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 , Xavier Kestelyn , Guillaume Denis , Victor Champaney , Francisco Chinesta [*]

Posted Date: 27 June 2023

doi: 10.20944/preprints202306.1895.v1

Keywords: Stable integrator; Hybrid Twin; Machine Learning; Dynamical system; Power transformer; Monitoring

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, Franscisco.Chinesta@ensam.eu
[6] CNRS@CREATE,1 Create way, 04-05 Create Tower, Singapore 138602
[*] Correspondence: Francisco.Chinesta@ensam.eu (F.C.)
[‡] All authors contributed equally to this work.

Abstract: Many engineering systems are 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 learnt model remains stable. The proposed methodology is applied on the simulation of the top-oil temperature evolution of a power transformer, whose data is provided by RTE, the French Electricity Transmission System Operator.

Keywords: Stable integrator; Hybrid Twin; Machine Learning; Dynamical system; Power transformer; Monitoring

1. Introduction

The two most usual approaches involved in simulation based engineering –SBE–, the one mainly based on physics, and the more recent one, the one based on the use of data manipulated by advanced machine learning techniques, both present their inherent limitations.

In general, the physics-based modeling framework produces responses that approximate quite well the real ones, as soon as an accurate enough model exists. The main difficulties when considering such a physics-based approach are: (i) the fidelity of the model itself, assumed calibrated; (ii) the impact that variability and uncertainty induces; 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 noise-free) represent well the reality, its extrapolation or interpolation in space and time from the collected data (in particular locations and times instants) entail usually a noticeable accuracy loss.

Of course, by increasing the number of collected data, one could expect approximating the real solution with more fidelity, however, data is not always simple to collect, not always possible to access, and in all cases collecting data is expensive (cost of sensors, cost of communication and analysis, ...). 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 was collected.

In view of the limitations of both existing procedures, a 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 deviation, gap, discrepancy, or simply, ignorance).

1.1. The three main SBE methodologies revisited

To introduce and discuss different simulation based engineering 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 it exists a certain knowledge on the addressed physics, described by the model $\mathcal{M}$, that 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 solvability. The solution provided by the just referred model, that as indicated, represents the existing knowledge on the problem at hand, is represented by $u^M(\mathbf{x})$.

Due to the partial knowledge on 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 reducing that error different possibilities exist:

- **Physics-based model improvement**. This approach consists of refining the modelling, by enriching the model itself, $\mathcal{M} \to \widehat{\mathcal{M}}$, 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 making a large sampling of the space Ω, $\mathbf{x}_1, ..., \mathbf{x}_{\mathtt{P}}$, with $\mathtt{P}$ large enough, and with the points location $\mathbf{x}_i$, $i = 1, ..., \mathtt{P}$, maximizing the domain Ω coverage. These points are grouped into the set $\mathcal{S}$.
  The coverage is defined by the convex hull $\omega(\mathcal{S})$ of the set $\mathcal{S} = \{\mathbf{x}_1, ..., \mathbf{x}_{\mathtt{P}}\}$, ensuring interpolation for $\mathbf{x} \in \omega(\mathcal{S})$, 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 the ones based on the use of Gauss-Lobatto quadratures, related to approximations making use of orthogonal polynomials [3], fail when the dimensionality of the space $\mathtt{D}$ increases.
  When $\mathtt{D} \gg 1$, sparse sampling is preferred, LHP (Latin hyper cube) for instance. Samplings based on gaussian processes –GP– 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 aiming at maximizing the representation of a certain goal oriented quantity of interest [32].
  In what follows, we assume a generic sampling to access to the reference solution $u(\mathbf{x}_i)$, $i = 1, ..., \mathtt{P}$, assuming a perfect measurability.

Now, to infer the solution at $\mathbf{x} \in \omega(\mathcal{S})$, it suffices constructing an interpolation or approximation, more generally, an adequate regression $u^{\mathcal{S}}(\mathbf{x})$:

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

with $u_1 \equiv u(\mathbf{x}_1), ..., u_{\mathtt{P}} \equiv u(\mathbf{x}_{\mathtt{P}})$.
Different possibilities exist, among them: regularized polynomial regressions [29], neural networks –NN– [16,31], support vector regression –SVR– [9], decision trees and their random forest counterparts [4,23], ... to name few.
The trickiest issue concerns the error evaluation, that is quantified from a part of the data kept outside the training-set, the so-called test-set, used to quantify the performances of the trained regression.
The main drawbacks 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 data-set ($\mathtt{P}$), scaling with the problem dimensionality $\mathtt{D}$.
– The optimal sampling to cover Ω while guaranteeing the accuracy of $u^{\mathcal{S}}(\mathbf{x})$, or the one of the goal oriented quantities of interest.

- **Hybrid approach**. It proceeds by embracing the physics-based and data-driven approaches, that as described in the next section, will improve the physics-based accuracy (while profiting of the physics-based explanatory capabilities) from the use of a data-driven enrichment, that at its turn, and under certain conditions, needs less amount of data than the fully data-driven approach just discussed [6].

