Data-Completion and Model Correction by Means of Evanescent Regularization

Chady Ghnatios, Di Jiang, Yves Tourbier, Alain Cimetière, Francisco Chinesta

▶ **To cite this version:**

Chady Ghnatios, Di Jiang, Yves Tourbier, Alain Cimetière, Francisco Chinesta. Data-Completion and Model Correction by Means of Evanescent Regularization. Applied Sciences, 2023, 13 (17), pp.9616. 10.3390/app13179616 . hal-04694815

HAL Id: hal-04694815
https://hal.science/hal-04694815v1

Submitted on 11 Sep 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

applied sciences

MDPI

Article

Data-Completion and Model Correction by Means of Evanescent Regularization

Chady Ghnatios [1,*], Di Jiang [2], Yves Tourbier [1], Alain Cimetière [3] and Francisco Chinesta [1,4]

[1] PIMM Laboratory, Arts et Métiers Institute of Technology, CNRS, CNAM, HESAM Université, 151 Boulevard de l'Hôpital, 75013 Paris, France; yves.tourbier@ensam.eu (Y.T.); francisco.chinesta@ensam.eu (F.C.)
[2] Renault, 1 Avenue du Golf, 78084 Guyancourt, France; di.jiang@renault.com
[3] Pprime Institute, University of Poitiers, 2 Bd des Frères Lumière, 86360 Chasseneuil-du-Poitou, France; alain.cimetiere@univ-poitiers.fr
[4] CNRS, CREATE Ltd., 1 Create Way, #08-01 CREATE Tower, Singapore 138602, Singapore
* Correspondence: chady.ghnatios@ensam.eu

Abstract: System components are often regarded as part of a whole system, especially when it comes to data-driven modeling. Thus, subsystem modeling is disregarded in general when building a data-driven response, especially since multiple subsystem outputs are never measured in real applications. However, subsystem knowledge and accurate modeling are of utmost importance when aiming to repair, tune or troubleshoot a system. This work proposes a holistic modeling of subsystems in an embedded system setting. A hybrid modeling starting from the physics-based model is proposed in this work, correcting or enhancing the model, and predicting output variables, even when a measurement is never available for some of those variables. The process relies on the variables' history, and employs an adjoint-free neural ordinary differential equation technique, along with evanescent regularization to enhance the convergence on the unmeasurable variables. The updated model converges to the exact measurements, for both the measurable and the unmeasurable variables. Multiple examples are presented using synthetic data, to allow an easy evaluation of the hidden or unmeasurable variables. The relative error offered by the updated model is around 0.001% for the measurable quantities and 0.1% for the unmeasurable ones.

Keywords: model reduction; machine learning; complex systems; partially observable variables; evanescent regularization

check for updates

Citation: Ghnatios, C.; Jiang, D.; Tourbier, Y.; Cimetière, A.; Chinesta, F. Data-Completion and Model Correction by Means of Evanescent Regularization. *Appl. Sci.* **2023**, *13*, 9616. https://doi.org/10.3390/app13179616

Academic Editor: Alberto Corigliano

Received: 26 June 2023
Revised: 9 August 2023
Accepted: 22 August 2023
Published: 25 August 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction and Related Works

Data-driven modeling is gaining popularity in engineering. Multiple novel applications are appearing on a daily basis. A combination of data-driven modeling and the more experienced physics-based modeling, to enhance diagnosis and prognosis, taking the best of both paradigms, is also an active research topic [1]. Multiple data-driven approaches, combined with physical modeling, are used and referred to as hybrid models [2].

Hybrid modeling has enhanced the predictive ability of the available models [3]. However, when a typical physics-based model is not accurate enough, such as when it requires a correction through data-driven approach, the industrial world would be asking to investigate the origin of the error. Such an investigation has been hindered until now by the fact that many intermediate variables used in linking/connecting subsystems are never measured, nor are they measurable. Thus, a component by component data-driven modeling becomes tricky to achieve, unless a methodology for data completion is available.

Multiple works exist on the completion of missing data [4], however, these works focus on data imputation, when data is missing from a data set. Data can be either completely missing at random time samples, missing at random time sample, or not randomly missing at a given time sample. However, in the present work, data related to a given output

are never measured, and probably are unmeasurable. Unmeasurable variables have been recently considered by the scientific community, with several application in the engineering field [5]. In [5], the authors used physical modeling and simulation to complement data-driven approach with the missing unmeasurable data, however, there was no data-driven correction methodology to update these unmeasurable data. In [6], the authors aimed to build the most suitable differential equation of a physical domain, through using a pool of multiple linear and non-linear operators. Discovering physical equations can later be used to predict missing data in regions where the data are not measured in the same domain. This is suitable in the case of continuum mechanics and continuous quantities in a domain, obeying to a certain differential equation in space and time. However, this approach is not suitable when the learning parameters that are not correlated between each other through differentiation of the inputs. In another work [7], the authors removed or hid the unmeasurable quantities through substituting the equations of the unmeasurable parameters into the equations of the measurable ones, without a need of computing the unmeasurable values.

Multiple unmeasurable variables are listed in the literature, and workarounds are being created to overcome limitations [8]. The present work aims at addressing the problem of non-observed quantities that are of interest in a dynamical system. The method relies on the use of Taken's theorem. In fact, Taken's theorem affirms that the state of a system's output can be determined from previous historical observation of the same outputs [9]. Thus, considering the historical evolution of a system, one can predict existing dependencies, even if not all the inputs are observable, using mutual information [10].

This work proposes a holistic framework for model updating and correction through a data-driven methodology, while also updating the unmeasurable variables, even if they are never observed. This approach is novel and has never been addressed until now. The starting point of the proposed methodology is the existence of prior knowledge, a model, which requires enhancement or correction. In Section 2, the available simulation is improved based on measured real data, available only for the measurable quantities of interest. In Section 3, a hybrid modeling of the available simulation, enhancing the results, is built. The methodology is illustrated on an academic use case to clearly illustrate the adopted approach, without any loss of generality.

2. Correction of the Available Model

This section starts by addressing a simple model, to grasp the proposed methodology. Later, details on the approach and results will be given.

2.1. Methods

This section revisits some numerical methods and describes those used in this work.

