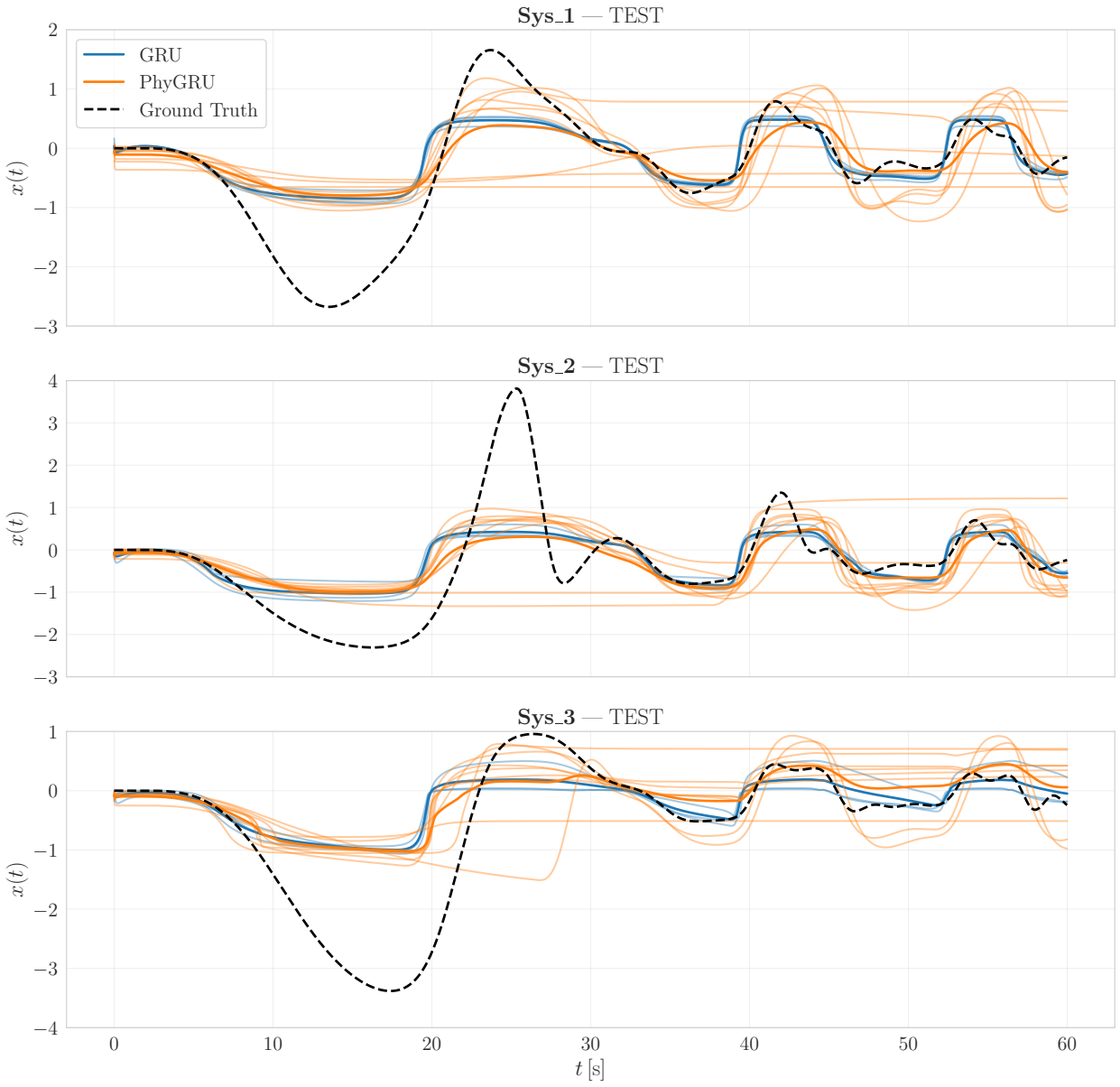

Figure 13: Test results trajectories over the considered systems, and stacked models, in time varying parameters. GRU and PhyGRU indicates the first layer of the stack