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 representing reasonably the physics at hand, and noted by $u(\mathbf{x})$, and that was noted previously 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 $\mathbf{x}_i \in \Omega,\ i = 1, ..., \mathtt{P}$.

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

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

that involve the unknown coefficients $a_0^{\Delta}, a_1^{\Delta}, ..., a_{\mathtt{D}}^{\Delta}$.

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

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

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

If we assume for a while that $u(\mathbf{x})$ is fully available, and then $R^M(\mathbf{x})$ too, the previous expression can be rewritten in a continuous form

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

The linear regression of the reference solution $u(\mathbf{x})$, noted by $u^{\mathtt{L}}(\mathbf{x}; a_0^u, ..., a_{\mathtt{D}}^u)$, reads at its turn

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

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

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

whose continuous expression reads

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

Eq. (5) implies

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

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

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

then

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

that 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 addressed considers a simple linear regression involving a linear approximation, however it remains valid when considering regressions involving richer nonlinear approximations.

The first part of Eq. (11), i.e.

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

affirms that the simulated model solution is enriched in a L2-norm sense, and under the constraint of having enough amount of 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 Eq. (11), i.e.

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

works under some conditions involving the data and the model, i.e. on $u(x)$ and $u^M(x)$, to be checked, and again needs having an amount of data enabling the calculation of the L2-norms.

2.3. On the domain of application of the hybrid modeling approach

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

This hybrid approach seem particularly appealing in different situations, 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 treated in [27,28], where the same amount of data performed better within the hybrid 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 that can, in that case, be captured by a data-driven learned model, as considered in [26] for addressing the inelastic behavior of spot-welds.
- When considering the constitutive modeling of materials, the augmented rationale (or hybrid paradigm) expresses the real behavior from a first order behavior (calibrated from the available data) complemented by an enrichment (or correction) filling the gap between the collected data and the predictions obtained from the assumed model [15].
- When addressing plates (or shells) which 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 [25], that can be learned from data.
- The hybrid modeling can also transfer the existing knowledge slightly outside its domain of applicability with small amount of collected data, as performed in [24] for correcting state-of-the-art structural beam models.
- Sometimes the discrepancy concerns an imperfect alignment of the solution between the prediction and the measures. That discrepancy seems 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 [34,35].
- In [20] 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, of different granularities, can exhibit advantages (amount of data, ability to explain, knowledge transfer, ...), at the heart of the digital twin developments [1,5,6,13,14,21,36].

3. Methods

In what follows, a system characterized by the state x that evolves in time, i.e. $x(t)$, is considered, and the stability of its numerical integration discussed [11]. When the state is multi-valued it will be noted by $\mathbf{x}$.

3.1. On the integration stability

We consider for a while 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 noticed 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 writes $a \leq 0$.

Now, 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, it finally results

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

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 $\mathbf{x} = \mathbf{P}\mathbf{y}$, it results

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

that premultiplying by $\mathbf{P}^T$ results

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

that 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 $\mathbf{D}$ diagonal.

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

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

with y_i 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 Eq. (20), it results

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

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

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

that using the recurrence results in

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

Thus, stability requires the spectral radius ρ of $\hat{\mathbf{D}}$ being 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 know, 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, 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.

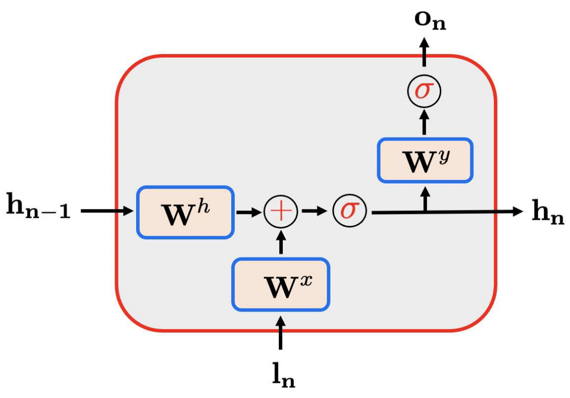


Figure 1. Recurrent Neural Network architecture

In what follows, we revisit, and discuss, different machine learning techniques that enable learning dynamical systems and applying them for the integration of dynamical systems, as recurrent NN –rNN– and long short time memory –LSTM– techniques perform [10,37].

In this section **h** refers to the hidden state, whereas **l** and **o** refer, respectively, to the inputs (loading) and observable outputs.

3.2.1. Recurrent Neural Network

If $\mathbf{W}^\bullet$ represents dense matrices associated to 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:

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

whose architecture is depicted in Fig. 1.

To perform 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, assumed available, allows calculating the different weights composing the different **W** matrices.

To highligth the connexion between the rNN and usual dynamical system integrators, we consider for a while 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 **M** and **K** are two matrices that results from the spatial discretization. Here, **h** refers to the state and **l** to the system loading.