2.1.1. Dynamical Model

We consider that a model is available, coming from a system modeling software, for example. For the sake of simplicity, we consider:

$$\frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{y} \tag{1}$$

where $\mathbf{A}$ is a no-singular and diagonalizable matrix, with negative eigenvalues to ensure stability. The fact of being diagonalizable and having negative eigenvalues is not a limitation of the proposed methodology, however, they allow us to define sound physical models. The simulated problem is supposed to be less accurate than the exact one, involving the matrix $\mathbf{B}$ such as:

$$\mathbf{B} = \mathbf{A} + \epsilon\mathbf{A} \tag{2}$$

with ϵ small enough. Thus, the modeling and its simulation considers:

$$\frac{d\mathbf{y}}{dt} = \mathbf{B}\mathbf{y} \tag{3}$$

The discreet counterparts of Equations (1) and (3) lead to $\mathbf{y}^e$ and $\mathbf{y}^s$, the reference and simulated solutions, respectively. Thus, using a simple Euler explicit integration scheme in time, one can write at the i-time step:

$$\begin{cases} \mathbf{y}^e_{i+1} = \mathbf{y}^e_i + \Delta t \mathbf{A}\mathbf{y}^e_i \\ \mathbf{y}^s_{i+1} = \mathbf{y}^s_i + \Delta t \mathbf{B}\mathbf{y}^s_i \end{cases} \tag{4}$$

where Δt is the time step.

The model update and correction algorithm takes as input a partial measurements of $\mathbf{y}$, which is named $\mathbf{y}^m$ and aims to predict both $\mathbf{y}^m$ of dimension N_m, as well as $\mathbf{y}^{nm}$ of dimension N_{nm}, the non-measurable part of $\mathbf{y}$, such as $N_m + N_{nm} = N$, the dimension of the quantities of interest $\mathbf{y}$. First, the measurable and non-measurable parts of the model are separated such as:

$$\begin{pmatrix} \mathbf{y}^{s,m}_{i+1} \\ \mathbf{y}^{s,nm}_{i+1} \end{pmatrix} = \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} + \Delta t \begin{pmatrix} B^{m,m} & B^{m,nm} \\ B^{nm,m} & B^{nm,nm} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} \tag{5}$$

Identifying the components of matrix $\mathbf{B}$ in the previous equation seems a tricky issue, because at each time step t_i, the solution involves the hidden variables at the previous time step t_{i-1}. Therefore, one could think that there are too many unknowns to be identified; the components of $\mathbf{B}$ and the hidden estate at each time step, a number that exceeds the number of available equations, the latter being the number of observable states at each time step.

However, by using recurrence, all the solutions at any time step will depend on only the initial one, which consists of some observable and hidden values. Thus, the number of equations (involving the observable state at each time step) should be large enough (when considering a sufficient number of time steps, which it is usually the case) for computing the matrix and the hidden state at only the initial time.

Since this work also aims at identifying a correction for both the measurable and non-measurable outputs, a substitution of $\mathbf{y}^{s,nm}$ through the use of the Schur complement is not the most suitable approach. Instead, a second time iteration will show that measurable and non-measurable quantities are coupled:

$$\begin{gathered} \begin{pmatrix} \mathbf{y}^{s,m}_{i+2} \\ \mathbf{y}^{s,nm}_{i+2} \end{pmatrix} = \begin{pmatrix} \mathbf{y}^{s,m}_{i+1} \\ \mathbf{y}^{s,nm}_{i+1} \end{pmatrix} + \Delta t \begin{pmatrix} B^{m,m} & B^{m,nm} \\ B^{nm,m} & B^{nm,nm} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{s,m}_{i+1} \\ \mathbf{y}^{s,nm}_{i+1} \end{pmatrix} = \\ \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} + \left(2\Delta t \begin{pmatrix} B^{m,m} & B^{m,nm} \\ B^{nm,m} & B^{nm,nm} \end{pmatrix} + \Delta t^2 \begin{pmatrix} B^{m,m} & B^{m,nm} \\ B^{nm,m} & B^{nm,nm} \end{pmatrix}^2 \right) \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} = \\ \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} + 2\Delta t \begin{pmatrix} B^{m,m} & B^{m,nm} \\ B^{nm,m} & B^{nm,nm} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} + \\ \Delta t^2 \begin{pmatrix} B^{m,m}B^{m,m} + B^{m,m}B^{nm,m} & B^{m,m}B^{m,nm} + B^{m,nm}B^{nm,nm} \\ B^{nm,m}B^{m,m} + B^{nm,nm}B^{nm,m} & B^{nm,m}B^{m,nm} + B^{nm,nm}B^{nm,nm} \end{pmatrix} \begin{pmatrix} \mathbf{y}^{s,m}_{i} \\ \mathbf{y}^{s,nm}_{i} \end{pmatrix} \end{gathered} \tag{6}$$

From Equation (6), one can note that after a second iteration of the integration scheme, all the non-measurable quantities $\mathbf{y}^{s,nm}$, as well as the parts of the modeling matrix $\mathbf{B}$ multiplying them, appear in the equation of the measurable quantities $\mathbf{y}^{s,m}$. Thus, an optimization problem can be formulated on the measurable quantities only, and calculating a correction of the model matrix $\mathbf{B}$ for both the measurable and non-measurable quantities. This is only possible by taking into consideration the history of the measurable quantities $\mathbf{y}^{s,m}$.

The optimization problem can be formulated under the following form:

$$\left(\mathbf{B}, \mathbf{y}_0^{s,nm}\right) = \underset{\substack{\mathbf{B} \in \mathbb{R}^{N^2} \\ \mathbf{y}_0^{s,nm} \in \mathbb{R}^{N_{nm}}}}{arg\,\min} \left\{ \sum_{i=1}^{n_t} \left(\sum_{j=1}^{N_m} \left(y_i^{e,j} - y_i^{s,j} \right)^2 \right) \right\}, \tag{7}$$

where $\mathbf{y}_0^{s,nm}$ is the initial condition of the non-measurable variables, and n_t is the number of time steps performed in the integration scheme. Selecting several values of n_t can generate as much equations as required for the problem's solution. However, the loss function defined in Equation (7) is highly non-linear and involves high degrees of the problem's unknowns. To improve the convergence of the optimization problem using a gradient descent algorithm, a regularization is required. Our proposal is making use of the evanescent regularization, revisited in Section 2.1.2.

