Learning Data-Driven Stable Corrections of Dynamical Systems-Application to the Simulation of the Top-Oil Temperature Evolution of a Power Transformer

Chady Ghnatios, Xavier Kestelyn, Guillaume Denis, Victor Champaney, Francisco Chinesta

energies

MDPI

Article

Learning Data-Driven Stable Corrections of Dynamical Systems—Application to the Simulation of the Top-Oil Temperature Evolution of a Power Transformer

Chady Ghnatios [1,†], Xavier Kestelyn [2,†], Guillaume Denis [3,†], Victor Champaney [4,†] and Francisco Chinesta [5,6,*,†]

1 SKF Chair, PIMM Lab, Arts et Metiers Institute of Technology, 151 Boulevard de l'Hôpital, 75013 Paris, France; chady.ghnatios@ensam.eu
2 ULR 2697-L2EP, Centrale Lille, Junia ISEN Lille, Arts et Metiers Institute of Technology, University of Lille, 59000 Lille, France; xavier.kestelyn@ensam.eu
3 RTE R&D, 7C Place du Dôme, 92073 Paris La Defense, CEDEX, France; guillaume.denis@rte-france.com
4 ESI Chair, PIMM Lab, Arts et Métiers Institute of Technology, 151 Boulevard de l'Hôpital, 75013 Paris, France; victor.champaney@ensam.eu
5 RTE Chair, PIMM Lab, Arts et Métiers Institute of Technology, 151 Boulevard de l'Hôpital, 75013 Paris, France
6 CNRS@CREATE, 1 Create Way, 04-05 Create Tower, Singapore 138602, Singapore
* Correspondence: Francisco.Chinesta@ensam.eu
† All authors contributed equally to this work.

Abstract: Many engineering systems can be described by using differential models whose solutions, generally obtained after discretization, can exhibit a noticeable deviation with respect to the response of the physical systems that those models are expected to represent. In those circumstances, one possibility consists of enriching the model in order to reproduce the physical system behavior. The present paper considers a dynamical system and proposes enriching the model solution by learning the dynamical model of the gap between the system response and the model-based prediction while ensuring that the time integration of the learned model remains stable. The proposed methodology was applied in the simulation of the top-oil temperature evolution of a power transformer, for which experimental data provided by the RTE, the French electricity transmission system operator, were used to construct the model enrichment with the hybrid rationale, ensuring more accurate predictions.

Keywords: stable integrator; hybrid twin; machine learning; dynamical system; power transformer; monitoring

Citation: Ghnatios, C.; Kestelyn, X.; Denis, G.; Champaney, V.; Chinesta, F. Learning Data-Driven Stable Corrections of Dynamical Systems—Application to the Simulation of the Top-Oil Temperature Evolution of a Power Transformer. *Energies* **2023**, *16*, 5790. https://doi.org/10.3390/en16155790

Academic Editors: Teke Gush and Raza Haider

Received: 25 June 2023
Revised: 26 July 2023
Accepted: 2 August 2023
Published: 4 August 2023

1. Introduction

The two most common approaches involved in simulation-based engineering (SBE), the one mainly based on physics and the more recent one based on the use of data manipulated by advanced machine learning techniques, both have inherent limitations.

In general, the physics-based modeling framework produces responses that approximate the real ones quite well as long as an accurate enough model exists. The main difficulties when considering such a physics-based approach are: (i) the fidelity of the model itself, which is assumed to be calibrated; (ii) the impact that variability and uncertainty induce; and (iii) the computing time required for solving the complex and intricate mathematical models.

On the other hand, the data-driven framework is not fully satisfactory because, even if the data (assumed to be noise-free) represent the reality well, the extrapolation or interpolation in space and time from the collected data (in particular, locations and times instants) usually entail a noticeable accuracy loss.

Of course, by increasing the amount of collected data, one could expect to be able to approximate the real solution with more fidelity; however, data are not always simple to

collect and not always possible to access, and in all cases, collecting data is expensive (cost of sensors, cost of communication and analysis, etc.). Equipping a very large industrial or civil infrastructure with millions of sensors to cover all its spatial dimension seems simply unreasonable.

Moreover, even when the solution is properly approximated, two difficulties persist: (i) the solution explainability, compulsory to certify solutions and decisions; and (ii) the domain of validity when extrapolating far from the domain where data were collected.

In view of the limitations of both existing procedures, one gateway consists of allying both to conciliate agility and fidelity. The hybrid paradigm seems a valuable and appealing option. It considers the reality expressible from the addition of two contributions: the existing knowledge (the state-of-the-art physics-based model or any other kind of knowledge-based model) and the part of the reality that the model ignores, the so-called ignorance (also called the deviation, gap, or discrepancy).

1.1. The Three Main Simulation-Based Engineering Methodologies Revisited

To introduce and discuss different simulation-based engineering (SBE) frameworks, we consider a certain field $u(\mathbf{x})$ describing the evolution of the variable u along the space defined by the coordinates $\mathbf{x} \in \Omega \subset \mathbb{R}^{\mathtt{D}}$.

We assume that a certain knowledge of the addressed physics exists, described by the model $\mathcal{M}$, which, in general, consists of a system of algebraic equations or a system of differential equations complemented with the appropriate boundary and initial conditions ensuring the problem's solvability. The solution provided by the model just referred to, which, as indicated, represents the existing knowledge of the problem at hand, is represented by $u^M(\mathbf{x})$.

Due to the partial knowledge of the addressed physical phenomenon, the calculated solution $u^M(\mathbf{x})$ is expected to differ from the reference one $u(\mathbf{x})$, which is intended to be represented with maximum fidelity.

Thus, we define the residual $R^M(\mathbf{x})$ according to

$$R^M(\mathbf{x}) = u(\mathbf{x}) - u^M(\mathbf{x}), \tag{1}$$

where the error can be computed from its norm, $\mathcal{E}^M = \|R^M(\mathbf{x})\|$, for which the L2 norm is usually employed.

For reducing that error, different possibilities exist:

- **Physics-based model improvement**. This approach consists of refining the modeling by enriching the model itself, $\mathcal{M} \rightarrow \widehat{\mathcal{M}}$, such that its solution $u^{\widehat{M}}$ exhibits a smaller error; i.e., $\mathcal{E}^{\widehat{M}} \leq \mathcal{E}^M$;
- **Fully data-driven description**. The data-driven route consists of widely sampling the space Ω, $\mathbf{x}_1, \ldots, \mathbf{x}_{\mathtt{P}}$, with large enough $\mathtt{P}$ and with the location of the points $\mathbf{x}_i$, $i = 1, \ldots, \mathtt{P}$, maximizing the Ω domain coverage. These points are grouped into the set $\mathcal{S}$.
  The coverage is defined by the convex hull $\omega(\mathcal{S})$ of the set $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{\mathtt{P}}\}$, ensuring interpolation for $\mathbf{x} \in \omega(\mathcal{S})$ and limiting the risky extrapolation to the region of Ω outside the convex hull $\omega(\mathcal{S})$.
  Factorial samplings try to maximize the coverage; however, factorial samplings, or those based on the use of Gauss–Lobatto quadratures, related to approximations making use of orthogonal polynomials [1], fail when the dimensionality of the space $\mathtt{D}$ increases.
  When $\mathtt{D} \gg 1$, sparse sampling is preferred, the Latin hyper cube (LHP), for instance. Samplings based on Gaussian processes (GPs) aim at distributing the points in locations where the uncertainty is maximum (with respect to the predictions inferred from the previously collected data).

Finally, the so-called *active learning* techniques drive the sampling with the aim of maximizing the representation of a certain goal-oriented quantity of interest [2].
In what follows, we assume generic sampling to access the reference solution $u(\mathbf{x}_i)$, $i = 1, \ldots, \mathtt{P}$ and perfect measurability.
Now, to infer the solution at $\mathbf{x} \in \omega(\mathcal{S})$, it suffices to construct an interpolation or approximation; more generally, an adequate regression $u^{\mathcal{S}}(\mathbf{x})$:

$$u^{\mathcal{S}}(\mathbf{x}) \equiv u(\mathbf{x}; u_1, \ldots, u_{\mathtt{P}}), \tag{2}$$

where $u_1 \equiv u(\mathbf{x}_1), \ldots, u_{\mathtt{P}} \equiv u(\mathbf{x}_{\mathtt{P}})$.
Different possibilities exist, including regularized polynomial regressions [3], neural networks (NNs) [4,5], support vector regression (SVR) [6], decision trees, and their random forest counterparts [7,8], to name a few.
The trickiest issue concerns the error evaluation, which is quantified from a part of the data kept outside the training set, the so-called test set, used to quantify the performance of the trained regression.
The main challenges of such a general procedure, particularly exacerbated in the multi-dimensional case ($\mathtt{D} \gg 1$), are the following:

- Ability to explain the regression $u^{\mathcal{S}}(\mathbf{x})$;
- The size of the dataset ($\mathtt{P}$), which scales with the problem dimensionality $\mathtt{D}$;
- The optimal sampling to cover Ω while guaranteeing the accuracy of $u^{\mathcal{S}}(\mathbf{x})$ or that of the goal-oriented quantities of interest;

- **Hybrid approach**. The hybrid approach proceeds by embracing the physics-based and data-driven approaches. As described in the next section, it can improve the physics-based accuracy (while profiting of the physics-based explanatory capabilities) through the use of data-driven enrichment, which, for its part, and under certain conditions, needs less data than the fully data-driven approach just discussed [9].