Then, the explicit discretization of Eq. (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

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

that can be rewritten as

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

that correspond to the particularization of Eq. (26) to the linear case. Thus, the intimate connexion between rNN and usual discretization techniques becomes explicit. The last operates on a known model whereas the former learns the model itself (the different $\mathbf{W}^\bullet$ matrices).

Remark 1. In the just discussed linear model, 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 the present time can be computed from the only 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, as for instance the one of the just described rNN, suffices.

However, sometimes, larger memory are 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

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

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

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

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 that circumstances the key question is: can we learn a model relating $l_1(t)$ and $h_1(t)$, both accessible and measurable, 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 applies, here noted by the symbol $\cdot^*$, acting on the different time-dependent variables. Thus, Eq. (32) writes

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

From the second equation in (33) we obtain

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

that inserted in the first equation in (33) leads to

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

that coming back to the time domain results

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

that 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 the Taken 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, larger memory than the one that recurrent NN offers, seems compulsory.

3.2.3. Learners with larger memory

LSTM (Long Short Time Memory) [19] becomes an appealing technique for learning time-dependent problems while ensuring larger memory. LSTM combines a short and a long time memory units, with an evanescent memory for the long-time path, and a combination of long and short leads to the short memory response..

A particular architecture is depicted in Fig. 2, that is considered for including not only memory, but also the previous inputs as well as forgetting 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 NN (rNN) and LSTM learn the state evolution, and special care must be paid to ensure time integration stability.

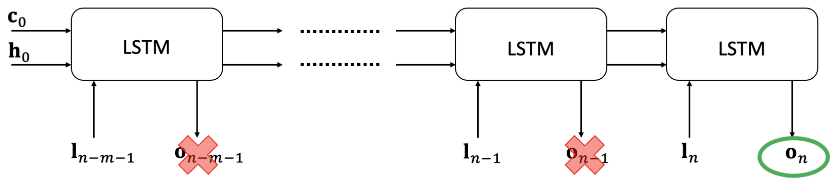


Figure 2. LSTM-block architecture: The output $\mathbf{o}_n$ results from $\mathbf{I}_n$ the previous $m-1$ inputs while it almost totally forget the initial hidden long and short memory states $\mathbf{h}_0$ and $\mathbf{c}_0$ that can be initialized with zero values.

Sometimes it seems preferably learning the forcing term of the dynamical system, as Residual Nets or Neural Differential Equations techniques perform. Next section revisits Residual Nets and discuss stability issues.

3.2.4. Residual Nets

Residual Neural Networks (ResNet) [2,17,18] aim at emulating the backward integration of a dynamical system. When we consider 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 $d\mathbf{x}/dt = (\mathbf{x}_{n+1} - \mathbf{x}_n)/\Delta t$, and 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 that purpose ResNet creates a NN trained from $\mathbf{x}_n$ and $\mathbf{x}_{n+1} - \mathbf{x}_n$, $n = 1, ...$ Then, as soon as the NN is trained, the output $\mathbf{x}_{n+1}$ is calculated by applying on $\mathbf{x}_n$ the NN just constructed adding the identity operator applied on the input.

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

4. Simple procedures for stabilizing ResNets-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 of the results just discussed.

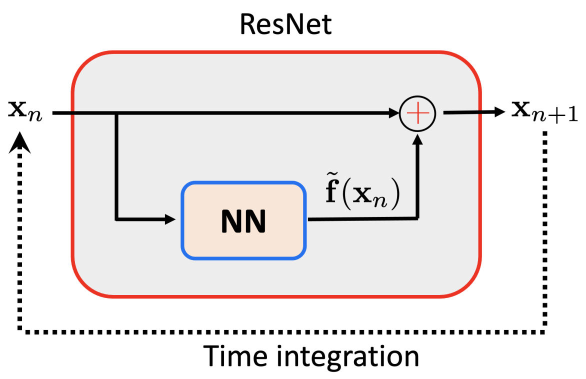


Figure 3. ResNet global architecture and integration mode (dotted line)

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 parametrized 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 depending not only on the state, but also on a series of parameters (input features) here grouped in vector $\mathbf{z}$. Thus, the learning procedure aims at computing the regression $f(x;\mathbf{z})$.

However, for obtaining a stable integrator, certain constraints must be fulfilled. As these constraints have an easy form in the linear case, as previously discussed, here we 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 $g(\mathbf{z}) \leq 0$, 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 to the DMD (dynamic mode decomposition) [30], with probably an easier introduction of the parametric dimension.

4.2. Learning stable nonlinear dynamical systems

Inspired from the just discussed rationale, a possible route for enforcing stability without detriment of 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 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 consider two cases, one linear, where both methodologies just described should work, and a nonlinear one, where the former is expected failing, whereas the later (the nonlinear counterpart) is expected 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 $h(\mathbf{z})$, that will be described in detail in Section 5. Moreover, when the learnt functions are employed for integrating the dynamical system incrementally, by using a standard first order explicit time-marching, the computed solution will be noted with *hat* symbol, $\hat{\cdot}$.

4.3.1. Linear dynamical system

We consider data generated from the integration of the dynamical system

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

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

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

The solution reads

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

that is used for generating the synthetic data that will be used then, for learning the dynamical system by using the two procedures previously described.