2.1.2. The Evanescent Regularization

The evanescent regularization is a regularization technique initially developed to address the Cauchy problem, while ensuring convergence of the quantities of interest [11]. The authors in [12] used the Evanescent regularization to address the solution inside a given domain, while only knowing the solution on the boundary of the domain. In that case, the inverse problem had observable or "measurable" degrees of freedom on the boundary, and unmeasurable ones inside the domain. The use of the evanescent regularization is demonstrated to lead to the slow convergence of the unmeasurable quantities of interest, as well as a strong convergence of the measurable ones, towards the exact solution of the problem [13]. The evanescent regularization is also used when the extra boundary conditions are available on a part of the domain, while totally missing on another part of the boundary [11].

Thus, the loss function $\mathcal{L}$ is reformulated in the following form:

$$\begin{aligned} \mathcal{L} = \sum_{i=1}^{n_t} \left(\sum_{j=1}^{N_m} \left(y_i^{e,j} - y_i^{s,j} \right)^2 \right) + \lambda_1 \left(B^{m,m} - \hat{B}^{m,m} \right)^2 + \\ \lambda_2 \left(B^{m,nm} - \hat{B}^{m,nm} \right)^2 + \lambda_3 \left(\mathbf{y}_0^{s,nm} - \hat{\mathbf{y}}_0^{s,nm} \right)^2 \end{aligned} \tag{8}$$

with λ_1, λ_2 and λ_3 the evanescent regularization parameters. The $\hat{}$ quantities are the best estimation of the regularized outputs of the optimization problem, which can be taken as the known values from the previous iteration of the gradient descent algorithm. In fact, the evanescent regularization will improve the convergence of both quantities, with a faster convergence of the measurable quantities of interest. Thus, the regularized optimization problem is reformulated as:

$$\left(\mathbf{B}, \mathbf{y}_0^{s,nm}\right) = \underset{\substack{\mathbf{B} \in \mathbb{R}^{N^2} \\ \mathbf{y}_0^{s,nm} \in \mathbb{R}^{N_{nm}}}}{arg\,\min} \{\mathcal{L}\}, \tag{9}$$

To solve the optimization problem defined in Equation (9), the gradients can always be computed numerically, at the expense of dramatically increasing the computation time of the gradient descent algorithm. It is therefore preferable to use an integration scheme with automatic gradient differentiation through time steps, such as the Neural Ordinary Differential Equations (Neural ODE) methods [14]. This will allow a fast and reliable gradient calculation through automatic differentiation and will accelerate the convergence of the optimization algorithm.

2.1.3. Time Integration and Adjoint-Free Neural ODE

The use of neural Ordinary Differential Equation (ODE), with the adjoint method, involves the integration of another differential equation, backward in time, to compute the gradients. This integration can involve very low time step requirements and increase

dramatically the computation time. Therefore, in this work, this issue is alleviated through proposing an adjoint-free neural ODE. This adjoint-free neural ODE uses a neural network that estimates the derivatives of the quantities of interest $\mathbf{y}$, while computing analytically, through automatic differentiation, their gradients with respect to the neural network training variables. In this case, the matrix $\mathbf{B}$ appearing in Equation (9) is interpreted as the weights of a single layer dense neural network, with no bias.

To illustrate the adjoint-free neural ODE, the derivative of the quantities of interest is written as:

$$\frac{d\mathbf{y}}{dt} = \mathbf{h}(t, \mathbf{y}; p), \tag{10}$$

with $h(t, \mathbf{y}; p)$ is a neural network and p are any input parameters. The feed-forward evaluation is possible though explicit integration using:

$$\begin{gathered} \mathbf{y}_{i+1} = \mathbf{y}_i + \Delta t \mathbf{h}(t_i, p_i, \mathbf{y}_i) \\ \mathbf{y}_{i+2} = \mathbf{y}_{i+1} + \Delta t \mathbf{h}(t_{i+1}, p_{i+1}, \mathbf{y}_{i+1}) = \\ = \mathbf{y}_i + \Delta t \mathbf{h}(t_i, p_i, \mathbf{y}_i) + \Delta t \mathbf{h}(t_{i+1}, p_{i+1}, \mathbf{y}_i + \Delta t \mathbf{h}(t_i, p_i, \mathbf{y}_i)) \\ \vdots \end{gathered} \tag{11}$$

The derivation of $\mathbf{y}_{i+n_t}$ with respect to the trainable parameters of $\mathbf{h}$, $\boldsymbol{\theta}$ for instance, can be performed through a straightforward manner by the means of the chain rule, and using the property $(f \circ g(\boldsymbol{\theta}))' = f'(g(\boldsymbol{\theta})) \cdot g'(\boldsymbol{\theta})$. Thus, if $\boldsymbol{\theta}$ is the vector of the neural network trainable variables, one can write $(\mathbf{h} \circ \mathbf{h}(\boldsymbol{\theta}))' = \mathbf{h}'(\mathbf{h}(\boldsymbol{\theta})) \cdot \mathbf{h}'(\boldsymbol{\theta})$. The gradients of $\mathbf{y}_{i+n_t}$ are therefore easily computed, using the chain rule, while $\mathbf{h}'(\boldsymbol{\theta})$ is computed through automatic differentiation.

The neural ODE can therefore compute several values for $\mathbf{y}_{i+n_t}$, to be used in the loss function and the optimization problem defined in Equation (9), which can be re-written in its general form:

$$(\boldsymbol{\theta}, \mathbf{y}_0^{s,nm}) = \underset{\substack{\boldsymbol{\theta} \in \mathbb{R}^{N^2} \\ \mathbf{y}_0^{s,nm} \in \mathbb{R}^{N_{nm}}}}{arg\min} \{\mathcal{L}\}, \tag{12}$$

$\boldsymbol{\theta}$ being the model hyperparameters to optimize, which are the values of the matrix $\mathbf{B}$ of dimensionality N^2 in our selected example ($\boldsymbol{\theta} \equiv \mathbf{B}$ in our example). The final algorithm for updating the simulation model and training the model weights is illustrated in Algorithm 1. The ADAM stochastic optimization algorithm [15] is used for the gradient descent in our examples.