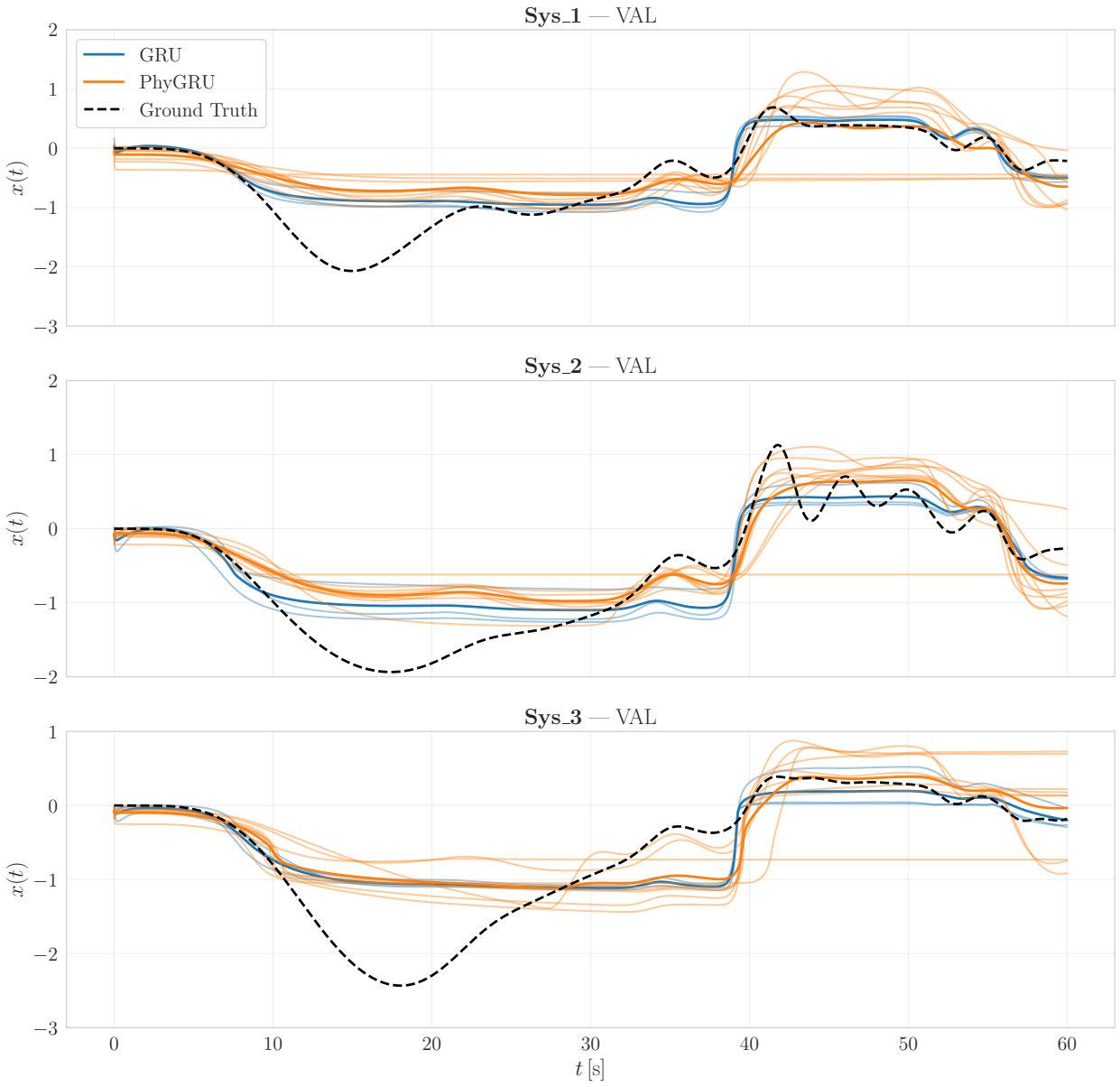

Figure 14: Validation results trajectories over the considered systems, and stacked models, in time varying parameters. GRU and PhyGRU indicates the first layer of the stack

5.4 Discussion & Limitations

The results show that PhyGRU is effective when the physics prior closely matches the real system. As the prior deviates, performance degrades but remains a valid internal proxy. Even when standard GRU metrics are comparable, trajectory analysis shows PhyGRU produces responses closer to physical dynamics, unlike the stepped, unnatural GRU output, supporting its potential application.