Figure 4 compares the solution obtained by integrating numerically the models learnt by employing both, the linear and nor linear learning procedures. The problem being linear both procedures are expected performing identically, as Fig. 4 proves.

4.3.2. Nonlinear dynamical system

Now, a nonlinear dynamical system is considered, expressed by

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

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

Using the same rationale than before, we can conclude on the stability of the associated long-time solution $x = 1$.

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

4.4. Final remarks

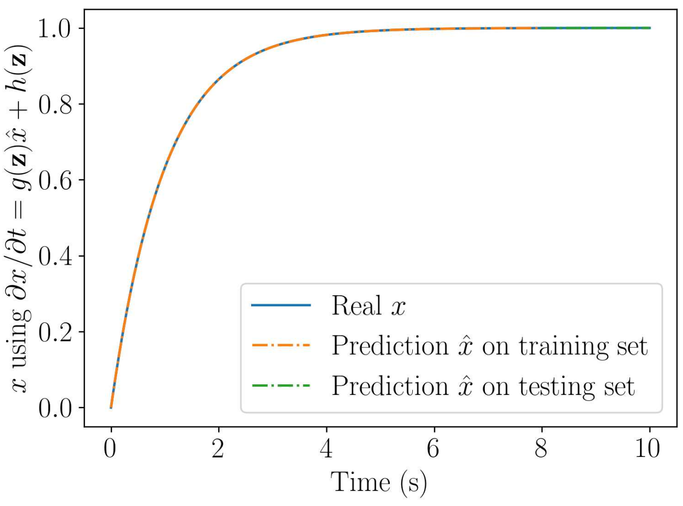
Another potential neural architecture with improved robustness with respect to stability is the NeuralODE [7,8,12], that learns the model while considering many integration steps, instead of learning for one-step time increments as ResNet performs. Even is stability is significantly enhanced, the NeuralODE training efforts are higher than the ones 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, that, 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 applicative case of industrial relevance. Aging of transformers exhibits a correlation with the temperature of the oil all along their lives in service, and 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 delivered power and the ambient temperature. Standards propose different simplified models for that purpose, as the one considered later.

Because of the complexity of a transformer, large in size and embracing numerous physics and scales, an appealing modeling route consists of using a simplified model and then enrich it from the available collected data [22].



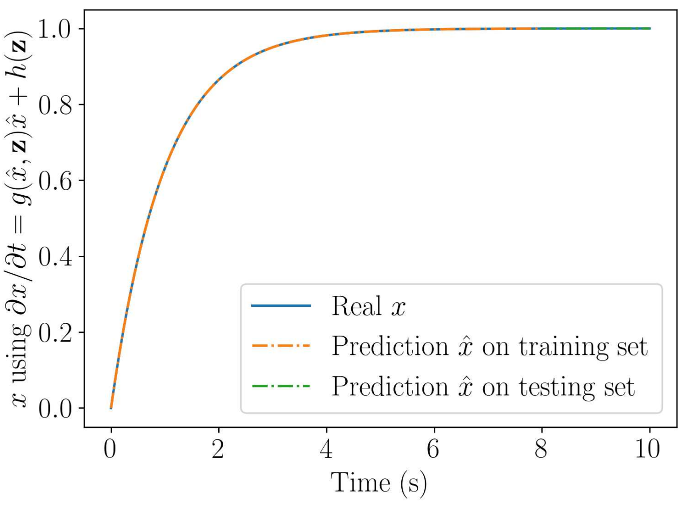
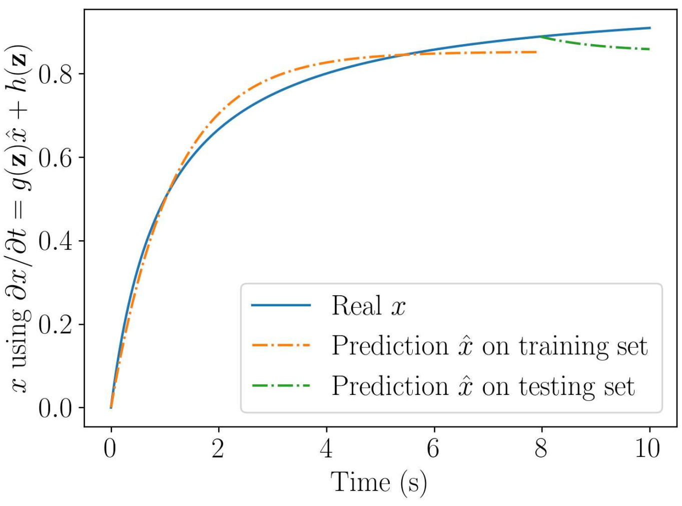


Figure 4. Solution computed by integrating the linear (top) and nonlinear (bottom) learning procedures.