Algorithm 1 Algorithm used for optimizing the simulation model and detecting the initial condition of the unmeasurable parameters, using the adjoint-free neural ODE and the evanescent regularization

Initialize $\boldsymbol{\theta}$ as $\hat{\boldsymbol{\theta}}$, $\mathbf{y}_0^{s,nm}$ as $\hat{\mathbf{y}}_0^{s,nm}$
while $\mathcal{L} \geq \epsilon$ **do**
  while $i < n_t$ **do**
    Compute $\mathbf{y}_i$ using adjoint-free Neural ODE integration, Equation (11)
  end while
  Compute $\mathcal{L}$
  Evaluate $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}}$ and $\frac{\partial \mathcal{L}}{\partial \mathbf{y}_0^{s,nm}}$
  $\begin{pmatrix} \boldsymbol{\theta} \\ \mathbf{y}_0^{s,nm} \end{pmatrix} \leftarrow ADAM\left(\begin{pmatrix} \boldsymbol{\theta} \\ \mathbf{y}_0^{s,nm} \end{pmatrix}; \begin{pmatrix} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{y}_0^{s,nm}} \end{pmatrix} \right)$
end while
return $(\boldsymbol{\theta}; \mathbf{y}_0^{s,nm})$

2.2. Results: Model Correction

In this section, for the sake of simplicity, a simple model with $N = 5$ is considered (given in the Appendix A), and a single dense neural network layer connecting the inputs and the outputs. No input parameters p are used in this case, the time is the only model input. The trainable parameters $\boldsymbol{\theta}$ and $\mathbf{y}_0^{s,nm}$ are initiated with the values coming from the available simulation. The initial curves are shown in Figure 1, for the exact model (in solid lines) and the simulated one (in dashed lines). In this section, only the last output is assumed to be unmeasurable.

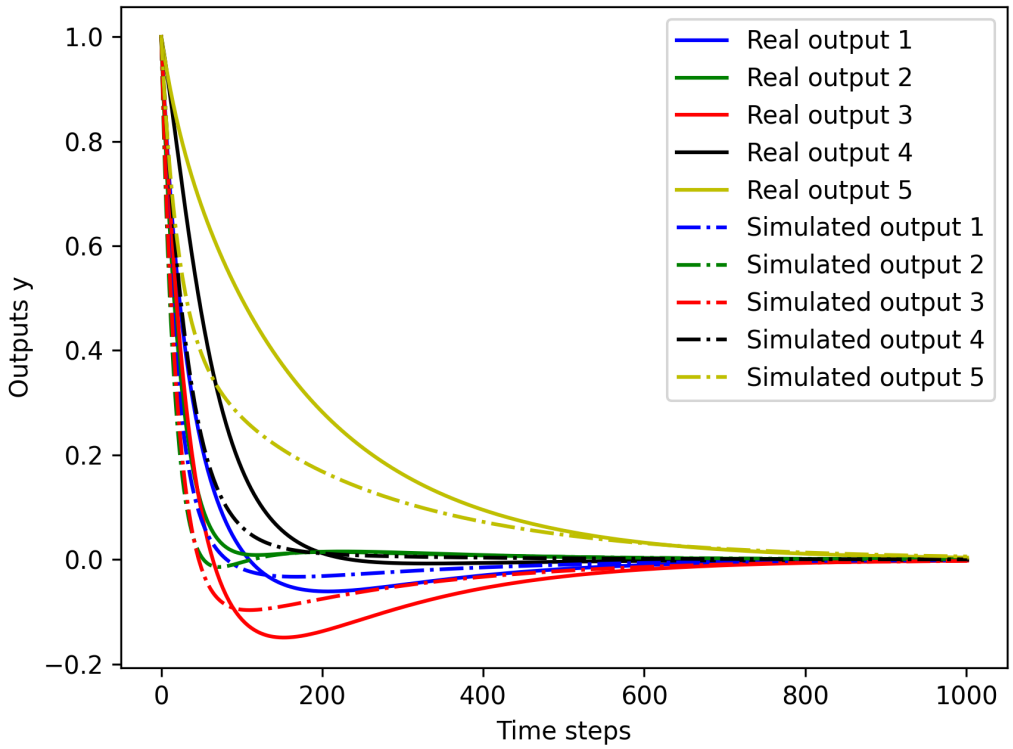


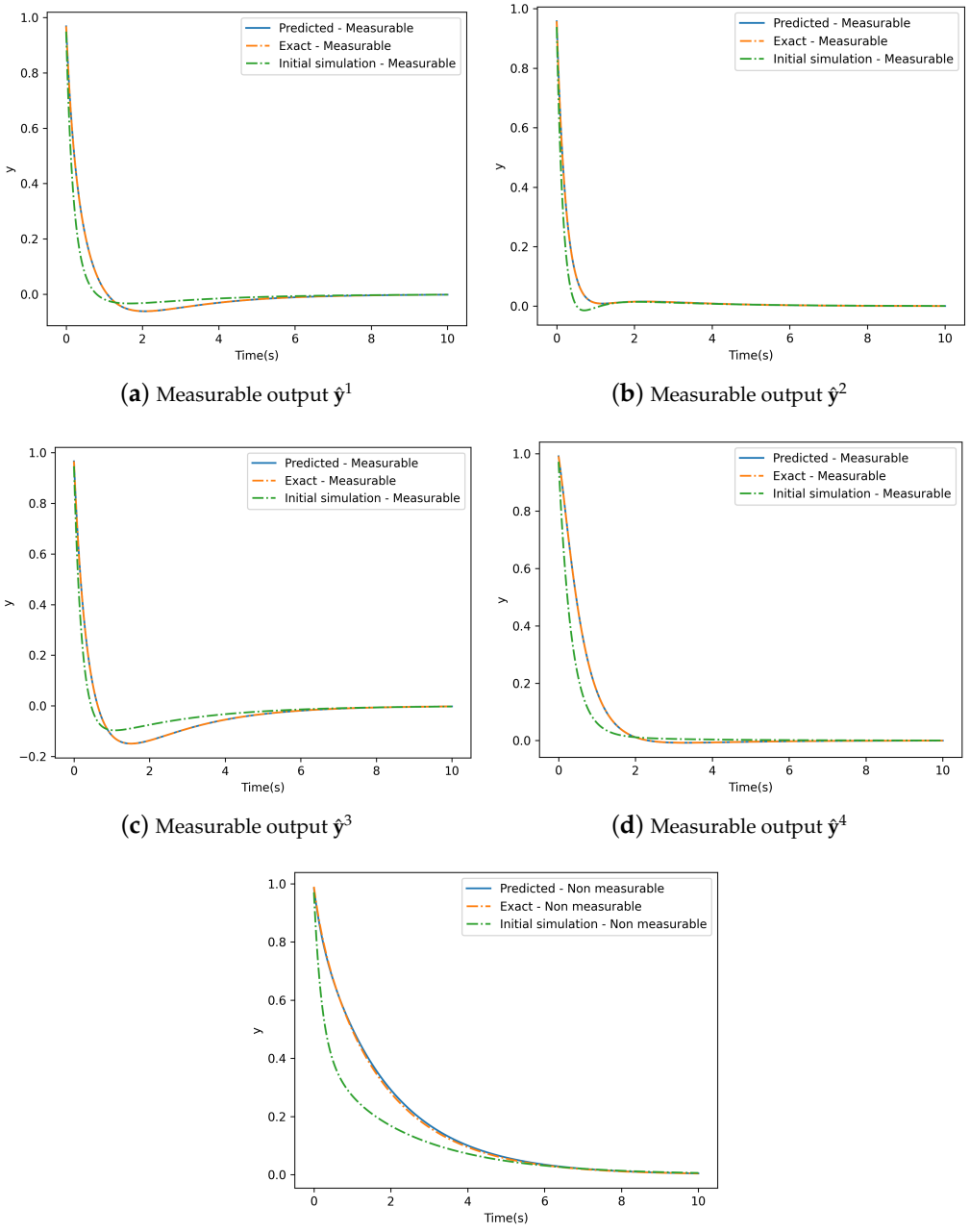
Figure 1. Exact solution and simulation of the considered model with $N = 5$.