1.2. Paper Organization

The present paper aims at addressing the methodologies involved in the construction of data-driven model corrections and then proving their potential with a problem of practical relevance.

For that purpose, the paper is organized as follows. After this brief introduction that emphasizes the different ways of enriching models to improve their ability to represent the observed reality, Section 2 describes and discusses the hybrid approach where physics-based and data-driven models are combined to improve each of them.

Then, Section 3 focuses on the methodologies involved in the procedure of learning stable dynamical systems, where the role of memory is widely discussed.

Section 4 proposes a simple procedure that, combined with ResNet architectures, enables the construction of stable integrators. The procedure is validated with some simple dynamical systems.

Finally, Section 5 addresses a problem of practical relevance, that of predicting the oil temperature in an electric power transformer. More importantly, it proves the superiority of the hybrid approach with respect to the physics-based or the data-driven modeling approaches. The paper ends with some final concluding remarks.

2. Illustrating the Hybrid Approach

2.1. A Simple Linear Regression Reasoning

In the hybrid approach, we consider first the contribution of a model expected to reasonably represent the physics at hand and denoted by $u(\mathbf{x})$, which was previously denoted as $u^M(\mathbf{x})$, with $\mathbf{x} \in \Omega \subset \mathbb{R}^{\mathtt{D}}$.

As discussed before, it entails a residual $R^M(\mathbf{x})$ and the associated error, $\mathcal{E}^M$.

Imagine for a while the simplest (and cheapest) regression of that residual, a linear regression involving $\mathtt{P}$ data associated with the locations $\mathbf{x}_i \in \Omega$, $i = 1, \ldots, \mathtt{P}$.

Thus, the linear regression involving a linear approximation of the residual, denoted by $\Delta u^{\mathtt{L}}(\mathbf{x})$, reads:

$$\Delta u^{\mathtt{L}}(\mathbf{x}; a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}) = a_0^{\Delta} + a_1^{\Delta} x_1 + \ldots + a_{\mathtt{D}}^{\Delta} x_{\mathtt{D}}, \tag{3}$$

and involves the unknown coefficients $a_0^{\Delta}, a_1^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}$.

By using the L2 norm, the linear regression results from the least-square minimization problem:

$$\{a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}\} = \mathtt{argmin}_{a_0^*, \ldots, a_{\mathtt{D}}^*} \left\{ \sum_{i=1}^{\mathtt{P}} \left(\Delta u^{\mathtt{L}}(\mathbf{x}_i; a_0^*, \ldots, a_{\mathtt{D}}^*) - R^M(\mathbf{x}_i) \right)^2 \right\}, \tag{4}$$

which defines an interpolation in the case where $\mathtt{P} = \mathtt{D} + 1$ and an approximation when $\mathtt{P} \geq \mathtt{D} + 1$.

If we assume for a while that the reference solution $u(\mathbf{x})$ is fully available, and the residual previously defined as $R^M(\mathbf{x})$ is also available, the previous expression can be rewritten in a continuous form

$$\{a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}\} = \mathtt{argmin}_{a_0^*, \ldots, a_{\mathtt{D}}^*} \| \Delta u^{\mathtt{L}}(\mathbf{x}; a_0^*, \ldots, a_{\mathtt{D}}^*) - R^M(\mathbf{x}) \|_2. \tag{5}$$

The linear regression of the reference solution $u(\mathbf{x})$, denoted by $u^{\mathtt{L}}(\mathbf{x}; a_0^u, \ldots, a_{\mathtt{D}}^u)$, reads

$$u^{\mathtt{L}}(\mathbf{x}; a_0^u, \ldots, a_{\mathtt{D}}^u) = a_0^u + a_1^u x_1 + \ldots + a_{\mathtt{D}}^u x_{\mathtt{D}}, \tag{6}$$

where the coefficients $a_0^u, \ldots, u_{\mathtt{D}}^u$ result again from the least-square minimization problem

$$\{a_0^u, \ldots, a_{\mathtt{D}}^u\} = \mathtt{argmin}_{a_0^*, \ldots, a_{\mathtt{D}}^*} \left\{ \sum_{i=1}^{\mathtt{P}} \left(u^{\mathtt{L}}(\mathbf{x}_i; a_0^*, \ldots, a_{\mathtt{D}}^*) - u(\mathbf{x}_i) \right)^2 \right\}, \tag{7}$$

whose continuous expression reads

$$\{a_0^u, \ldots, a_{\mathtt{D}}^u\} = \mathtt{argmin}_{a_0^*, \ldots, a_{\mathtt{D}}^*} \| u^{\mathtt{L}}(\mathbf{x}; a_0^*, \ldots, a_{\mathtt{D}}^*) - u(\mathbf{x}) \|_2. \tag{8}$$

Equation (5) implies

$$\| \Delta u^{\mathtt{L}}(\mathbf{x}; a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}) - R^M(\mathbf{x}) \|_2 \leq \| R^M(\mathbf{x}) \|_2. \tag{9}$$

Thus, as soon as the error related to the residual becomes smaller than the one related to the linear approximation of the reference solution; i.e., if

$$\mathcal{E}^M = \| R^M(\mathbf{x}) \|_2 \leq \| u^{\mathtt{L}}(\mathbf{x}, a_0^u, \ldots, a_{\mathtt{D}}^u) - u(\mathbf{x}) \|_2, \tag{10}$$

then

$$\| \Delta u^{\mathtt{L}}(\mathbf{x}; a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}) - R^M(\mathbf{x}) \|_2 \leq \| R^M(\mathbf{x}) \|_2 \leq \| u^{\mathtt{L}}(\mathbf{x}, a_0^u, \ldots, a_{\mathtt{D}}^u) - u(\mathbf{x}) \|_2, \tag{11}$$

which proves the higher accuracy of the hybrid approximation as soon as the solution provided by the model $\mathcal{M}$ represents a better approximation to the reference solution $u(\mathbf{x})$ than a linear regression could attain.

2.2. General Remarks

The analysis just discussed considers a simple linear regression involving a linear approximation; however, it remains valid when considering regressions involving richer nonlinear approximations.

The first part of Equation (11),

$$\| \Delta u^{\mathtt{L}}(\mathbf{x}; a_0^{\Delta}, \ldots, a_{\mathtt{D}}^{\Delta}) - R^M(\mathbf{x}) \|_2 \leq \| R^M(\mathbf{x}) \|_2,$$

affirms that the simulated model solution is enriched in an L2-norm sense and under the constraint of having enough data to evaluate the considered norms. It is important to note that, even if the simulated model solution is enriched in the L2-norm sense, locally, the accuracy of the enriched solution could be degraded.

The second part pf Equation (11),

$$\|R^M(\mathbf{x})\|_2 \leq \|u^{\mathrm{L}}(\mathbf{x}, a_0^u, \ldots, a_{\mathrm{D}}^u) - u(\mathbf{x})\|_2,$$

works under some conditions involving the data and the model (i.e., with $u(x)$ and $u^M(x)$) to be checked and again needs enough data to enable the calculation of the L2 norms.

2.3. On the Domain of Application of the Hybrid Modeling Approach

The hybrid approach can be applied in different settings. It can be viewed as a sort of transfer learning, where rich behaviors are approached from others that are close enough and well established.

