Chaos, Solitons and Fractals 186 (2024) 115215

Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

journal homepage: www.elsevier.com/locate/chaos

Nonlinear discrete-time observers with Physics-Informed Neural Networks

Hector Vargas Alvarez [a], Gianluca Fabiani [a,c], Nikolaos Kazantzis [b,*], Ioannis G. Kevrekidis [c,d,e], Constantinos Siettos [f,*]

[a] Scuola Superiore Meridionale, Naples, Italy
[b] Department of Chemical Engineering, Worcester Polytechnic Institute, Worcester, USA
[c] Department of Chemical & Biomolecular Engineering, Johns Hopkins University, Baltimore, USA
[d] Department of Applied Mathematics and Statistics, Johns Hopkins University, Baltimore, USA
[e] Medical School, Department of Urology, Johns Hopkins University, Baltimore, USA
[f] Dipartimento di Matematica e Applicazioni "Renato Caccioppoli", Università degli Studi di Napoli "Federico II", Naples, Italy

ARTICLE INFO

Keywords:
Artificial intelligence
Physics-Informed neural networks
Nonlinear discrete time observers
Approximation of nonlinear operators
Uncertainty quantification

ABSTRACT

We use physics-informed neural networks (PINNs) to numerically solve the discrete-time nonlinear observer-based state estimation problem. Integrated within a single-step exact observer linearization framework, the proposed PINN approach aims at learning a nonlinear state transformation operator by solving a system of functional equations. The performance of the proposed approach is assessed via two illustrative case studies, for which the observer linearizing transformation operator can be derived analytically. We also perform an uncertainty quantification analysis for the proposed scheme. The performance and numerical approximation accuracy of the proposed scheme is compared with conventional power-series numerical implementation.

1. Introduction

In modern feedback control systems theory and practice, reliable access to the dynamically evolving system states is needed at both the implementation stage of advanced control algorithms and for process/system condition and performance monitoring purposes [1–6]. Traditionally, an explicit use of an available dynamic model complemented by sensor measurements (involving measurable physical and chemical variables) represented a first option to respond to the above need. However, in practice, key critical state variables are often not available for direct on-line measurement due to inherent physical as well as technical/economic limitations associated with the current state of sensor technology [1–3]. In light of the above remarks, a better, scientifically sound and practically insightful option is the design of a state estimator (an observer).

An observer is an appropriately structured dynamical system itself that utilizes all information provided by a system model as well as available sensor measurements to accurately reconstruct the dynamic profiles of all other unmeasurable state variables [1,2,4,5]. Such a state observer is endowed with the requisite design degrees of freedom to arbitrarily assign the rate of convergence of the state estimate it generates to the actual state (equivalently, the dynamic modes of the induced estimation error dynamics).

Since digital and Artificial Intelligence (AI) technology are increasingly integrated into advanced modern control and system performance monitoring strategies, the design of state estimators/observers with mathematical representations realized in the discrete-time domain (known also as "software" or "soft" sensors) appears as an amply motivated line of contemporary research activity. Even though the well-known Kalman filter and its deterministic analogue, the Luenberger observer, represent comprehensive solutions to the linear discrete-time observer design problem [2,4], the nonlinear discrete-time one is considerably more challenging and has attracted appreciable attention in the pertinent literature. For nonlinear systems, the conventional approaches have focused on the linearization of the dynamic equations around: (i) either a reference equilibrium point and the subsequent implementation of a standard linear Luenberger observer, or (ii) a reference trajectory followed by a standard Kalman filter design method ("extended Kalman filter") [1,2,4,5]. However, both traditional approaches exhibit local validity within a small region in state space around the steady state, and could therefore lead to unacceptable observer performance since dominant system nonlinearities are conveniently ignored. Within the geometric system-theoretic framework, systematic discrete-time nonlinear observer design methodologies have been introduced [7–9] in an effort to rigorously develop discrete-time analogues of the pioneering work by Krener and Isidori (1983) [10],