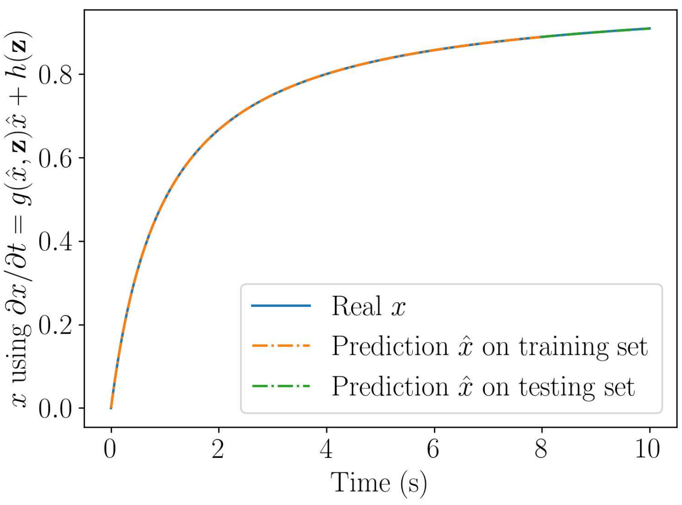


Figure 5. Solution computed by integrating the linear (top) and nonlinear (bottom) learning procedures.

The just referred correction comes from the time-integration of a dynamical system, involving a parametric loading (delivered power and ambient temperature), learnt 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 results 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 [33]:

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

where

- X^{tp}: The position of the tap-changer;
- K^{load}: the load factor, a 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, supposed constant) and the losses that depend on the transformer load (copper and supplementary losses) that depend on the average windings temperature and the load factor, according to: $P = P_0 + P_{load}$, with $P_{load} = {K^{load}}^2 P_{load_r}$ and $P_{load_r} = kP_{Joule_r} + \frac{P_{suppr_r}}{k}$ (k being a correction factor related to the material resistivity);
- T^{amb}: The ambient temperature;
- Θ^{TO}: The simulated top oil temperature
- $\Delta\Theta^{TO}$: The temperature difference between the simulated top oil temperature and the ambient temperature;
- R_{th} and C_{th}: thermal resistance and thermal capacitance of the equivalent transformer thermal circuit;
- Θ^{W}: The average winding temperature;
- $\Delta\Theta^{OW}$: The difference between the average winding temperature and the simulated oil temperature Θ^{TO}. It is supposed constant and found during the commissioning test (standards)

The physics-based model (46) is calibrated from the available experimental data, from which, 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 non-linear model (46), $\Theta^{TO}(t)$, and the deviation between both them, $d(t) = \Theta(t) - \Theta^{TO}$, are all depicted in Fig. 6.

The model correction (the deviation model, $d(t)$) is obtained also 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:

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 27 June 2023 doi:10.20944/preprints202306.1895.v1

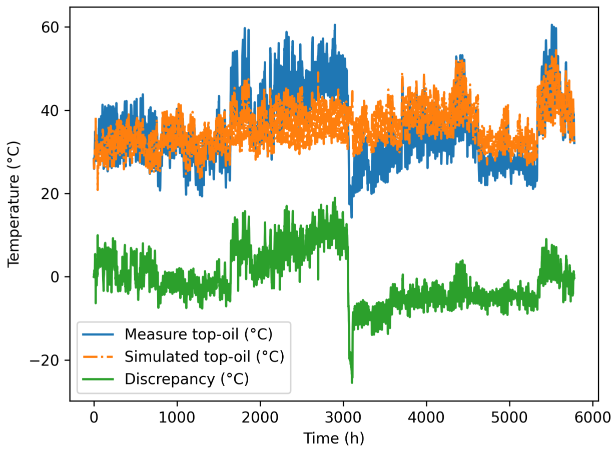


Figure 6. Experimental data $\Theta(t)$, simulated solution $\Theta^{TO}(t)$ and deviation $d(t) = \Theta(t) - \Theta^{TO}$.

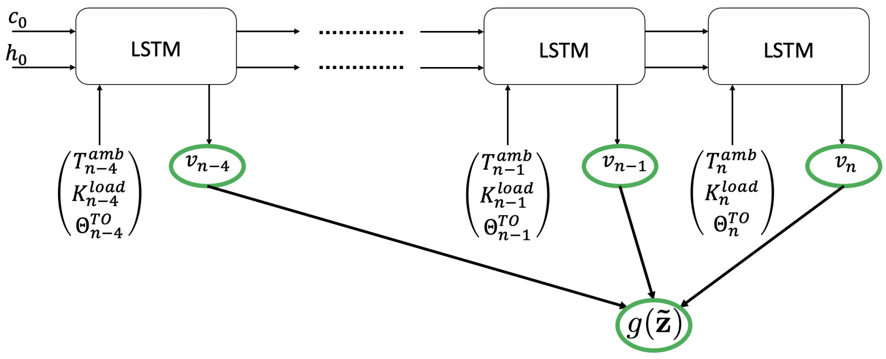

c_0
h_0
LSTM
LSTM
LSTM
$\begin{pmatrix} T^{amb}_{n-4} \\ K^{load}_{n-4} \\ \Theta^{TO}_{n-4} \end{pmatrix}$
v_{n-4}
$\begin{pmatrix} T^{amb}_{n-1} \\ K^{load}_{n-1} \\ \Theta^{TO}_{n-1} \end{pmatrix}$
v_{n-1}
$\begin{pmatrix} T^{amb}_{n} \\ K^{load}_{n} \\ \Theta^{TO}_{n} \end{pmatrix}$
v_n
$g(\tilde{\mathbf{z}})$

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 is considered for modeling $h(\tilde{\mathbf{z}})$.

where $\tilde{\mathbf{z}}$ represents the augmented features 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 4 time-steps.