This hybrid approach seems particularly appealing in different situations, such as the ones reported below:

- When the model captures most of the solution features, the correction must describe a discrepancy that exhibits smaller nonlinearities, as was the case in [10,11], where the same amount of data performed better within the hybrid framework than within the fully data-driven framework;
- Sometimes, the physics-based model operates very accurately in a part of the domain, whereas strong nonlinearities localize in a small region, which can, in that case, be captured by a data-driven learning model, as considered in [12] to address the inelastic behavior of spot-welds;
- When considering the constitutive modeling of materials, the augmented rationale (or hybrid paradigm) expresses the real behavior from first-order behavior (calibrated from the available data) complemented by enrichment (or correction) filling the gap between the collected data and the predictions obtained from the assumed model [13];
- When addressing plates (or shells) with noticeable 3D behaviors (deviating from the usual shell theory) a valuable solution consists of using an enriched kinematics consisting of two contributions: the first-order one (usual shell kinematics) enriched with a second-order contribution [14] that can be learned from data;
- The hybrid modeling can also transfer existing knowledge slightly outside its domain of applicability with small amounts of collected data, as performed in [15] to correct state-of-the-art structural beam models;
- Sometimes, the discrepancy concerns an imperfect alignment in the solution between the prediction and the measures. That discrepancy may seem very high when evaluating it at each location; however, a small transport allows aligning both solutions. Optimal transport is very suitable in these situations where the hybrid model consists of the usual nominal model enriched from a parametric correction formulated in an optimal transport setting, as described in [16,17];
- In [18], a correction of a Mises yield function was performed from the deviation between the results predicted by using it and the measures obtained at the structure level;
- Finally, when addressing processing and performances, the hybrid formulation, which can be used with different granularities, can exhibit advantages (amount of data, ability to explain, knowledge transfer, etc.) that pertain to the heart of digital twin developments [9,19–24].

3. Methods

In what follows, a system characterized by a state x that evolves in time, (i.e., $x(t)$) is considered and the stability of its numerical integration is discussed [25]. When the state is multi-valued, it is denoted by $\mathbf{x}$.

3.1. On the Integration Stability

We consider a simple linear dynamical system expressed by

$$\frac{dx}{dt} = ax,\ a \in \mathbb{R}. \tag{12}$$

The integration reads

$$\frac{dx}{x} = a\ dt \rightarrow \mathtt{ln}(x) = C + a\ t \rightarrow x(t) = \hat{C}\ e^{at}, \tag{13}$$

with $\hat{C} = e^C$ and $\hat{C} = x(t = 0)$.

It can be noted that the existence of a bounded solution requires that $a \leq 0$ because if $a > 0$, $x(t \rightarrow \infty) = \infty$. Thus, the stability condition is $a \leq 0$.

Next, we consider the discrete case, in which the first-order derivative is discretized by a first-order finite difference

$$\frac{dx}{dt} = \frac{x_n - x_{n-1}}{\Delta t}, \tag{14}$$

where $x_n = x(t = t_n)$, with $t_n = n\Delta t$.

In that case, the discrete time evolution reads

$$x_n = ax_{n-1}\Delta t + x_{n-1} = (a\Delta t + 1)x_{n-1} = bx_{n-1}, \tag{15}$$

with $b = (a\Delta t + 1)$.

Now, because $x_{n-1} = bx_{n-2}$ and so on, we finally obtain

$$x_n = b^n x_0, \tag{16}$$

from which it can be concluded that bounded solutions are subjected to the constraint $b \leq 1$.

Similar results can be obtained in the case of multi-valued states. In what follows, we consider the differential system

$$\frac{d\mathbf{x}}{dt} = \mathbf{A}\mathbf{x}. \tag{17}$$

For the sake of simplicity, we assume that matrix $\mathbf{P}$ diagonalizes $\mathbf{A}$, with $\mathbf{P}^T\mathbf{P} = \mathbf{I}$ ($\mathbf{I}$ is the unit matrix). Thus, if we define $\mathbf{x} = \mathbf{P}\mathbf{y}$, we get

$$\mathbf{P}\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{P}\mathbf{y}, \tag{18}$$

and multiplying by $\mathbf{P}^T$ results in

$$\mathbf{P}^T\mathbf{P}\frac{d\mathbf{y}}{dt} = \mathbf{P}^T\mathbf{A}\mathbf{P}\mathbf{y}, \tag{19}$$

which, taking into account that $\mathbf{P}^T\mathbf{P} = \mathbf{I}$, can be rewritten as

$$\frac{d\mathbf{y}}{dt} = \mathbf{D}\mathbf{y}, \tag{20}$$

with $\mathbf{D}$ being diagonal.

The fact that $\mathbf{D}$ is diagonal allows the decoupling of the solution of Equation (20). Thus, we have

$$\frac{dy_i}{dt} = D_i y_i,\ \forall i, \tag{21}$$

with y_i being the i-component of $\mathbf{y}$ and D_i the (i, i)-diagonal component of $\mathbf{D}$.

Thus, using the previous results, the stability condition reads

$$\texttt{max}_i D_i \leq 0. \tag{22}$$

By discretizing Equation (20), we get

$$\mathbf{y}_n = (\Delta t \mathbf{D} + \mathbf{I})\mathbf{y}_{n-1}, \tag{23}$$

or, noting $\hat{\mathbf{D}} = \Delta t \mathbf{D} + \mathbf{I}$,

$$\mathbf{y}_n = \hat{\mathbf{D}}\mathbf{y}_{n-1}, \tag{24}$$

which, using the recurrence, results in

$$\mathbf{y}_n = \hat{\mathbf{D}}^n \mathbf{y}_0. \tag{25}$$

Thus, stability requires that the spectral radius ρ of $\hat{\mathbf{D}}$ is lower than the unity; i.e., $\rho(\hat{\mathbf{D}}) \leq 1$. These conditions should be satisfied by the dynamical learned models, as discussed later.

3.2. Learning Integrators

When the dynamical system is known, different numerical integration schemes (explicit, implicit, or semi-implicit) can be applied with different convergence orders; for instance, the Euler or the more accurate Runge–Kutta orders, among many other choices. Thus, the discretization time step Δt must be chosen to ensure stability and to guarantee convergence; that is, a numerical solution close enough to the reference solution of the problem.

In what follows, we revisit and discuss different machine learning techniques that enable the learning of dynamical systems and apply them for the integration of dynamical systems, as performed by recurrent NN (rNN) and long short-term memory (LSTM) techniques [26,27].

In this section, $\mathbf{h}$ refers to the hidden state, whereas $\mathbf{l}$ and $\mathbf{o}$ refer, respectively, to the inputs (loading) and observable outputs.

3.2.1. Recurrent Neural Network

If $\mathbf{W}^\bullet$ represents dense matrices associated with the variables $\bullet$, $\sigma(\cdot)$ the activation function, $\mathbf{h}_n$ the internal state at the n-time step t_n, and $\mathbf{l}_n$ and $\mathbf{o}_n$ the associated input and output at the present time t_n, then the rNN proceeds from:

$$\begin{cases} \mathbf{h}_n = \sigma(\mathbf{W}^h \mathbf{h}_{n-1} + \mathbf{W}^l \mathbf{l}_n) \\ \mathbf{o}_n = \sigma(\mathbf{W}^o \mathbf{h}_n), \end{cases}, \tag{26}$$

the architecture of which is depicted in Figure 1.

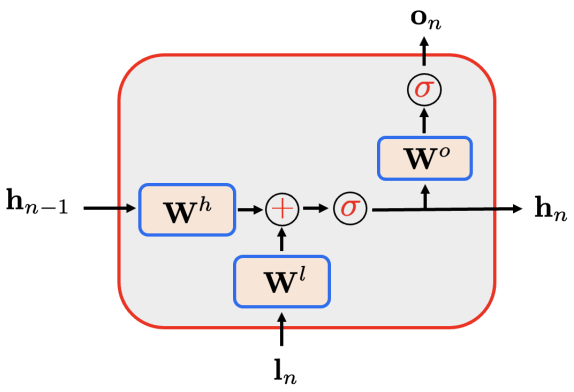

Figure 1. Recurrent neural network architecture.

To implement the time evolution, it suffices to use the rNN and re-inject the rNN output $\mathbf{h}_n$ as the input at the next time step.

A temporal sequence of inputs and outputs, which is assumed to be available, allows calculating the different weights composing the different $\mathbf{W}$ matrices.

To highlight the connection between the rNN and the usual dynamical system integrators, we consider the finite element semi-discretized form of a linear parabolic differential equation

$$\mathbf{M}\frac{d\mathbf{h}}{dt} = \mathbf{K}\mathbf{h} + \mathbf{l}, \tag{27}$$

where $\mathbf{M}$ and $\mathbf{K}$ are two matrices that result from the spatial discretization. Here, $\mathbf{h}$ refers to the state and $\mathbf{l}$ to the system loading.