From the comparative analyses performed by introducing a reset gate on the latent component, it emerges that this modification is not consistently beneficial, furthermore, in the most relevant use case, where the physical model provided a priori is significantly different from the system to be modeled, as in the case of System 3, in Table 8, the reset gate proves to be detrimental, thereby supporting the design choice of minimalism and, consequently, its omission.

By inspecting the internal states of the system, one can observe, in Figure 8 and 9 the different activation patterns of the update gate, highlighting how the PhyGRU model strongly exploits the flexibility provided by this mechanism. Moreover, given the model structure, it is straightforward to analyze how the different candidates—those driven by the provided physical model and those arising from the latent dynamics—evolve during prediction, making PhyGRU more interpretable than a standard GRU for use cases involving dynamical systems.

Furthermore, from the analysis carried out on the identified parameters for System 1 under time-invariant conditions, available in Table 9 and Figure 7, aimed at understanding whether PhyGRU was actually able to identify the parameters of the original system, it is observed that, despite some deviations from the provided initial guesses, PhyGRU did not primarily focus on tuning the physical parameters. Instead, the dominant contribution came from the update gate. This is particularly interesting, as it shows that the system operated in a somewhat counterintuitive manner even in an ideal and relatively simple case, effectively exploiting the update gate while using the provided physical model mainly as a reference rather than as the primary source of adaptation.

From the study of incremental stability properties, in Figure 10, it is observed that the obtained results are consistent with what is expected from the theoretical considerations. In the tested case, PhyGRU effectively exhibits a global Lipschitz well-posedness, whereas the standard GRU counterpart cannot theoretically provide any structural remark, although it generally shows good practical behavior. However, the results highlight that not all GRU simulations remain close to the reference profile, some trajectories diverge and follow different paths. Although these cases are few, they provide experimental evidence of what was theoretically intended to be shown.

From the first analysis obtained by stacking a PhyGRU layer followed by a standard GRU, in order to provide an "informed" input to the subsequent GRU, it is observed that, despite better global metrics for the GRU-GRU stack, reported in Table 13, the PhyGRU-GRU version allowed the extraction of features in the response signal that are closer to the target system, as can be noted in Figure 14 from $30 - 40$ s, producing trajectories more coherent with the expected response. While there is still a noticeable dominance of the output by the standard GRU, this first analysis demonstrates that the introduction of recurrent cells like PhyGRU can be beneficial.

Nevertheless, despite the encouraging results obtained, it is necessary to clearly discuss the limitations of the proposed solution and to provide appropriate remarks on the evaluation methodology adopted. The proposed GRU variant shows promising performance, but it should not be regarded as a generally superior approach beyond the specific use case considered. In scenarios where extensive and detailed knowledge of the underlying system is available, or where a large amount of data can be exploited, this approach is not expected to outperform state-of-the-art solutions, although it may still be relevant within systems already based on GRU architectures. As highlighted by the experimental evidence and analytical considerations, while the model is theoretically expected to scale asymptotically in time similarly to standard GRUs, in practice it introduces a larger constant

factor. In specific real-time use cases where inference time is a critical constraint, this overhead may become limiting, especially considering that highly optimized GRU implementations are already available in modern Machine Learning frameworks, achieving significantly lower per-sample inference times. Furthermore, the adopted methodology aims to replicate a use case characterized by a limited amount of data and only a few trajectories available for training. By definition, this represents a highly challenging scenario for purely data-driven models. Additionally, since this solution effectively includes an integration step, while it provides a response more consistent with the physics being modeled, it also inherits all the limitations of numerical integration, making it more sensitive during hyperparameter tuning. From the sensitivity analyses performed by varying the timestep during training and inference, reported in Table 6 and Figure 4, it is observed that this parameter is important and must be chosen carefully, ideally aligned with the data sampling rate and taking into account the stability of the integrator. In particular, It is noted that a model trained with an inadequate timestep can, to some extent, be corrected a posteriori at inference time to yield a coherent prediction; however, this practice is not ideal and has not been sufficiently explored to be considered reliable. It is also noteworthy that predictions obtained with a very large timestep tend to resemble the responses produced by standard GRU models, which suggests that PhyGRU is operating primarily in a data-driven mode, assigning most of the weight to the latent component, and opening interesting avenues for future, more focused investigation.