Thus, the linearized correction dynamical model reads

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

The neural network architecture 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-up by using Tensorflow Keras libraries. The inputs involved in Eq. (49) are shown in Fig. 7.

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

The prediction performed from the corrected (enriched) model is depicted in Fig. 8, on the testing interval. It is important to note, that the learnt 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,

Layer	Building block	Activation
1	LSTM layer with 5 outputs, return sequence true	*sigmoid+tanh*
2	Flatten	No activation
3	Dense connection with 1 output	*relu*
4	Lambda layer returning $-1 \times inputs$	No activation

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

Layer	Building block	Activation
1	LSTM layer with 5 outputs, return sequence true	*sigmoid+tanh*
2	Flatten	No activation
3	Dense connection with 1 output	*Linear*

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

Eq. (49), that is, the computed deviation at time t_i was computed from the one at time t_{i-1}, and then it served to compute the one 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 learnt dynamics $g(\tilde{\mathbf{z}})$ and $h(\tilde{\mathbf{z}})$, the later will be noter 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 the stability performances.

The fact that the linearized dynamical system operating on the solution correction exhibits 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:

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

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 learnt using the same rationale but now applied on the data, according to:

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

where $g^{\theta}(\mathbf{z})$ and $h^{\theta}(\mathbf{z})$ refer to the parametric functions related to the measured oil temperature, both them 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 data-sets.

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 depicts the data-driven model integration, proving the stability performances. However, when comparing the residual error (with respect to the experimental data) of the fully data-driven prediction and the one related to the physics-based enriched solution

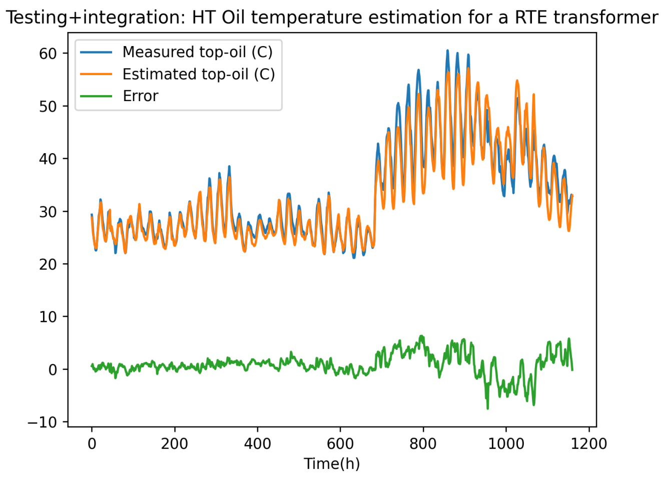


Figure 8. Physics-based simplified model correction from a stabilized linearized ResNet model, here illustrated on the test data-set.

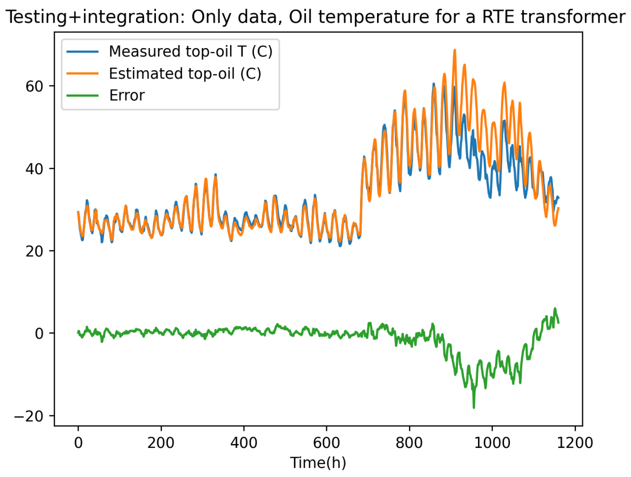


Figure 9. Full data-driven model consisting on a stabilized ResNet model, here illustrated on the test data-set.