Then, the explicit discretization of Equation (27) reads

$$\mathbf{M}\mathbf{h}_n = \Delta t\mathbf{K}\mathbf{h}_{n-1} + \Delta t\mathbf{l}_n + \mathbf{M}\mathbf{h}_{n-1} = (\Delta t\mathbf{K} + \mathbf{M})\mathbf{h}_{n-1} + \Delta t\mathbf{l}_n, \tag{28}$$

from which the state updating results in

$$\mathbf{h}_n = \mathbf{M}^{-1}(\Delta t\mathbf{K} + \mathbf{M})\mathbf{h}_{n-1} + \Delta t\mathbf{M}^{-1}\mathbf{l}_n, \tag{29}$$

which can be rewritten as

$$\mathbf{h}_n = \mathbf{W}^h\mathbf{h}_{n-1} + \mathbf{W}^l\mathbf{l}_n, \tag{30}$$

This corresponds to the particularization of Equation (26) to the linear case. Thus, the intimate connection between the rNN and the usual discretization techniques becomes explicit. The latter operates with a known model whereas the former learns the model itself (the different $\mathbf{W}^\bullet$ matrices).

Remark 1. *In the linear model just discussed, when the whole state is observed (i.e.,* $\mathbf{o}_n = \mathbf{h}_n$*),* $\mathbf{W}^o = \mathbf{I}$.

3.2.2. On the Model Memory

When considering a first-order dynamical system as just discussed, the solution at a particular time can be computed from only the knowledge of the previous solution (at the previous time step) and the present loading (also known as action or input). Thus, one could imagine that, for learning first-order dynamical systems, a short memory, such as, for instance, the rNN just described, would suffice.

However, sometimes, larger memory is needed even when addressing first-order models, as described in this section.

For the sake of simplicity, we consider the system state given by $\mathbf{h}(t) = (h_1(t), h_2(t))^T$, whose evolution is governed by a simple linear first-order dynamical system

$$\begin{cases} \dot{h}_1(t) = K_{11}h_1(t) + K_{12}h_2(t) + l_1(t) \\ \dot{h}_2(t) = K_{21}h_1(t) + K_{22}h_2(t) + l_2(t) \end{cases}, \tag{31}$$

where, without loss of generality, it is assumed that $K_{22} = 0$ and $l_2(t) = 0$; i.e.,

$$\begin{cases} \dot{h}_1(t) = K_{11}h_1(t) + K_{12}h_2(t) + l_1(t) \\ \dot{h}_2(t) = K_{21}h_1(t) \end{cases}. \tag{32}$$

In what follows, it is assumed that $h_1(t)$ and $x_1(t)$ are accessible but that nothing, even the existence of $l_2(t)$, is known. In these circumstances, the key question is: can we learn a model relating $l_1(t)$ and $h_1(t)$ that is both accessible and measurable while knowing that the state $h_1(t)$ depends on another one, $h_2(t)$, that evolves in time and remains inaccessible and consequently unknown?

To facilitate the model manipulation, the Fourier transform can be applied, here denoted by the symbol * , to the different time-dependent variables. Thus, Equation (32) can be written as

$$\begin{cases} i\omega h_1^* = K_{11}h_1^* + K_{12}h_2^* + l_1^* \\ i\omega h_2^* = K_{21}h_1^* \end{cases} . \tag{33}$$

From the second part of Equation (33), we obtain

$$h_2^* = K_{21}\frac{h_1^*}{i\omega} \tag{34}$$

which, inserted into the first part of Equation (33), leads to

$$i\omega h_1^* = K_{11}h_1^* + K_{12}K_{21}\frac{h_1^*}{i\omega} + l_1^*, \tag{35}$$

Coming back to the time domain results in

$$\frac{dh_1(t)}{dt} = K_{11}h_1(t) + K_{12}K_{21}\int_0^t h_1(\tau)d\tau + l_1(t), \tag{36}$$

which proves that ignoring components of the state manifests in the measurable state history, here symbolized by the integral. In a certain way, this result can be interpreted as a consequence of Takens' delay embedding theorem.

Thus, memory is not only a consequence of the order of the subjacent dynamics; the memory in the learning procedure also depends on the ignored states. Thus, sometimes, when addressing poorly described physical systems or partially accessible ones, larger memory than that offered by recurrent NNs seems compulsory.

3.2.3. Learners with Larger Memory

Long short-term memory (LSTM) [28] is an appealing technique for learning time-dependent problems while ensuring larger memory. LSTM combines short- and long-term memory units, with an evanescent memory for the long-term path and a combination of long and short leads for the short memory response.

A particular architecture is depicted in Figure 2 that includes not only memory but also the previous inputs, as well as the forgetting of the initial conditions of the long $\mathbf{c}_0$ and short $\mathbf{h}_0$ memory channels to facilitate its use as a generic time integrator.

Recurrent NNs (rNNs) and LSTM learn the state evolution, and special care must be paid to ensure time integration stability.

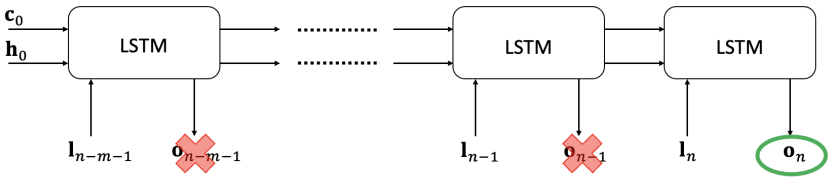

Figure 2. LSTM block architecture: The output $\mathbf{o}_n$ results from $\mathbf{l}_n$, the previous $m-1$ inputs, while the model almost totally forgets the initial hidden long and short memory states $\mathbf{h}_0$ and $\mathbf{c}_0$, which can be initialized with zero values.

Sometimes, it seems preferable to learn the forcing term of the dynamical system, as in residual nets and neural differential equation techniques. The next section revisits residual nets and discusses stability issues.

3.2.4. Residual Nets

Residual neural networks (ResNets) [29–31] aim at emulating the backward integration of a dynamical system. For the dynamical system

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}), \quad \mathbf{x}(t=0) = \mathbf{x}_0, \tag{37}$$

the ResNet looks for the function $f(\mathbf{x})$ that allows reproducing the state time evolution $\mathbf{x}(t)$ from its discrete knowledge; that is, from $\mathbf{x}_n = \mathbf{x}(t_n)$, with $t_n = n\Delta t$.

By considering the simplest time stepping, the time derivative can be approximated by $d\mathbf{x}/dt = (\mathbf{x}_{n+1} - \mathbf{x}_n)/\Delta t$. Then, one can use the following updating

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta t \mathbf{f}(\mathbf{x}_n), \tag{38}$$

and learn the forcing term $\mathbf{f}(\mathbf{x})$ that induced the state evolution by using an appropriate regression.

For this purpose, ResNet creates an NN trained from $\mathbf{x}_n$ and $\mathbf{x}_{n+1} - \mathbf{x}_n$, $n = 1, \ldots$. Then, as soon as the NN is trained, the output $\mathbf{x}_{n+1}$ is calculated by applying to $\mathbf{x}_n$ the NN just constructed, adding the identity operator applied to the input.

Now, the dynamical system integration consists in applying the ResNet in a recursive way, with the output at time n becoming the input of the next time step. The ResNet architecture and the integration procedure are illustrated in Figure 3.

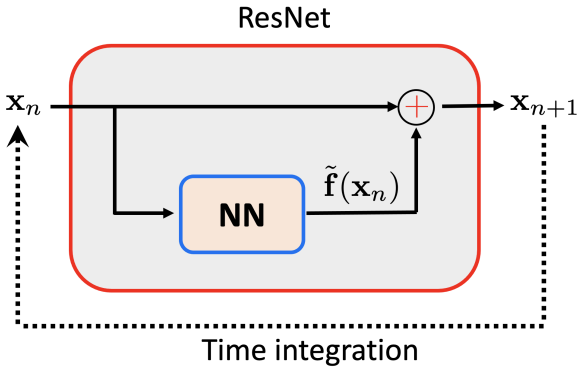

Figure 3. ResNet global architecture and integration mode (dotted line).

4. Simple Procedures for Stabilizing ResNet-Based Integration

In what follows, the stabilization of the time integration performed with ResNet is discussed. First, a simple stabilization is accomplished by linearizing the dynamical system while using the results just discussed.

Then, inspired by that accomplishment, a more general stabilization able to operate in nonlinear settings is proposed.

Finally, the performances of both strategies are compared.

4.1. Learning Stable Linear Dynamical Systems

In what follows, we consider a parameterized dynamical system:

$$\frac{dx}{dt} = f(x; \mathbf{z}), \tag{39}$$

where the state at time t_n, x_n, depends on the previous state x_{n-1} and the loading $f(\cdot)$ driving the state change. The last is assumed to depend not only on the state but also on a series of parameters (input features) here grouped as a vector $\mathbf{z}$. Thus, the learning procedure aims at computing the regression $f(x; \mathbf{z})$.

However, to obtain a stable integrator, certain constraints must be fulfilled. As these constraints have an easy form in the linear case, as previously discussed, we here proceed to linearize the forcing term as follows

$$f(x;\mathbf{z}) \approx g(\mathbf{z})x + h(\mathbf{z}). \tag{40}$$

Now, the stability is ensured as soon as $g(\mathbf{z}) \leq 0$, a constraint that can be easily enforced by employing a penalty term in the loss function of the NN associated with $g(\mathbf{z})$.