* Corresponding authors.
E-mail addresses: nikolas@wpi.edu (N. Kazantzis), constantinos.siettos@unina.it (C. Siettos).

https://doi.org/10.1016/j.chaos.2024.115215
Received 21 February 2024; Received in revised form 19 June 2024; Accepted 26 June 2024
Available online 9 July 2024

previously developed in the continuous-time nonlinear observer design problem. In all the aforementioned approaches, the authors seek *for a nonlinear state transformation, an operator, to linearize the system up to an additional output injection term*, followed by a second step whereby a linear Luenberger observer design method is applied to the transformed system (known as exact observer linearization methods). However, such nonlinear observer design methods, while theoretically rigorous and insightful, rely on a set of rather restrictive conditions that can hardly be met by physical or engineering systems [7–9]. Other interesting approaches, presented in [11–14], propose nonlinear discrete-time observer design methods based on either restrictive global Lipshitz conditions, on past values of certain output variables and/or become applicable to special classes of nonlinear systems with mild nonlinearities. Within the exact observer linearization framework, Kazantzis and Kravaris (2001) [15] introduced a new nonlinear discrete-time observer design method through a single-step problem reformulation that no longer requires the redundant linearization of the system's output map, thus effectively overcoming the restrictive conditions associated with the aforementioned approaches. In particular, it was shown that for linearly observable, real analytic nonlinear discrete-time systems whose spectrum of the linear part lies wholly in the Poincaré domain, the single-step nonlinear discrete-time observer design problem admits a solution that is based on a fairly general set of assumptions. In subsequent work by Xiao et al. (2003) [16] the above Poincaré requirement on the spectrum of the system's linear part was relaxed and replaced by an even milder assumption of the Siegel-type. Finally, recent important contributions further extended this approach to cases where relaxed regularity assumptions imposed on the system dynamic equations guarantee the existence of such an observer [17,18].

Recent theoretical and technological advances have reignited the systems and control community's interest in the development of new machine learning-based schemes by revisiting techniques developed back in the 90s [19–33] as well as introducing new ones [6,34–43]. Furthermore, rigorous research studies aiming at integrating various neural network architectures into the continuous-time single-step exact observer linearization framework introduced in Kazantzis and Kravaris (1998) [44] were presented in [45–49]. Recently, in another interesting research study, an unsupervised deep learning neural network method has been introduced in Peralez et al. (2021) [50], and integrated into the discrete-time single-step exact observer linearization framework introduced in Kazantzis and Kravaris (2001) [15]. It should be pointed out that deep learning neural network structures employed for learning nonlinear maps have been evaluated in the pertinent literature as well [51]. Within a conceptually similar nonlinear discrete-time observer design context, in Peralez et al. (2022) [52], a new method is proposed that leads to an enhancement of the numerical approximation of the map and its extension in the transient phase with an ANN architecture.

Motivated by previous work on the feedback linearization of discrete time maps using Physics-Informed Neural Networks (PINNs) in Vargas Alvarez et al. (2023) [39], PINNs are used in the present research study to solve the nonlinear observer design problem in the discrete-time domain within the single-step exact observer linearization framework originally introduced by Kazantzis and Kravaris (2001) [15]. In particular, we use PINNs to learn the state transformation operator that linearizes the observer dynamic equations up to an output injection map. The PINN-based discovery of the above operator, that in the original single-step problem formulation has to satisfy a system of first-order functional equations, enables the design of a discrete-time observer with linearizable error dynamics and assignable convergence rates under a set of rather mild assumptions. In particular, for the training process, a step/greedy-wise training method is employed in Vargas Alvarez et al. (2023) [39] to enhance the PINN scheme's ability to numerically approximate nonlinear maps in the presence of steep gradients.

The paper is organized as follows: in Section 2, we present a brief overview and the necessary mathematical preliminaries associated with the single-step exact observer linearization framework in the discrete-time domain, followed by a description of the structure of the proposed PINN scheme. In Section 3, the two benchmark case studies are introduced. The numerical results derived from the proposed PINN-methods are presented in Section 4, along with a discussion on a comparative performance assessment against the classical power series expansion method. Finally, concluding remarks are offered in Section 5.