The initial condition for the unmeasurable variable is deliberately set to a wrong initial value $\mathbf{y}_0^{h,nm} = 0.8$, while the exact initial conditions are considered for the measurable ones, $\mathbf{y}_0^{h,m} = 1$. The training is performed on a time domain $t \in [0, 10]s$ with $n_t = 1000$ time steps and $\Delta t = 0.01s$. The training with an adjoint-free neural ODE is performed on a portable PC within half an hour. The results after the training are illustrated in Figure 2 for the measurable and non-measurable variables.The used values of the regularization coefficients in this section are $\lambda_1 = 10^{-5}$, $\lambda_2 = 10^{-5}$, and $\lambda_3 = 1$.

The results illustrated in Figure 2 highlight the ability of the network to correct the existing simulation even though neither the initial value nor the output of the last quantity of interest are available. The mean relative errors are:

- Relative error on $\mathbf{y}_0^{s,nm}$, the initial condition of the non-measurable quantity: 0.49%
- Relative error on the measurable quantities $\mathbf{y}^m$: 0.01%
- Relative error on the unmeasurable quantities $\mathbf{y}^{nm}$: 0.43%

The predictive ability is highlighted, and shows that the proposed approach is able to correct an existing simulation through time variation of only several quantities of interest.



(**a**) Measurable output $\hat{\mathbf{y}}^1$

(**b**) Measurable output $\hat{\mathbf{y}}^2$

(**c**) Measurable output $\hat{\mathbf{y}}^3$

(**d**) Measurable output $\hat{\mathbf{y}}^4$

(**e**) Non measurable output $\hat{\mathbf{y}}^5$

Figure 2. Final output of the trained neural network for both the measurable and unmeasurable outputs.

3. A Hybrid Modeling Approach

3.1. A Single Unmeasurable Quantity

When it comes to hybrid modeling, the approach is similar to the previously proposed one; using the same methods described in Section 2.1. However, this approach assumes the existence of another model **C** existing in parallel to the simulation, and integrating the results simultaneously. For instance, in the linear case for example, the hybrid model solution $\mathbf{y}^h_{i+1}$ at any time steps can be written as:

$$\begin{cases} \mathbf{y}^e_{i+1} = \mathbf{y}^e_i + \Delta t\ \mathbf{A}\ \mathbf{y}^e_i \\ \mathbf{y}^s_{i+1} = \mathbf{y}^s_i + \Delta t\ \mathbf{B}\ \mathbf{y}^s_i \\ \mathbf{y}^h_{i+1} = \mathbf{y}^h_i + \Delta t\ \mathbf{B}\ \mathbf{y}^h_i + \Delta t\ \mathbf{C}\ \mathbf{y}^h_i \end{cases} \tag{13}$$

In the non-linear situation, the hybrid model would be written as:

$$\frac{d\mathbf{y}^h}{dt} = \mathbf{h}(t, p, \mathbf{y}^h) + \mathbf{g}(t, p, \mathbf{y}^h) \tag{14}$$

where $\mathbf{g}$ is the correction term and $\mathbf{h}$ is the initial simulation model assumed non-trainable in this case. The loss function is changed in the hybrid modeling to include the L_2 norm regularization with respect to the hyperparameters of the trainable model $\mathbf{g}(t, p, \mathbf{y})$, because it is expected that the correction is not large with respect to $\mathbf{B}$. For instance, the new loss function $\mathcal{J}$ is written in the linear case by:

$$\begin{aligned}\mathcal{J} = \sum_{i=1}^{n_t}\left(\sum_{j=1}^{N_m}\left(y_i^{e,j} - y_i^{h,j}\right)^2\right) + \lambda_1\left(C^{m,m} - \hat{C}^{m,m}\right)^2 + \lambda_2\left(C^{m,nm} - \hat{C}^{m,nm}\right)^2 + \\ + \lambda_3\left(\mathbf{y}_0^{h,nm} - \hat{\mathbf{y}}_0^{h,nm}\right)^2 + \lambda_4\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\hat{C}_{ij}\right)^2 + \lambda_5\left(\|C^{m,nm}\| - \|\hat{C}^{m,m}\|\right)^2\end{aligned} \tag{15}$$

where $\hat{C}_{ij}$ includes all of the components of $\mathbf{C}$, and enforce the sparsity in the correction, even if a stronger enforcement could be applied by using the L_1 norm. The new loss function can be generalized for the non-linear case using:

$$\begin{aligned}\mathcal{J} = \sum_{i=1}^{n_t}\left(\sum_{j=1}^{N_m}\left(y_i^{e,j} - y_i^{h,j}\right)^2\right) + \lambda_1\left(\boldsymbol{\theta}^{m,m} - \hat{\boldsymbol{\theta}}^{m,m}\right)^2 + \lambda_2\left(\boldsymbol{\theta}^{m,nm} - \hat{\boldsymbol{\theta}}^{m,nm}\right)^2 + \\ + \lambda_3\left(\mathbf{y}_0^{h,nm} - \hat{\mathbf{y}}_0^{h,nm}\right)^2 + \lambda_4\sum_{k=1}^{N_\theta}\left(\hat{\boldsymbol{\theta}}_k\right)^2 + \lambda_5\left(\|\boldsymbol{\theta}^{m,nm}\| - \|\hat{\boldsymbol{\theta}}^{m,m}\|\right)^2,\end{aligned} \tag{16}$$

with $\boldsymbol{\theta}$ being the trainable parameters of the non-linear model $\mathbf{g}$, having a total number of N_θ trainable parameters.

The exact error matrix $\mathbf{C}$ used in this example is shown in Appendix A. The last term multiplying λ_5 couples the amplitudes of the equations computing the measurable part and the non-measurable part, to reduce the possibility of having a scale bias. The optimization problem is now written as:

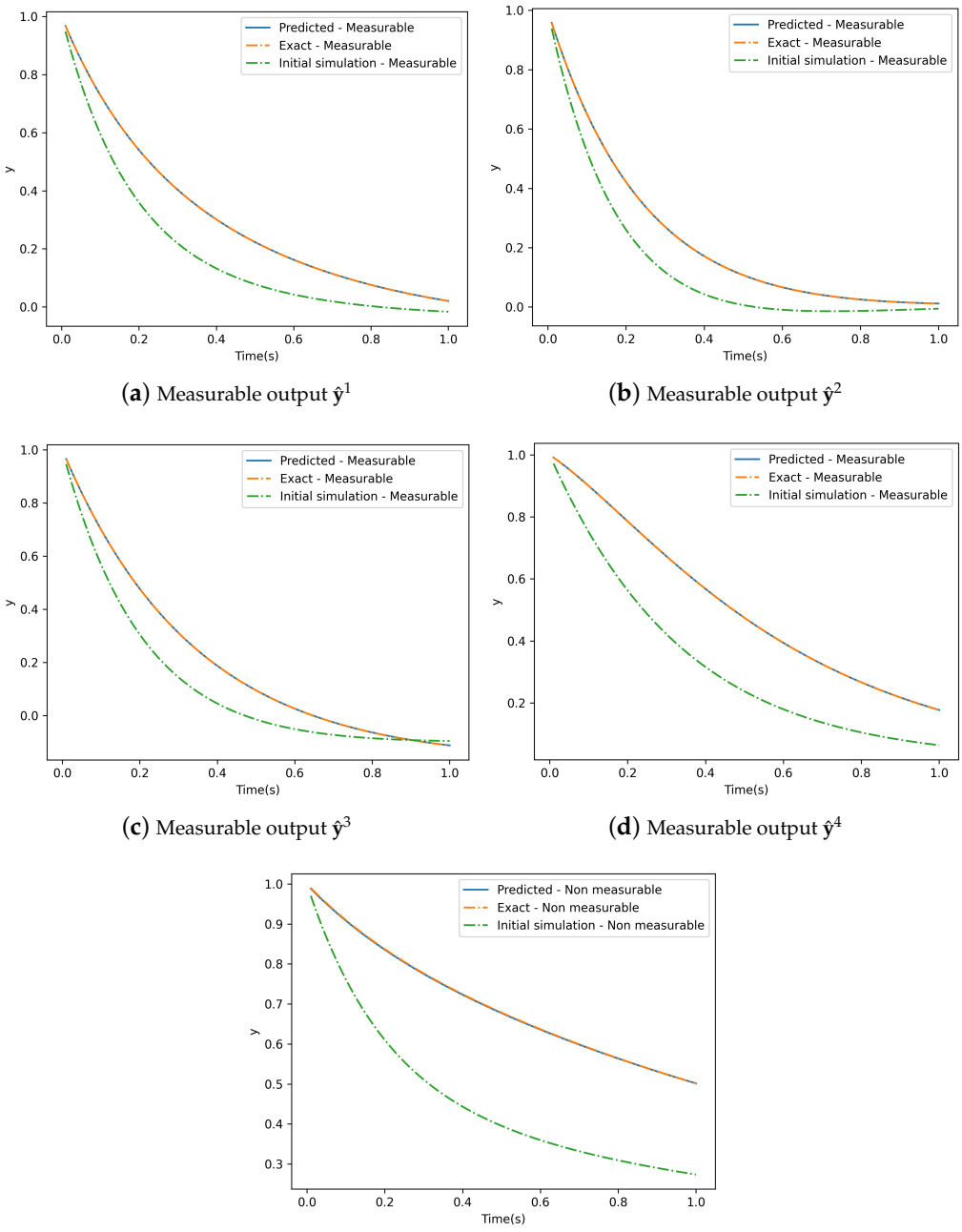
$$\left(\boldsymbol{\theta}, \mathbf{y}_0^{h,nm}\right) = \underset{\substack{\boldsymbol{\theta} \in \mathbb{R}^{N^2} \\ \mathbf{y}_0^{h,nm} \in \mathbb{R}^{N_{nm}}}}{arg\,\min}\ \{\mathcal{J}\}, \tag{17}$$

where $\boldsymbol{\theta}$ are the hyperparameters to optimize. The training is performed on a standard laptop within few minutes, even when starting from a very wrong initial condition $\mathbf{y}_0^{h,nm}$. The results are illustrated in Figure 3 for both the measurable and the unmeasurable quantities of interest, when performing the training over 100 time steps only, and an initial guess $\mathbf{y}_0^{h,nm} = 0.5$, while 1 was the correct initial condition in this example. The used values of the regularization coefficients in this example are $\lambda_1 = 10^{-3}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-3}$, $\lambda_4 = 10^{-4}$, and $\lambda_5 = 10^{-2}$.