Therefore, it is not claimed that PhyGRU is always superior to or an improvement over a standard GRU; rather, in the specific use case considered, the presented results suggest that PhyGRU can be a convenient choice when operating with scarce data by exploiting the known physics of the system. In conditions where a large amount of data is available, which were not tested in this work, it is expected that a standard GRU would be significantly more effective in representing the target system than what is observed in this particular application scenario.

6 Conclusion

In this work, PhyGRU, a variant of the Gated Recurrent Unit (GRU) architecture is presented, designed to accept an ODE as its internal state. By doing so, an inductive bias is enforced during training, based on a model ideally close to what will later emerge from the data. Several exploratory tests were conducted to highlight and clarify the contexts in which a variant of this type exhibits an effective advantage over a standard GRU, and initial experiments were also carried out to evaluate the performance of a mixed PhyGRU–GRU architecture. These results indicate that PhyGRU can provide a meaningful contribution by integrating prior knowledge of the system. Across the considered cases, PhyGRU has shown, according to the analyzed metrics and over the examined systems, ranging from systems structurally consistent with the provided physics prior to systems characterized by different orders and nonlinearities, to be an effectively improved solution. The trajectories produced by PhyGRU appear more natural and more consistent with the features of the target signal. In contrast, standard GRUs tend to yield solutions that, although robust, are rigid and step-like, failing to capture several characteristics of the target system. By inserting PhyGRU as the first layer in a PhyGRU–GRU stack, It is instead observed that the metrics favor the benchmark counterpart, consisting of two GRUs in series. Nevertheless, the resulting trajectories appear more consistent with the system to be modeled and reveal the emergence of features in the predictions that are closer to the true system behavior. Therefore, despite the unfavorable metrics, its use within systems based on traditional GRUs is still considered of interest.

These advantages are accompanied by several limitations, most notably increased inference times. Moreover, since the recurrent cell internally includes an integration step, it fully inherits the associated limitations related to numerical stability and sensitivity to stiff systems, the integration timestep has been shown to be a fundamental parameter that requires a careful and well-considered choice for successful training. Considerations regarding the incremental stability of the PhyGRU model have been presented from a theoretical perspective and supported by experimental results, showing that PhyGRU is able to exhibit global Lipschitz well-posedness behavior, which cannot be observed by standard GRUs.

Nevertheless, based on the obtained results, it is confident that a solution such as the one proposed can provide a useful tool for the community. In particular, clear practical applicability could be appreciated in a niche yet common use case: scenarios in which low-order knowledge of the system is available, together with a limited amount of data, possibly derived from only a few or even a single informative trajectory, where the goal is to build a model by effectively combining these sources of information exploiting the well known GRU structure or replace them, with minimal effort, in systems that already implement standard GRUs and falls in the operative case identified.

Acknowledgments

The author would like to acknowledge Matteo Parsani and Roberto Nuca of the Advanced Algorithms and Numerical Simulation Lab at King Abdullah University of Science and Technology (KAUST) for stimulating discussions and valuable insights.

A Appendix: PhyGRU Pytorch Implementation

This appendix provides a reference PyTorch (2.9.0+cpu) implementation of the PhyGRU architecture described in the paper. The implementation closely follows the equations in Section 3, with the following design choices:

- The `PhyGRUCell` implements the physics-informed candidate update using explicit Euler integration for the physical state and an optional latent state for unmodeled effects.
- The reset gate is omitted to preserve the physical consistency of the integrated state.
- The update gate is implemented as a learned sigmoid layer to blend the previous state with the physics-informed candidate.
- The `PhyGRU` class iterates the cell over a sequence of inputs, returning the first dimension of the state as the observable output.

```
class PhyGRUCell(nn.Module):
    def __init__(self, state_dim, input_dim, physics_law, latent_dim=0, dt=0.01):
        super().__init__()
        self.state_dim = state_dim
        self.latent_dim = latent_dim
        self.physics_law = physics_law
        self.dt = dt
        total_state = state_dim + latent_dim
        if latent_dim > 0:
            self.latent_dyn = nn.Linear(total_state + input_dim, latent_dim)
        else:
            self.latent_dyn = None
        self.z_gate = nn.Sequential(
            nn.Linear(total_state + input_dim, total_state),
            nn.Sigmoid()
        )

    def forward(self, state, u):
        phys_dot = self.physics_law(state[:, :self.state_dim], u)
        phys_next = state[:, :self.state_dim] + self.dt * phys_dot
        if self.latent_dim > 0:
            latent = state[:, self.state_dim:]
            latent_dot = self.latent_dyn(torch.cat([state, u], dim=1))
            latent_next = latent + self.dt * latent_dot
            candidate = torch.cat([phys_next, latent_next], dim=1)
        else:
```

```
            candidate = phys_next
        z = self.z_gate(torch.cat([state, u], dim=1))
        return z * candidate + (1 - z) * state

class PhyGRU(nn.Module):
    def __init__(self, physics_law, state_dim, input_dim, latent_dim=0, dt=0.01):
        super().__init__()
        self.cell = PhyGRUCell(state_dim, input_dim, physics_law, latent_dim, dt)
        self.state_dim = state_dim
        self.latent_dim = latent_dim

    def forward(self, u_seq):
        B, Tt, _ = u_seq.shape
        state = torch.zeros(B, self.state_dim + self.latent_dim,
                dtype=u_seq.dtype, device=u_seq.device)
        ys = []
        for t in range(Tt):
            state = self.cell(state, u_seq[:, t])
            ys.append(state[:, 0:1])
        return torch.stack(ys, dim=1)

class PhysicsPriorLaw(nn.Module):
    def __init__(self, learn_a=True, learn_b=True, learn_c=True):
        super().__init__()
        self.a = nn.Parameter(torch.tensor(0.5), requires_grad=learn_a)
        self.b = nn.Parameter(torch.tensor(0.6), requires_grad=learn_b)
        self.c = nn.Parameter(torch.tensor(0.7), requires_grad=learn_c)

    def forward(self, state, u):
        x, xd = state[:, 0], state[:, 1]
        u_s = u.squeeze()
        xdd = (u_s - self.b * xd - self.c * x) / (self.a + 1e-12)
        return torch.stack([xd, xdd], dim=1)

## model instantiation
phygru = PhyGRU(PhysicsPriorLaw(), 2, 1, latent_dim=ld, dt=t_step)
```

References

[1] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, 2014. DOI: https://doi.org/10.3115/v1/D14-1179

[2] X. Jia, S. Willard, A. Karpatne, J. Read, J. Zwart, M. Steinbach, V. Kumar, "Physics Guided RNNs for Modeling Dynamical Systems: A Case Study in Simulating Lake Temperature Profiles,", 2019 DOI: https://doi.org/10.48550/arXiv.1810.13075

[3] M. Lin, M. Bolderman, M. Lazar "Physics-guided gated recurrent units for inversion-based feedforward control,"2025. DOI: https://doi.org/10.48550/arXiv.2507.14052

[4] M. Zarzycki, M. Ławryńczuk, "Physics-Informed Hybrid GRU Neural Networks for MPC Prediction," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 8726-8731, 2023. DOI: https://doi.org/10.1016/j.ifacol.2023.10.055

[5] R. Kumar, A. Gupta, S. P. Muthukrishnan, L. Kumar, S. Roy "sEMG-Driven Physics-Informed Gated Recurrent Networks for Modeling Upper Limb Multi-Joint Movement Dynamics,", 2025. DOI: https://doi.org/10.48550/arXiv.2408.16599

[6] Y. Zheng, C. Hu, X. Wang, Z. Wu "Physics-informed recurrent neural network modeling for predictive control of nonlinear processes," *J. Process Control*, vol. 128, p. 103005, 2023. DOI: https://doi.org/10.1016/j.jprocont.2023.103005

[7] J. Fernández, J. Chiachío, J. Barros, M. Chiachío, C. S. Kulkarni "Physics-guided recurrent neural network trained with approximate Bayesian computation: A case study on structural response prognostics," *Reliability Engineering & System Safety*, vol. 243, p. 109822, 2024. DOI: https://doi.org/10.1016/j.ress.2023.109822