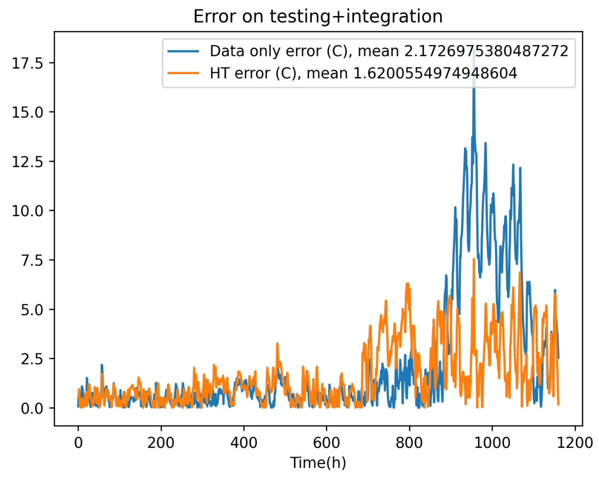


Figure 10. Comparing the errors of the data-only linearized ResNet and the linearized ResNet hybrid model, on the test data-set.

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

Table 3. Comparing different built models: mean error (in C) on the testing set. The nonlinear physics-based model mean error was 3.91C when used alone, over the same testing set, and 3.25C for the linear physics based model.

obtained from the data-driven model of the deviation, both compared in Fig. 10, the hybrid modeling framework performs better, ensuring higher accuracy.

Table 3, compares the mean value 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 a richer nonlinear physics-based model outperform with respect to enriching the linear counterpart of the simplified physics-based model.
- When the considered physics-based models are too far from the reference solution (experimental data) the data-driven model can outperform the hybrid modeling.

To show the effect of the stabilization, a ResNet is 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

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

provides 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.

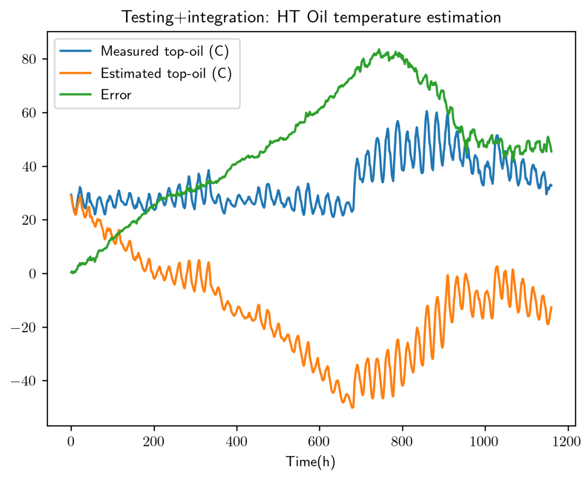


Figure 11. Integration performed from a ResNet learnt without enforcing stability constraints.

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

6. Conclusion

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 predictions 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 the one of the data) must ensure the integration stability. A stabilization has been proposed and its performance proved.

The 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 could be available under request to the authors, after agreement of RTE.

Acknowledgments: 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.

Sample Availability: Samples of the compounds ... are available from the authors.

References

1. C. Argerich, A. Carazo, O. Sainges, E. Petiot, A. Barasinski, M. Piana, L. Ratier, F. Chinesta. Empowering Design Based on Hybrid Twin: Application to Acoustic Resonators. Designs, 4, 44, 2020 https://doi.org/10.3390/designs4040044
2. P.C. Blaud, P. Chevrel, F. Claveau, P. Haurant, A. Mouraud. Resnet and polynet based identification and (mpc) control of dynamical systems: a promising way. IEEE Access, 11, 20657-20672, 2022.
3. D. Borzacchiello, J.V. Aguado, F. Chinesta. Non-intrusive sparse subspace learning for parametrized problems. Archives of Computational Methods in Engineering, 26, 303-326, 2019.