The linearized ResNet resembles dynamic mode decomposition (DMD) [32] with what is probably an easier introduction of the parametric dimension.

4.2. Learning Stable Nonlinear Dynamical Systems

Inspired by the rationale just discussed, a possible route for enforcing stability without detriment to the nonlinear behavior consists of writing

$$f(x;\mathbf{z}) \approx g(\mathbf{z},x)x + h(\mathbf{z}). \tag{41}$$

while enforcing in the construction of the regression $g(x;\mathbf{z})$ the negativity constraint; that is, $g(x;\mathbf{z}) \leq 0$.

4.3. Numerical Examples and Discussion

To prove the expected performances, we considered two cases: one linear, where both methodologies just described were expected to work, and a nonlinear one, where the former methodology was expected to fail and the latter (the nonlinear counterpart) was expected to performing correctly.

This section only addresses the stability concerns without giving details on the neural architectures employed for learning functions $g(\mathbf{z})$, $\mathbf{g}(x;\mathbf{z})$, and $h(\mathbf{z})$, which are described in detail in Section 5. Moreover, when the learned functions are employed for integrating the dynamical system incrementally by using standard first-order explicit time-marching, the computed solution will be denoted with the *hat* symbol ˆ.

4.3.1. Linear Dynamical System

We consider data generated from the integration of the dynamical system

$$\frac{dx}{dt} = 1 - x, \tag{42}$$

from the initial condition $x(t = 0) = 0$.

The long-term solution $x = 1$ is stable because, if $x < 1$, the solution increases, whereas if $x > 1$, it decreases.

The solution reads

$$x(t) = 1 - e^{-t}, \tag{43}$$

and is used for generating the synthetic data that will be used for learning the dynamical system with the two procedures previously described.

Figure 4 compares the solution obtained by integrating numerically the models learned by employing the linear and nonlinear learning procedures. The problem being linear, both procedures were expected to perform identically, as Figure 4 proves.

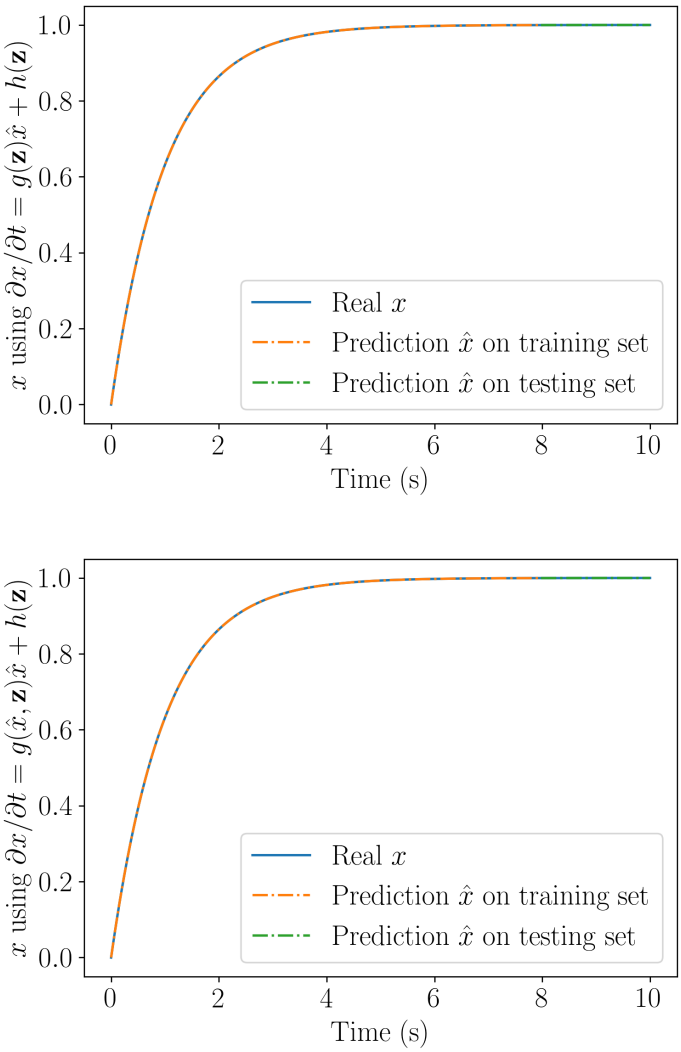

Figure 4. Solution computed by integrating the linear (**top**) and nonlinear (**bottom**) learning procedures.

4.3.2. Nonlinear Dynamical System

Next, a nonlinear dynamical system is considered expressed by

$$\frac{dx}{dt} = (1 - x)^2, \tag{44}$$

which is integrated from the initial condition $x(t = 0) = 0$.

Using the same rationale as before, we can reach a conclusion about the stability of the associated long-term solution $x = 1$.

Figure 5 compares the solutions obtained with the linear and nonlinear learning procedures. As expected, the nonlinear learning procedure ensures stability and accurately represents the nonlinear behavior, whereas the linear model learned results in stability but remains inaccurate.

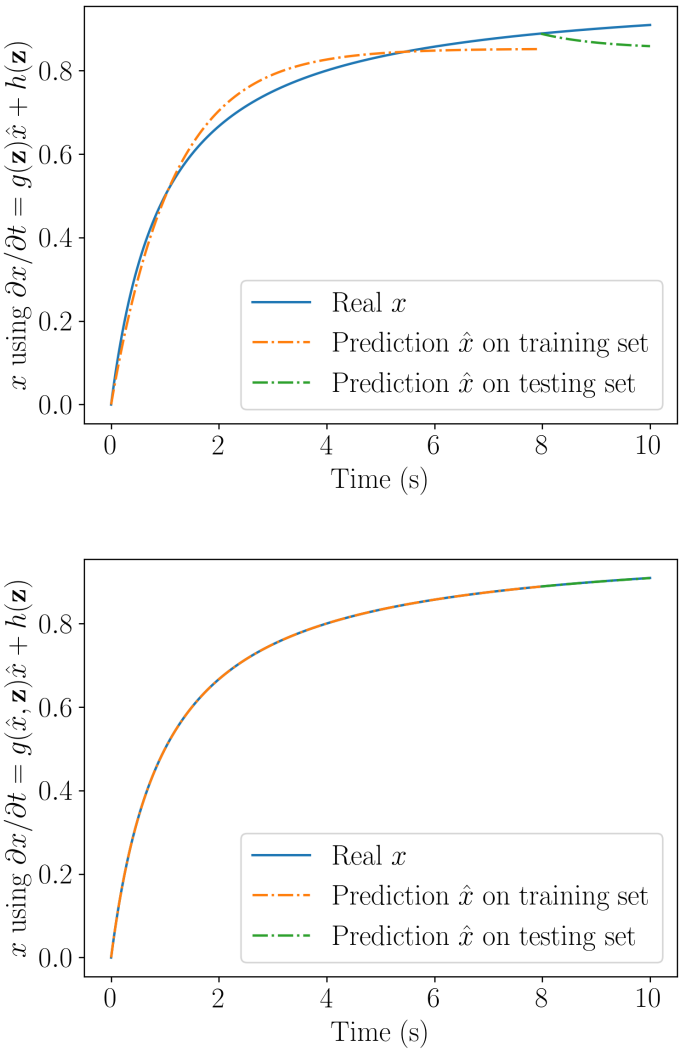

Figure 5. Solution computed by integrating the linear (**top**) and nonlinear (**bottom**) learning procedures.

4.4. Final Remarks

Another potential neural architecture with improved robustness with respect to stability is the NeuralODE [33–35], which learns a model while considering many integration steps instead of learning in one-step time increments as ResNet does. Even if stability is significantly enhanced, the NeuralODE's training efforts are greater than those required by the ResNet, mainly due to the fact that the back-propagation in the loss function minimization needs the solution of an adjoint problem, which, in some cases, exhibits poor convergence.

5. Application to the Evaluation of the Top-Oil Temperature of an Electric Power Transformer

This section addresses an application case of industrial relevance. Aging of transformers exhibits a correlation with the temperature of the oil throughout their lives in service. Moreover, the oil temperature seems to be an appealing feature to anticipate faults and, consequently, to be used in predictive maintenance.

Some simplified models exist to evaluate the oil temperature depending on the transformer's delivered power and the ambient temperature. Standards propose different simplified models for that purpose, such as the one considered later.

Due to the complexity of a transformer, which is large in size and embraces numerous types of physics and scales, an appealing modeling route consists of using a simplified model and then enriching it with the available collected data [36].