[8] M. A. Saleh, A. N. Alquennah, A. Ghrayeb, S. S. Refaat, H. Abu-Rub, S. P. Khatri "A Self-Adaptive Physics-Informed Gated Recurrent Unit Neural Networks Model for Estimating the Lifetime of Li-ion Batteries,", 2024. DOI: https://www.techrxiv.org/doi/full/10.36227/techrxiv.171779482.20091989/v1

[9] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, "Neural ordinary differential equations,", 2019. DOI: https://doi.org/10.48550/arXiv.1806.07366

[10] Y. Rubanova, R. T. Q. Chen, D. K. Duvenaud, "Latent ODEs for irregularly-sampled time series,", 2019. DOI: https://doi.org/10.48550/arXiv.1907.03907

[11] P. Kidger, J. Morrill, J. Foster, T. Lyons, "Neural controlled differential equations for irregular time series,", 2020. DOI: https://doi.org/10.48550/arXiv.2005.08926

[12] M. Habiba, B. A. Pearlmutter, "Neural ordinary differential equation based recurrent neural network model," in *Proc. 31st Irish Signals and Systems Conf. (ISSC)*, 2020. DOI: https://doi.org/10.1109/ISSC49989.2020.9180182

[13] W. Zhai, Y. Bao, and D. Tao, "State space model-based Runge–Kutta gated recurrent unit networks for structural response prediction," *Nonlinear Dyn.*, 2024. DOI: https://doi.org/10.1007/s11071-024-10229-2

[14] S. S. Eshkevari, M. Takáč, S. N. Pakzad, M. Jahani, "DynNet: physics-based neural architecture design for nonlinear structural response modeling and prediction," *Eng. Struct.*, 2021. DOI: https://doi.org/10.1016/j.engstruct.2020.111582

[15] Z. Guo, J. Xu, "Physics-guided hybrid network for predicting nonlinear dynamic response of structures under bi-directional ground motions," *Comput. Methods Appl. Mech. Eng.*, 2026. DOI: https://doi.org/10.1016/j.cma.2025.118422

[16] T. Wang, H. Li, M. Noori, R. Ghiasi, S. C. Kuok, W. A. Altabey "Seismic response prediction of structures based on Runge–Kutta recurrent neural network with prior knowledge," *Eng. Struct.*, 2023. DOI: https://doi.org/10.1016/j.engstruct.2022.115576

[17] J. Long, J. Zhu, N. Wang, K. Luo, Y. Zhao, Y. Zhao, "Neural Ordinary Differential Equation and supervised gated recurrent units embedded with historical variables for petrochemical process prediction," *Ind. Eng. Chem. Res.*, 2025. DOI: https://doi.org/10.1021/acs.iecr.5c02157

[18] I. K. Deo, "Physics-guided deep learning for dynamical systems: applications to fluid flow and ocean acoustics," Univ. British Columbia, 2025. DOI: https://dx.doi.org/10.14288/1.0449988

[19] T. Hagge, P. Stinis, E. Yeung, A. M. Tartakovsky, "Solving differential equations with unknown constitutive relations as recurrent neural networks,", 2017 DOI: https://doi.org/10.48550/arXiv.1710.02242

[20] Y. Yang, H. Wei, J. Mao, J. Yang, W. Xu, Y. Xu, "A hybrid physics-informed gated recurrent unit model for coupled motion prediction in deep-sea mining systems," *Ocean Eng.*, 2026. DOI: https://doi.org/10.1016/j.oceaneng.2025.123472

[21] M. Ravanelli, P. Brakel, M. Omologo, Y. Bengio, "Improving speech recognition by revising gated recurrent units," , 2017. DOI: https://doi.org/10.48550/arXiv.1710.00641

[22] G. B. Zhou, J. Wu, C. L. Zhang, Z. H. Zhou, "Minimal gated unit for recurrent neural networks," *Int. J. Autom. Comput.*, vol. 13, pp. 226–234, 2016. DOI: https://doi.org/10.1007/s11633-016-1006-2

[23] J. Heck and F. M. Salem, "Simplified Minimal Gated Unit Variations for Recurrent Neural Networks,", 2017. DOI: https://doi.org/10.48550/arXiv.1701.03452