4. L. Breiman. Random Forests. Machine Learning 45, 5-32, 2001.
5. F. Casteran, K. Delage, P. Cassagnau, R. Ibanez, C. Argerich, F. Chinesta. Application of Machine Learning tools for the improvement of reactive extrusion simulation. Macromolecular Materials and Engineering, 2020 https://doi.org/10.1002/mame.202000375
6. F. Chinesta, E. Cueto, E. Abisset-Chavanne, J.L. Duval, F. El Khaldi, Virtual, Digital and Hybrid Twins: A New Paradigm in Data-Based Engineering and Engineered Data, Archives of Computational Methods in Engineering, 27, 105-134, 2020.
7. R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Neural ordinary differential equations. Conference on Neural Information Processing Systems (NeurIPS 2018), Vol. 32, Montreal, Canada, 1-18, 2018.
8. R.T.Q. Chen, B. Amos, M. Nickel. Learning Neural Event Functions for Ordinary Differential Equations, 2020 https://arxiv.org/abs/2011.039
9. N. Cristianini, J. Shawe-Taylor. An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, New York, 2000.
10. S.H. Dar, W. Chen, F. Zheng, S. Gao, K. Hu. An LSTM with Differential Structure and Its Application in Action Recognition, Mathematical Problems in Engineering, 2022, 7316396, 2022.
11. G.P. Distefano. Stability of numerical integration techniques. AIChE Journal, 14(6), 946-955, 1968.
12. L. Enciso-Salas, G. Perez-Zuniga, J. Sotomayor-Moriano. Fault Detection and Isolation for UAVs using Neural Ordinary Differential Equations. IFAC-PapersOnLine, 55(6), 643-648, 2022.
13. R. Ghanem, C. Soize, L. Mehrez, V. Aitharaju. Probabilistic learning and updating of a digital twin for composite material systems, IJNME, 2020 https://doi.org/10.1002/nme.6430
14. C. Ghnatios, P. Gérard, A. Barasinski. An advanced resin reaction modeling using data-driven and digital twin techniques. International Journal of Material Forming, 16, 5, 2023.
15. D. Gonzalez, F. Chinesta, E. Cueto. Learning corrections for hyper-elastic models from data. Frontiers in Materials - section Computational Materials Science, 6, 2019.
16. I. Goodfellow, Y. Bengio, A. Courville. Deep learning. MIT Press, Cambridge, 2016
17. K. He, X. Zhang, S. Ren, J. Sun. Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 770-778, 2016.
18. K. He, X. Zhang, S. Ren, J. Sun. Identity Mappings in Deep Residual Networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision - ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, Vol 9908. Springer, Cham, 2016.
19. S. Hochreiter, J. Schmidhuber. Long Short-Term Memory. Neural Computation 9(8), 1735-1780, 1997.
20. R. Ibanez, E. Abisset-Chavanne, D. Gonzalez, J.L. Duval, E. Cueto, F. Chinesta. Hybrid Constitutive Modeling: Data-driven learning of corrections to plasticity models. International Journal of Material Forming, 12, 717-725, 2019.
21. M.G. Kapteyn, K.E. Willcox. From Physics-Based Models to Predictive Digital Twins via Interpretable Machine Learning, 2020 arXiv:2004.11356v3
22. X. Kestelyn, G. Denis, V. Champaney, N. Hascoet, C. Ghnatios, F. Chinesta. Towards a hybrid twin for infrastructure asset management: Investigation on power transformer asset maintenance management. 7th International Advanced Research Workshop on Transformers (ARWtr), 109-114, 2022.
23. C.W. Kirkwood. Decision Tree primer, 2002, http://creativecommons.org/licenses/by-nc/3.0/
24. B. Moya, A. Badias, I. Alfaro, F. Chinesta, E. Cueto. Digital twins that learn and correct themselves. International Journal for Numerical Methods in Engineering, https://doi.org/10.1002/nme.6535
25. G. Quaranta, M. Ziane, E. Haug, J.L. Duval, F. Chinesta. A minimally-intrusive fully 3D separated plate formulation in computational structural mechanics. Advanced Modelling and Simulation in Engineering Sciences, 6, Article number 11, 2019.
26. A. Reille, V. Champaney, F. Daim, Y. Tourbier, N. Hascoet, D. Gonzalez, E. Cueto, J.L. Duval, F. Chinesta. Learning data-driven reduced elastic and inelastic models of spot-welded patches. Mechanics & Industry, 22, 32, 2021.
27. A. Sancarlos, M. Cameron, A. Abel, E. Cueto, J.L. Duval, F. Chinesta. From ROM of electrochemistry to AI-based battery digital and hybrid twin. Archives of Computational Methods in Engineering, 28, 979-1015, 2021.
28. A. Sancarlos, J.M. Le Peuvedic, J. Groulier, J.L. Duval, E. Cueto, F. Chinesta. Learning stable reduced-order models for hybrid twins. A. Sancarlos, M. Cameron. Data Centric Engineering, 2 , e10, 2021.
29. A. Sancarlos, V. Champaney, E. Cueto, F. Chinesta. Regularized regressions for parametric models based on separated representations. Advanced Modeling and Simulation in Engineering Sciences, 10:4, 2023. https://doi.org/10.1186/s40323-023-00240-4
30. P.J. Schmid. Dynamic mode decomposition of numerical and experimental data. J Fluid Mech., 656, 528, 2010.
31. J. Schmidhuber. Deep learning in neural networks: An overview. Neural Networks. 61, 85-117, 2015.
32. B. Settles, Active Learning Literature Survey. Computer Sciences Technical Report 1648. University of Wisconsin-Madison, 2009.
33. Standard. Loading guide for mineral-oil-immersed power transformers. International electrotechnical commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland, 1, 2018.
34. S. Torregrosa, V. Champaney, A. Ammar, V. Hebert, F. Chinesta. Surrogate Parametric Metamodel based on Optimal Transport. Mathematics and Computers in Simulation, 194, 36-63, 2022.
35. S. Torregrosa, V. Champaney, A. Ammar, V. Herbert, F. Chinesta. Hybrid Twins based on Optimal Transport. Computers and Mathematics with Applications, 127, 12-24, 2022.
36. E.J. Tuegel, A.R. Ingraffea, T.G. Eason, S.M. Spottswood. Reengineering Aircraft Structural Life Prediction Using a Digital Twin. International Journal of Aerospace Engineering, 154798, 2011.

37. G.B. Zhou, J. Wu, C.L. Zhang, Z.H. Zhou. Minimal gated unit for recurrent neural networks. International journal Automation and Computing, 13, 226-234, 2016.