The correction just referred to comes from the time integration of a dynamical system involving parametric loading (delivered power and ambient temperature) learned from the available data under the stability constraints.

To illustrate the construction of the enriched model for the top-oil temperature prediction Θ, we consider as input the ambient temperature T^{amb} and the transformer load K^{load}, which are both measured every hour. Thus, the model parameters read $\mathbf{z} = (T^{amb}; K^{load})^T$.

The transformer oil temperature can be obtained from

$$\Theta(t) = \Theta^{TO}(t) + d(t), \tag{45}$$

where $d(t)$ represents the deviation, and Θ^{TO} the temperature predicted by a state-of-the-art simplified model [37]:

$$\begin{cases} P(K^{load}, \Theta^W, X^{tp}) = C_{th} \frac{d\Delta\Theta^{TO}}{dt} + \frac{\Delta\Theta^{TO}}{R_{th}} \\ \Theta^{TO} = T^{amb} + \Delta\Theta^{TO} \\ \Theta^W = \Theta^{TO} + \Delta\Theta^{OW} \end{cases} \tag{46}$$

where

- X^{tp} is the position of the tap-changer;
- K^{load} is the load factor, the ratio between the nominal load current and the actual current;
- P represents the iron, copper, and supplementary losses. The power that heats the oil is composed of the losses that do not depend on the transformer load (iron losses, assumed to be constant) and the losses that do depend on the transformer load (copper and supplementary losses), which depend on the average winding temperature and the load factor in accordance with: $P = P_0 + P_{load}$, with $P_{load} = {K^{load}}^2 P_{load_r}$ and $P_{load_r} = kP_{Joule_r} + \frac{P_{suppr}}{k}$ (k being a correction factor related to the material resistivity);
- T^{amb} is the ambient temperature;
- Θ^{TO} is the simulated top-oil temperature
- $\Delta\Theta^{TO}$ is the temperature difference between the simulated top-oil temperature and the ambient temperature;
- R_{th} and C_{th} are the thermal resistance and thermal capacitance of the equivalent transformer thermal circuit;
- Θ^W is the average winding temperature;
- $\Delta\Theta^{OW}$ is the difference between the average winding temperature and the simulated oil temperature Θ^{TO}. It is assumed to be constant and found during the commissioning test (standards).

The physics-based model (Equation (46)) is calibrated from the available experimental data, from which the parameters C_{th} and R_{th} are obtained. Two physics-based models are used, a linear one where C_{th} and R_{th} are assumed constant and a nonlinear physics-based model where C_{th} and R_{th} are temperature-dependent; that is, both coefficients depend on the top-oil temperature Θ^{TO}.

The available experimental data $\Theta(t)$, the prediction from the calibrated physics-based nonlinear model (Equation (46)) $\Theta^{TO}(t)$, and the deviation between them $d(t) = \Theta(t) - \Theta^{TO}$ are all depicted in Figure 6.

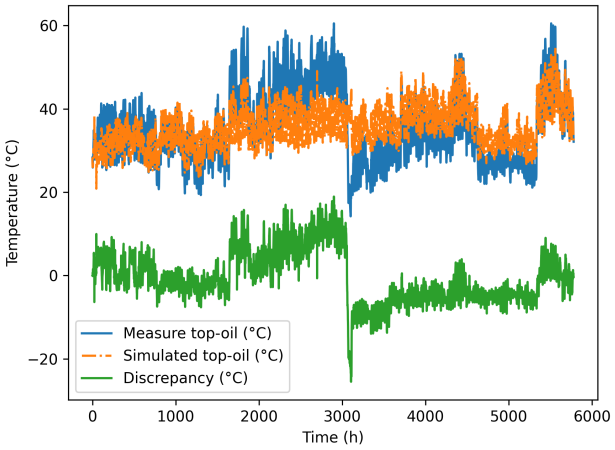

Figure 6. Experimental data $\Theta(t)$, simulated solution $\Theta^{TO}(t)$, and deviation $d(t) = \Theta(t) - \Theta^{TO}$.

The model correction (the deviation model $d(t)$) can also be obtained using two different approaches, the linearized ResNet:

$$\frac{\partial d}{\partial t} = g(\tilde{\mathbf{z}})d + h(\tilde{\mathbf{z}}), \tag{47}$$

and the nonlinear ResNet counterpart:

$$\frac{\partial d}{\partial t} = g(\tilde{\mathbf{z}}, d)d + h(\tilde{\mathbf{z}}), \tag{48}$$

where $\tilde{\mathbf{z}}$ represents the augmented feature set that contains the physical features $\mathbf{z}$ augmented with the model prediction Θ^{TO}.

Functions $g(\tilde{\mathbf{z}})$ and $h(\tilde{\mathbf{z}})$ are described by using two LSTM architectures (described in Tables 1 and 2, respectively, for the linearized ResNet) that consider the extended features involved in $\tilde{\mathbf{z}}$ at the present time, as well as at the previous four time steps.

Thus, the linearized-correction dynamical model reads

$$\begin{cases} d_n = \Delta t\, g(\tilde{\mathbf{z}})d_{n-1} + \Delta t\, h(\tilde{\mathbf{z}}) + d_{n-1} \\ \tilde{\mathbf{z}} = \begin{pmatrix} T^{amb}_{n-4} & K^{load}_{n-4} & \Theta^{TO}_{n-4} \\ T^{amb}_{n-3} & K^{load}_{n-3} & \Theta^{TO}_{n-3} \\ & \vdots & \\ T^{amb}_{n} & K^{load}_{n} & \Theta^{TO}_{n} \end{pmatrix} \end{cases}. \tag{49}$$

The neural network architectures considered for describing functions $g(\tilde{\mathbf{z}})$ and $h(\tilde{\mathbf{z}})$ are both based on the use of LSTM layers combined with a deep dense neural network layer, as described in Tables 1 and 2 for the linearized ResNet. They were built by using Tensorflow Keras libraries. The inputs involved in Equation (49) are shown in Figure 7.

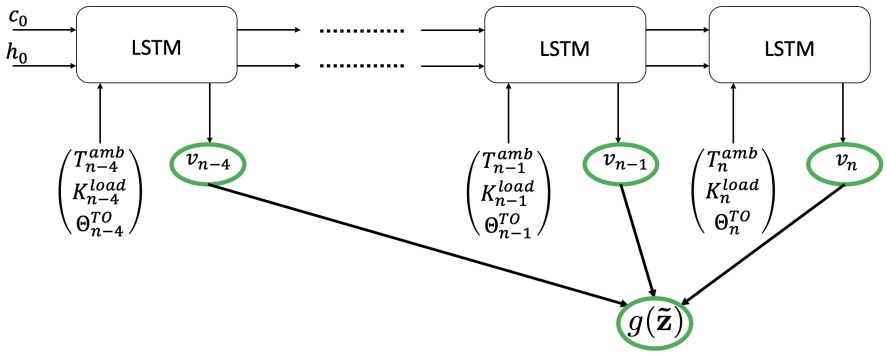

Figure 7. Network considered to model $g(\tilde{\mathbf{z}})$. Variables v_i are intermediate variables involved in the construction of $g(\tilde{\mathbf{z}})$. A similar architecture was considered for modeling $h(\tilde{\mathbf{z}})$.

Table 1. The building blocks of the LSTM-based surrogate of $g(\tilde{\mathbf{z}})$.

Layer	Building Block	Activation
1	LSTM layer with five outputs, return sequence true	sigmoid + tanh
2	Flatten	no activation
3	Dense connection with one output	relu
4	Lambda layer returning $-1\times$ inputs	no activation

Table 2. The building blocks of the LSTM-based surrogate of $h(\tilde{\mathbf{z}})$.

Layer	Building Block	Activation
1	LSTM layer with five outputs, return sequence true	sigmoid + tanh
2	Flatten	no activation
3	Dense connection with one output	linear

The training was performed with the initial 80% of the available measures while the testing was performed with the remaining 20%.

The prediction performed from the corrected (enriched) model is depicted in Figure 8 for the testing interval. It is important to note that the learned models $g(\tilde{\mathbf{z}})$ and $h(\tilde{\mathbf{z}})$ were used to integrate the dynamical problem that governs the time evolution of the deviation (Equation (49)); that is, the deviation at time t_i was computed from that at time t_{i-1}, and then it was used to compute the deviation at the next time step t_{i+1} and so on.

To distinguish between the known deviation $d(t)$ and the one computed from the integrator based on the learned dynamics $g(\tilde{\mathbf{z}})$ and $h(\tilde{\mathbf{z}})$, the later is denoted by the *hat* symbol; i.e.,

$$\hat{d}_{n+1} = \Delta t\, g(\tilde{\mathbf{z}})\hat{d}_n + \Delta t\, h(\tilde{\mathbf{z}}) + \hat{d}_n \tag{50}$$

From the computed correction, the model was enriched, exhibiting excellent accuracy and stability.