The mean relative errors in the updated model are found to be:

- Relative error on $\mathbf{y}_0^{h,nm}$, the initial condition of the non-measurable quantity: 0.15%
- Relative error on the measurable quantities $\mathbf{y}^m$: 0.0093%
- Relative error on the unmeasurable quantities $\mathbf{y}^{nm}$: 0.104%

We can conclude that the hybrid model is able to outperform the simulation correction, with a lower number of time steps. In fact, increasing the number of the available time steps increases the information that is used by the machine learning algorithm to correct the simulation. On the other hand, the increase in the number of time steps integration requires an increase in the computation time. The approach illustrated in Section 2.2 did correct the simulation, but even after 1000 time steps available, the results were less accurate than those of the hybrid model after 100 available time steps only.



(a) Measurable output $\hat{\mathbf{y}}^1$

(b) Measurable output $\hat{\mathbf{y}}^2$

(c) Measurable output $\hat{\mathbf{y}}^3$

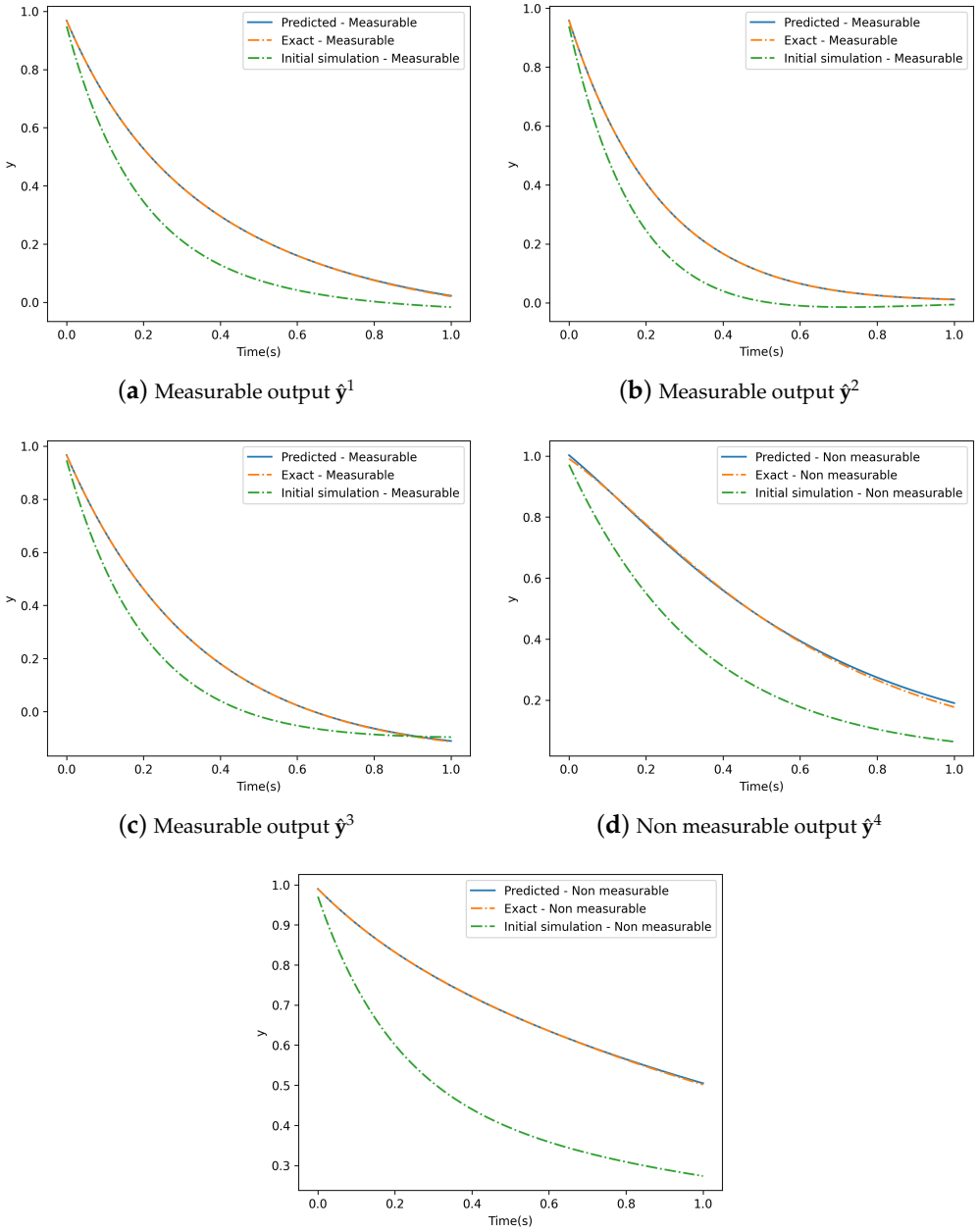
(d) Measurable output $\hat{\mathbf{y}}^4$

(e) Non measurable output $\hat{\mathbf{y}}^5$

Figure 3. Final output of the trained hybrid modeling using a neural network for the correction models. The shown results are for both the measurable and unmeasurable outputs.

3.2. A More Complex Case

This section addresses a more complex example, where the last two variables of interest in our model are unmeasurable. This leads to a fraction of 40% of our model being hidden or unmeasurable. Using the same loss function given in Equation (17), with the regularization coefficients $\lambda_1 = 10^{-3}$, $\lambda_2 = 10^{-3}$, $\lambda_3 = 10^{-3}$, $\lambda_4 = 10^{-4}$ and $\lambda_5 = 10^{-2}$, the results are presented in Figure 4 proving again an excellent accuracy. The initialization was set to $[0.5; 1.5]$ for the two unmeasurable quantities instead of 1, a 50% relative error with respect to the correct initial conditions.



(**a**) Measurable output $\hat{\mathbf{y}}^1$

(**b**) Measurable output $\hat{\mathbf{y}}^2$

(**c**) Measurable output $\hat{\mathbf{y}}^3$

(**d**) Non measurable output $\hat{\mathbf{y}}^4$

(**e**) Non measurable output $\hat{\mathbf{y}}^5$

Figure 4. Final output of the trained hybrid modeling using a neural network for the correction models with two unmeasurable quantities $\hat{\mathbf{y}}^4$ and $\hat{\mathbf{y}}^5$. The shown results are for both the measurable and unmeasurable outputs.