2. Preliminaries and methodological framework

We consider nonlinear discrete-time autonomous dynamical systems admitting the following state-space representation:

$$x(t+1) = \Phi(x(t))$$
$$y(t) = h(x(t)), \tag{1}$$

where $t = 0, 1, 2, \ldots$ is the discrete-time index, $x(t) \in \mathbb{R}^n$ is the state variables vector, and, here for the presentation of the methodology but not limited to, $y(t) \in \mathbb{R}$ represents the system's measured output variable. It is also assumed that $\Phi(x)$ is a real analytic vector function defined on $\mathbb{R}^n$ and $h(x)$ is a real analytic scalar function on $\mathbb{R}^n$. Furthermore, without loss of generality, let the origin $x = 0$ be an equilibrium point of (1): $\Phi(0) = 0$ with $h(0) = 0$. Please notice that in the case of a non-zero equilibrium point x_0: $\Phi(x_0) = x_0$, a simple state transformation: $\hat{x} = x - x_0$ leads to a transformed system of discrete-time dynamic equations: $\hat{x}(t+1) = \Phi(\hat{x}(t) + x_0) - x_0$, with an equilibrium point located at the origin: $\hat{x} = 0$ in the new coordinates. The following assumptions are made:

Assumption 2.1. The Jacobian matrix F of $\Phi(x)$ evaluated at $x = 0$: $F = (\frac{\partial \Phi}{\partial x})(0)$ has eigenvalues k_i, $i \in \mathbb{N}$ that all lie in the Poincaré domain.

Assumption 2.2. Denoting by H the $1 \times n$ matrix: $H = \frac{\partial h}{\partial x}(0)$, it is assumed that the following $n \times n$ matrix O:

$$O = \begin{bmatrix} H \\ HF \\ \vdots \\ HF^{n-1} \end{bmatrix} \tag{2}$$

has rank n. This assumption states that system in Eq. (1) is locally observable around the origin $x = 0$ [2,53].

A nonlinear discrete-time state observer driven by the available measurements of the output variable $y(t)$ and being capable of reconstructing the vector of state variables of system in Eq. (1) conforms to the structure delineated in the definition below [1,15]:

Definition 2.1. A dynamical system

$$z(t+1) = \Psi(z(t), y(t)) \tag{3}$$

with $z \in \mathbb{R}^n$; $y \in \mathbb{R}$; $\Psi : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, is called a *full-order observer* for Eq. (1), if there exists a locally invertible (around the origin) operator $T(x)$ with $T : \mathbb{R}^n \to \mathbb{R}^n$ and $T(0) = 0$, such that if $z(0) = T(x(0))$ with $x(0)$ near zero, then $z(t) = T(x(t)), \quad \forall t > 0$. In particular, when T is the identity operator, Eq. (3) is called the *identity observer* for Eq. (1).

Definition 2.1 implies that if Ψ and T are related through the following system of functional equations:

$$T(\Phi(x)) = \Psi(T(x), h(x))$$
$$T(0) = 0, \tag{4}$$

then the y-driven dynamical system (3) represents an observer for system (1), whenever the operator $T(x)$ is invertible. It is important

to note that the initial condition $T(0) = 0$ that accompanies the system of functional Eqs. (4) reflects the fact that equilibrium properties are preserved under the proposed state transformation operator. Please also notice that in the special case of an identity observer, condition (4) reduces to

$$\Phi(x) = \Psi(x, h(x)). \tag{5}$$

Condition (5) expresses the well-known mathematical requirement that the standard definition of a nonlinear observer entails [3,9,54]. Please notice that the vector function $\Psi(z, y)$ in the observer dynamic Eqs. (3) can be arbitrarily selected, as long as the system of functional Eqs. (4) admits an invertible solution $T(x)$. Furthermore, any stability requirement can be imposed at the observer design stage, where Ψ will be *a priori* selected to ensure asymptotic stability of the observer (3). For simplicity reasons, it would be more practical to request linear observer dynamics in the transformed states z up to an output injection term by choosing:

$$\Psi(z, y) = Az + b(y), \tag{6}$$

where A is a constant matrix and b a vector function defined on $\mathbb{R}$ with appropriate dimensionalities. Please notice that the asymptotic stability of the observer in Eq. (3) can now be enforced by appropriately selecting the eigenvalues of the matrix A. Under such a requirement, the operator $T(x)$ of Definition 2.1 ought to satisfy the following system of first-order non-homogeneous functional equations:

$$\begin{aligned} &T(\Phi(x)) = AT(x) + b(h(x)) \\ &T(0) = 0, \end{aligned} \tag{7}$$

with $T : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ being the solution of (7). Within the above single-step exact observer linearization method, the following Theorem ensures the existence of a nonlinear discrete-time observer in the sense of Definition 2.1 [15]:

Theorem 2.1. *Suppose that Assumptions 2.1 and 2.2 hold true. Let* $B = \frac{\partial b}{\partial y}(0)$ *and assume that the pair of matrices* $\{A, B\}$ *is chosen to be controllable and the eigenvalues* $k_i, (i = 1, \dots, n)$ *of matrix* $F = \frac{\partial \Phi}{\partial x}(0)$ *are not related to the eigenvalues* $\lambda_i, (i = 1, \dots, n)$ *of matrix* A *through any equation of the form:*

$$\prod_{i=1}^{n} k_i^{m_i} = \lambda_j \tag{8}$$

$(j = 1, \dots, n)$*, where all the* m_i*'s are non-negative integers that satisfy the condition:*

$$\sum_{i=1}^{n} m_i > 0 \tag{9}$$

Then, there exist a unique analytic and locally invertible solution $T(x)$ *to the system of functional Eqs. (7) as well as a nonlinear map* $z = T(x)$ *that make the dynamical system:*

$$z(t+1) = Az(t) + b(y(t)) \tag{10}$$

an observer for system (1) in the sense of Definition 2.1.

Notice that the proposed nonlinear discrete-time state observer expressed in the original state variables attains the following form:

$$\hat{x}(t+1) = T^{-1}\left[T(\Phi(\hat{x}(t))) + b(y(t)) - b(h(\hat{x}(t)))\right] \tag{11}$$

where $\hat{x} \in \mathbb{R}^n$ is the state estimate and $T(x)$, $T : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is the operator that is constructed by the solution to the associated system of functional Eqs. (7). The structure of the above observer is mathematically realized through the following two terms:

(i) $\Phi(\hat{x})$ is a replica of the system dynamics (1), and

(ii) $b(y(t)) - b(h(\hat{x}(t)))$ (which drives the observer dynamics in Eq. (11) is a feedback term that accounts for the mismatch between the actual sensor measurement $y(t)$ and its estimate $h(\hat{x}(t))$.

Furthermore, the proposed state observer in Eq. (10) induces linear estimation error dynamics with assignable dynamic modes in the transformed variables $z = T(x)$:

$$\begin{aligned} e_z(t+1) &= z(t+1) - \hat{z}(t+1) = T(x(t+1)) - T(\hat{x}(t+1)) \\ &= T(\Phi(x(t))) - T(\Phi(\hat{x}(t))) - b(y(t)) + b(h(\hat{x}(t))) \\ &= AT(x(t)) + b(h(x(t))) - AT(\hat{x}(t)) - b(h(\hat{x}(t))) - b(y(t)) + b(h(\hat{x}(t))) \\ &= AT(x(t)) - AT(\hat{x}(t)) = A(z(t) - \hat{z}(t)) = Ae_z(t) \end{aligned} \tag{12}$$

Therefore, if the matrix A is chosen with an eigenspectrum ensuring stability, i.e. a set of eigenvalues located inside the unit disc on the complex domain, the magnitude of these eigenvalues will also directly regulate the rate of decay of the estimation error to zero, or equivalently the convergence rate of the state estimate to the actual state. Notice that the local invertibility of the transformation operator $T(x)$ would imply that the state estimate $\hat{x}$ asymptotically approaches the actual state x. It is worth noting, that the choice of the matrix-pair $\{A, B\}$ reflects the design degrees of freedom with which the proposed observer is inherently endowed. The controllability condition is required to ensure the existence and uniqueness of the linearizing transformation operator $T(x)$ whereas the eigenspectrum of matrix A enables the assignment of the desirable rate of convergence of the state estimate to the actual state or equivalently, the dynamic modes of the above estimation error dynamics as shown in [15].