The fact that the linearized dynamical system operating with the solution correction exhibited good performances proves that most of the problem nonlinearities were captured by the first-order simplified model. When it comes to the nonlinear version of the correction effort, the dynamic version of the model is written in a similar manner as shown in Equation (49), with the dynamical integration form used:

$$\hat{d}_{n+1} = \Delta t\, g(\tilde{\mathbf{z}}, \hat{d}_n)\hat{d}_n + \Delta t\, h(\tilde{\mathbf{z}}) + \hat{d}_n \tag{51}$$

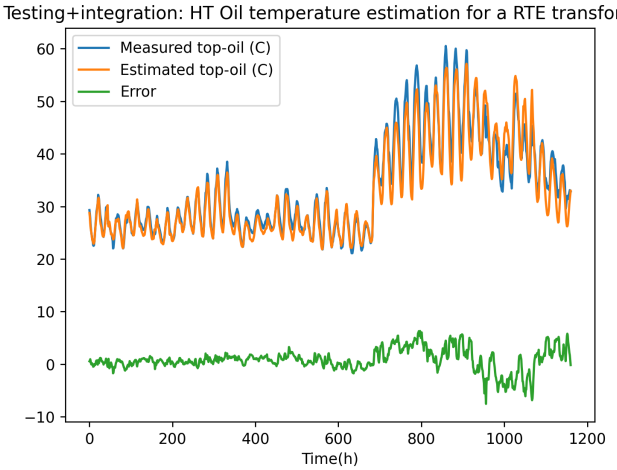

Figure 8. Physics-based simplified model correction from a stabilized linearized ResNet model, here illustrated with the test dataset.

To compare the performances of the hybrid model (data-driven enrichment of the simplified physics-based model), the model governing the experimental data evolution was learned using the same rationale but now applied to the data, as follows:

$$\begin{cases} \Theta_{n+1} = \Delta t g^{\theta}(\mathbf{z})\Theta_n + \Delta t h^{\theta}(\mathbf{z}) + \Theta_n \\ \mathbf{z} = \begin{pmatrix} T^{amb}_{n-4} & K^{load}_{n-4} \\ T^{amb}_{n-3} & K^{load}_{n-3} \\ & \vdots \\ T^{amb}_{n} & K^{load}_{n} \end{pmatrix} \end{cases}, \tag{52}$$

where $g^{\theta}(\mathbf{z})$ and $h^{\theta}(\mathbf{z})$ refer to the parametric functions related to the measured oil temperature, both depending exclusively on the input features $\mathbf{z}$.

Again, the training was performed using the same model presented in Tables 1 and 2 with the same 80% and 20% training and test datasets.

Figure 9 depicts the results obtained from the integration, where again the *hat* symbol refers the integrated temperature.

$$\hat{\Theta}_{n+1} = \Delta t g^{\theta}(\mathbf{z})\hat{\Theta}_n + \Delta t h^{\theta}(\mathbf{z}) + \hat{\Theta}_n. \tag{53}$$

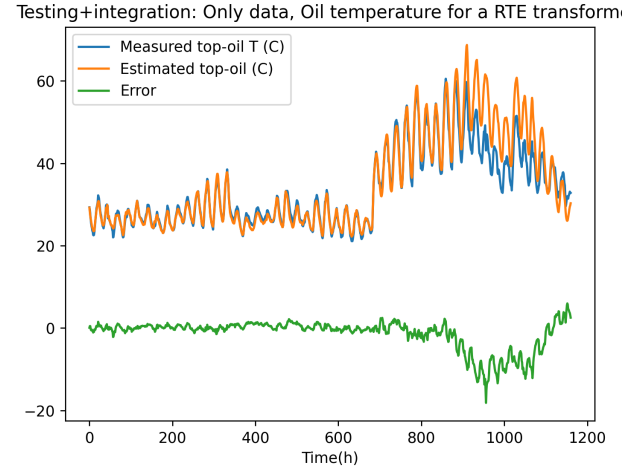

Figure 9. Full data-driven model consisting of a stabilized ResNet model, here illustrated with the test dataset.

Figure 9 depicts the data-driven model integration, proving the stability performances. However, when comparing the residual error (with respect to the experimental data) in the fully data-driven prediction and that related to the physics-based enriched solution obtained from the data-driven model of the deviation, both compared in Figure 10, the hybrid modeling framework performed better, ensuring higher accuracy.

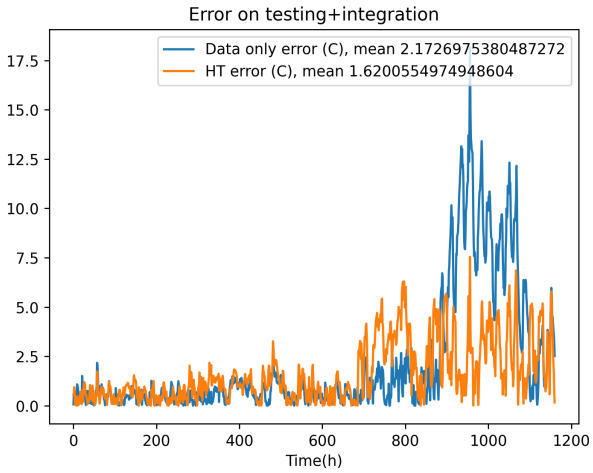

Figure 10. Comparing the errors of the data-only linearized ResNet and the linearized ResNet hybrid model with the test dataset.

Table 3 compares the mean values of the errors associated with the different tested models. From Table 3, one can infer that:

- The hybrid approach improves the physics-based model performances;
- Enriching the richer nonlinear physics-based model produced better results than enriching the linear counterpart of the simplified physics-based model;
- When the considered physics-based models were too far from the reference solution (experimental data), the data-driven model could outperform the hybrid modeling.

Table 3. Comparing different built models: mean error (in °C) with the testing set. The nonlinear physics-based model's mean error was 3.91 °C when used alone with the same testing set and that of the linear physics-based model was 3.25 °C.

ResNet	Fully Data-Driven	HT from a Linear Physical Model	HT from a Nonlinear Physical Model
Linear stabilized	2.173	3.143	1.620
Nonlinear stabilized	1.716	1.516	1.439

To show the effect of the stabilization, a ResNet was trained without enforcing the stability constraints previously proposed by using the formulation:

$$\frac{\partial d}{\partial t} = g(\tilde{\mathbf{z}}, d) + h(\tilde{\mathbf{z}}), \tag{54}$$

with no conditions imposed during the calculation of g and h.

The discrete form

$$\hat{d}_{n+1} = \Delta t\, g(\tilde{\mathbf{z}}, d_n) + \Delta t\, h(\tilde{\mathbf{z}}) + d_n, \tag{55}$$

provided excellent predictions. It is important note that, here, the solution at time t_{n+1}, $\hat{d}_{n+1}$, is computed from the exact deviation at time t_n, d_n.

However, a full integration where $\hat{d}_{n+1}$ is computed from the previously computed $\hat{d}_n$

$$\hat{d}_{n+1} = \Delta t\, g(\tilde{\mathbf{z}}, \hat{d}_n) + \Delta t\, h(\tilde{\mathbf{z}}) + \hat{d}_n, \tag{56}$$

produces extremely bad predictions, a direct consequence of the lack of stability.

Figure 11 proves the importance of using stable formulations when the learned model is expected to serve for performing integrations from an initial condition, as is always the case in prognosis applications.

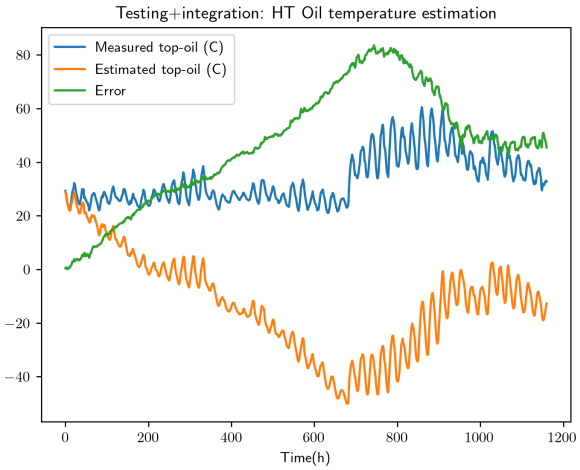

Figure 11. Integration performed from a ResNet learned without enforcing stability constraints.

6. Conclusions

The present paper proposed a hybrid framework where the data-driven model serves to enrich a physics-based model considered as a first approximation of the addressed physics.