Figure 4 showcases the ability of the proposed methodology to correct the model with nearly half of the variables set unmeasurable. The final mean relative errors for the updated model are found to be:

- Mean relative error on $\mathbf{y}_0^{h,nm}$, the initial condition of the non-measurable quantity: 0.67%
- Mean relative error on the measurable quantities $\mathbf{y}^m$: 0.051%
- Mean relative error on the unmeasurable quantities $\mathbf{y}^{nm}$: 0.3%

4. Conclusions

In this work, a systematic approach to update or correct any simulation by using an intrusive correction, which handles simulation component correction, is proposed. This approach conserves the simulation structure and is thus fully explainable. The evanescent regularization, as well as an adjoint-free neural ODE, are leveraged to achieve the correction. The work also shows that hybrid modeling with L_2 norm regularization of the cost function outperforms a direct correction of the model, even when using a lower number of available simulation and experimental time steps.

Author Contributions: Conceptualization, C.G., F.C. and Y.T.; methodology, C.G. and A.C.; software, C.G. and D.J.; validation, D.J.; formal analysis, C.G., F.C. and D.J.; writing—original draft preparation, C.G.; writing—review and editing, Y.T. and F.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data generator is included.

Acknowledgments: Authors knowledge the ESI, SKF and RTE Chairs at ENSAM, the support of Renault Group and the French ANRT through its CIFRE programme. This research is also part of the programme DesCartes and is supported by the National Research Foundation, Prime Minister Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme.

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

Appendix A

The data involved in the numerical examples addressed in this papers where generated by using the matrices **A** and **B** given by:

$$\mathbf{A} = \begin{bmatrix} -4.35611932 & -1.22951563 & 1.83881934 & 0.54265555 & -0.01757481 \\ -1.15562685 & -3.69112028 & 0.49813577 & -0.0742603 & 0.20472346 \\ 1.79339231 & 0.37881996 & -4.76496262 & 0.99545275 & -1.8469943 \\ 0.65640466 & -0.03439998 & 1.15462889 & -3.33237733 & 0.67350006 \\ 0.32680216 & 0.47521164 & -1.4571903 & 0.90412792 & -1.2567266 \end{bmatrix} \quad \text{(A1)}$$

and

$$\mathbf{B} = \begin{bmatrix} -4.84013258 & -1.63964011 & 1.30937905 & 0.1723914 & -0.1572111 \\ -1.63964011 & -4.10124475 & -0.03130452 & -0.44452445 & 0.06508717 \\ 1.30937905 & -0.03130452 & -5.29440291 & 0.6251886 & -1.98663059 \\ 0.1723914 & -0.44452445 & 0.6251886 & -3.70264148 & 0.53386377 \\ -0.1572111 & 0.06508717 & -1.98663059 & 0.53386377 & -1.39636289 \end{bmatrix} \quad \text{(A2)}$$

The considered time integration is the simplest Euler's integration schema, with a time step $\Delta t = 0.01$ s. The induced error **C** is the difference between **A** and **B**:

$$\mathbf{C} = \begin{bmatrix} 0.4840 & 0.4101 & 0.5294 & 0.3703 & 0.1396 \\ 0.4840 & 0.4101 & 0.5294 & 0.3703 & 0.1396 \\ 0.4840 & 0.4101 & 0.5294 & 0.3703 & 0.1396 \\ 0.4840 & 0.4101 & 0.5294 & 0.3703 & 0.1396 \\ 0.4840 & 0.4101 & 0.5294 & 0.3703 & 0.1396 \end{bmatrix} \quad \text{(A3)}$$

References

1. Chinesta, F.; Cueto, E. Empowering engineering with data, machine learning and artificial intelligence: a short introductive review. *Adv. Model. Simul. Eng. Sci.* **2022**, *9*, 21. [CrossRef]
2. Ghnatios, C. A hybrid modeling combining the proper generalized decomposition approach to data-driven model learners, with application to nonlinear biphasic materials. *Comptes Rendus Mécanique* **2021**, *349*, 259–273. [CrossRef]
3. 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. [CrossRef]
4. Sun, Y.; Li, J.; Xu, Y.; Zhang, T.; Wang, X. Deep learning versus conventional methods for missing data imputation: A review and comparative study. *Expert Syst. Appl.* **2023**, *227*, 120201. [CrossRef]
5. Shourangiz-Haghighi, A.; Tavakolpour-Saleh, A. A neural network-based scheme for predicting critical unmeasurable parameters of a free piston Stirling oscillator. *Energy Convers. Manag.* **2019**, *196*, 623–639. [CrossRef]
6. Chen, Z.; Liu, Y.; Sun, H. Physics-informed learning of governing equations from scarce data. *Nat. Commun.* **2021**, *12*, 6136. [CrossRef] [PubMed]
7. Ding, B.; Pan, H. Output feedback robust model predictive control with unmeasurable model parameters and bounded disturbance. *Chin. J. Chem. Eng.* **2016**, *24*, 1431–1441. [CrossRef]
8. Zhang, Y.; Wang, S.; Heinrich, M.K.; Wang, X.; Dorigo, M. 3D hybrid formation control of an underwater robot swarm: Switching topologies, unmeasurable velocities, and system constraints. *ISA Trans.* **2023**, *136*, 345–360. [CrossRef] [PubMed]
9. Taken, F. Detecting Strange Attractors in Turbulence. In *Dynamical Systems and Turbulence*; Lecture Notes in Mathematics; Springer: Berlin/Heidelberg, Germany, 1981; Volume 898, pp. 366–381.
10. Fraser, A.; Swinney, H. Independent coordinates for strange attractors from mutual information. *Phys. Rev. A* **1986**, *33*, 1134–1140. [CrossRef] [PubMed]
11. Delvare, F.; Cimetière, A. The Evanescent Regularieation Method for Elliptic Cauchy Problems. In *Inverse Problems and Computational Mechanics*; Editura Academiei: Bucharest, Romania, 2011; Volume 1, pp. 101–124.
12. Ghnatios, C.; Xu, G.; Leygue, A.; Visonneau, M.; Chinesta, F.; Cimetière, A. On the space separated representation when addressing the solution of PDE in complex domains. *Discret. Contin. Dyn. Syst. S* **2016**, *9*, 475–500. [CrossRef]
13. Caillé, L. Méthodes de Régularisation Évanescente pour la Complétion de Données. Ph.D. Thesis, Université de Caen Normandie, Caen, France, 2018.
14. 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, 3–8 December 2018; Volume 32, pp. 1–18.
15. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.