We proved that the hybrid framework enhances the prediction accuracy with respect to the physics-based model; however, the hybrid approach is superior to a fully data-driven model only under certain conditions.

The learning technique employed to model the time evolution of the deviation (or that of the data) must ensure the integration stability. A stabilization was proposed and its performance proved.

An application to a problem of practical relevance proved the excellent performances of the proposed methodology.

Author Contributions: Conceptualization, F.C.; methodology, C.G. and V.C.; software, C.G. and V.C.; validation, X.K. and G.D.; resources, G.D.; writing—original draft preparation, C.G. and F.C.; writing—review and editing, C.G., F.C. and X.K.; supervision, X.K. and G.D.; project administration, G.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are available upon request to the authors after the agreement of the RTE.

Acknowledgments: The authors acknowledge the support of the RTE, SKF, and ESI research chairs at Arts et Métiers Institute of Technology.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Borzacchiello, D.; Aguado, J.V.; Chinesta, F. Non-intrusive sparse subspace learning for parametrized problems. *Arch. Comput. Methods Eng.* **2019**, *26*, 303–326. [CrossRef]
2. Settles, B. Active Learning Literature Survey. In *Computer Sciences Technical Report 1648*; University of Wisconsin-Madison: Madison, WI, USA, 2009.
3. Sancarlos, A.; Champaney, V.; Cueto, E.; Chinesta, F. Regularized regressions for parametric models based on separated representations. *Adv. Model. Simul. Eng. Sci.* **2023**, *10*, 4. [CrossRef]
4. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
5. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]

6. Cristianini, N.; Shawe-Taylor, J. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*; Cambridge University Press: New York, NY, USA, 2000.
7. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
8. Kirkwood, C.W. Decision Tree Primer. 2002. Available online: https://www.public.asu.edu/~kirkwood/DAStuff/refs/decisiontrees/index.html (accessed on 24 June 2023).
9. Chinesta, F.; Cueto, E.; Abisset-Chavanne, E.; Duval, J.L.; Khaldi, F.E. Virtual, Digital and Hybrid Twins: A New Paradigm in Data-Based Engineering and Engineered Data. *Arch. Comput. Methods Eng.* **2020**, *27*, 105–134. [CrossRef]
10. Sancarlos, A.; Cameron, M.; Abel, A.; Cueto, E.; Duval, J.L.; Chinesta, F. From ROM of electrochemistry to AI-based battery digital and hybrid twin. *Arch. Comput. Methods Eng.* **2021**, *28*, 979–1015. [CrossRef]
11. Sancarlos, A.; Cameron, M.; Peuvedic, J.M.L.; Groulier, J.; Duval, J.L.; Cueto, E.; Chinesta, F. Learning stable reduced-order models for hybrid twins. *Data Centric Eng.* **2021**, *2*, e10. [CrossRef]
12. Reille, A.; Champaney, V.; Daim, F.; Tourbier, Y.; Hascoet, N.; Gonzalez, D.; Cueto, E.; Duval, J.L.; Chinesta, F. Learning data-driven reduced elastic and inelastic models of spot-welded patches. *Mech. Ind.* **2021**, *22*, 32. [CrossRef]
13. Gonzalez, D.; Chinesta, F.; Cueto, E. Learning corrections for hyper-elastic models from data. *Front. Mater.-Sect. Comput. Mater. Sci.* **2019**, *6*, 14.
14. Quaranta, G.; Ziane, M.; Haug, E.; Duval, J.L.; Chinesta, F. A minimally-intrusive fully 3D separated plate formulation in computational structural mechanics. *Adv. Model. Simul. Eng. Sci.* **2019**, *6*, 11. [CrossRef]
15. Moya, B.; Badias, A.; Alfaro, I.; Chinesta, F.; Cueto, E. Digital twins that learn and correct themselves. *Int. J. Numer. Methods Eng.* **2022**, *123*, 3034–3044. [CrossRef]
16. Torregrosa, S.; Champaney, V.; Ammar, A.; Hebert, V.; Chinesta, F. Surrogate Parametric Metamodel based on Optimal Transport. *Math. Comput. Simul.* **2022**, *194*, 36–63. [CrossRef]
17. Torregrosa, S.; Champaney, V.; Ammar, A.; Herbert, V.; Chinesta, F. Hybrid Twins based on Optimal Transport. *Comput. Math. Appl.* **2022**, *127*, 12–24. [CrossRef]
18. Ibanez, R.; Abisset-Chavanne, E.; Gonzalez, D.; Duval, J.L.; Cueto, E.; Chinesta, F. Hybrid Constitutive Modeling: Data-driven learning of corrections to plasticity models. *Int. J. Mater. Form.* **2019**, *12*, 717–725. [CrossRef]
19. Argerich, C.; Carazo, A.; Sainges, O.; Petiot, E.; Barasinski, A.; Piana, M.; Ratier, L.; Chinesta, F. Empowering Design Based on Hybrid Twin: Application to Acoustic Resonators. *Designs* **2020**, *4*, 44. [CrossRef]
20. Casteran, F.; Delage, K.; Cassagnau, P.; Ibanez, R.; Argerich, C.; Chinesta, F. Application of Machine Learning tools for the improvement of reactive extrusion simulation. *Macromol. Mater. Eng.* **2020**, *305*, 2000375. [CrossRef]
21. Ghanem, R.; Soize, C.; Mehrez, L.; Aitharaju, V. Probabilistic learning and updating of a digital twin for composite material systems. *Int. J. Numer. Methods Eng.* **2022**, *123*, 3004–3020. [CrossRef]
22. Ghnatios, C.; Gérard, P.; Barasinski, A. An advanced resin reaction modeling using data-driven and digital twin techniques. *Int. J. Mater. Form.* **2023**, *16*, 5. [CrossRef]
23. Kapteyn, M.G.; Willcox, K.E. From Physics-Based Models to Predictive Digital Twins via Interpretable Machine Learning. *arXiv* **2020**, arXiv:2004.11356v3.
24. Tuegel, E.J.; Ingraffea, A.R.; Eason, T.G.; Spottswood, S.M. Reengineering Aircraft Structural Life Prediction Using a Digital Twin. *Int. J. Aerosp. Eng.* **2011**, *2011*, 154798. [CrossRef]
25. Distefano, G.P. Stability of numerical integration techniques. *AIChE J.* **1968**, *14*, 946–955. [CrossRef]
26. Dar, S.H.; Chen, W.; Zheng, F.; Gao, S.; Hu, K. An LSTM with Differential Structure and Its Application in Action Recognition. *Math. Probl. Eng.* **2022**, *2022*, 7316396.
27. Zhou, G.B.; Wu, J.; Zhang, C.L.; Zhou, Z.H. Minimal gated unit for recurrent neural networks. *Int. J. Autom. Comput.* **2016**, *13*, 226–234. [CrossRef]
28. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
29. Blaud, P.C.; Chevrel, P.; Claveau, F.; Haurant, P.; Mouraud, A. Resnet and polynet based identification and (mpc) control of dynamical systems: A promising way. *IEEE Access* **2022**, *11*, 20657–20672. [CrossRef]
30. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
31. He, K.; Zhang, X.; Ren, S.; Sun, J. Identity Mappings in Deep Residual Networks. In *Computer Vision—ECCV 2016*; ECCV 2016. Lecture Notes in Computer Science; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; Volume 9908.
32. Schmid, P.J. Dynamic mode decomposition of numerical and experimental data. *J. Fluid Mech.* **2010**, *656*, 528. [CrossRef]
33. Chen, R.T.Q.; Rubanova, Y.; Bettencourt, J.; Duvenaud, D. Neural ordinary differential equations. In Proceedings of the Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, QC, Canada, 2–8 December 2018; Volume 32, pp. 1–18.
34. Chen, R.T.Q.; Amos, B.; Nickel, M. Learning Neural Event Functions for Ordinary Differential Equations. *arXiv* **2020**, arXiv:2011.03902.
35. Enciso-Salas, L.; Perez-Zuniga, G.; Sotomayor-Moriano, J. Fault Detection and Isolation for UAVs using Neural Ordinary Differential Equations. *IFAC-PapersOnLine* **2022**, *55*, 643–648. [CrossRef]

36. Kestelyn, X.; Denis, G.; Champaney, V.; Hascoet, N.; Ghnatios, C.; Chinesta, F. Towards a hybrid twin for infrastructure asset management: Investigation on power transformer asset maintenance management. In Proceedings of the 7th International Advanced Research Workshop on Transformers (ARWtr), Baiona, Spain, 23–26 October 2022; pp. 109–114.
37. *IEC 60076-7:2018*; Loading Guide for Mineral-Oil-Immersed Power Transformers. International electrotechnical Commission: Geneva, Switzerland, 2018.