Finally, it should be pointed out, that the above exact observer linearization approach is fundamentally different from the ones presented in other studies (see e.g., in [7–9]) where as a first step, a nonlinear coordinate transformation that transforms the original system (1) into a linear one (up to an output injection term) with a linear output map is sought:

$$\begin{aligned} &z(t+1) = Az(t) + b(y(t)) \\ &y(t) = cz(t) \end{aligned} \tag{13}$$

The design of the observer in [7–9] is then completed in a second step where a standard linear Luenberger observer is used for the transformed system (13):

$$\hat{z}(t+1) = A\hat{z}(t) - L(y - c\hat{z}(t)) + b(y(t)), \tag{14}$$

with L being the Luenberger observer gain and $\hat{z}$ the state estimate in the transformed coordinates. The estimate $\hat{x}$ of the original state vector x is then recovered, through the inverse transformation employed in the first step of the design procedure. Please notice that the requirement of a linear output map in the transformed system (13) introduced in the above two-step approach, represents the main mathematical reason for the emergence of quite restrictive conditions in [7–9]. Within the proposed exact observer linearization framework however, the state observer is a dynamical system that is driven by the measured output variable and designed through an appropriate coordinate transformation in a single step without the redundant requirement of linearity of the output map imposed on the transformed linear system (13).

A practical implementation of the proposed nonlinear discrete-time observer design method requires the development of a solution scheme for the nonlinear functional Eqs. (7). As mentioned earlier, the functions $\Phi(x)$, $h(x)$, as well as the transformation operator $T(x)$ are all locally analytic. Therefore, as suggested in [15], a solution method could be based on a multivariate Taylor series expansion of $\Phi(x)$, $h(x)$ and $T(x)$, followed by a procedure that equates same-order Taylor coefficients of both sides of functional Eqs. (7). As a result, recursive algebraic formulas can be generated that are linear with respect to the Taylor coefficients of the unknown solution. Indeed, one can calculate the Nth order Taylor coefficients of $T(x)$ given the Taylor coefficient values up to the order $N - 1$ already calculated in previous recursive steps. The above linear recursive formulas admit a compact mathematical representation if tensorial notation is used (for more details, see [15,39,55]).

The linearization of Eq. (1) around the equilibrium $(0,0)$ gives:

$$dx(t+1) = \frac{\partial \Phi}{\partial x}(0)dx(t); \tag{15}$$

on the other hand, since $\Phi(x(t)) = x(t+1)$, the linearization of the transformed system (7) around the equilibrium reads:

$$\frac{\partial T}{\partial x}(0)dx(t+1) = \left(A\frac{\partial T}{\partial x}(0) + b\frac{\partial T}{\partial x}(h(0))\frac{\partial y}{\partial x}(0)\right)dx(t). \tag{16}$$

Multiplying both sides of Eq. (15) by $\frac{\partial T}{\partial x}(0)$, the following is obtained:

$$\frac{\partial T}{\partial x}(0)dx(t+1) = \left(\frac{\partial T}{\partial x}(0)\frac{\partial \Phi}{\partial x}(0)\right)dx(t). \tag{17}$$

Consequently, from Eqs. (16),(17), it can be deduced that the nonlinear transformation centered on the equilibrium point (x_0) must meet the following (phase) condition:

$$\frac{\partial T}{\partial x}(0)\frac{\partial \Phi}{\partial x}(0) = A\frac{\partial T}{\partial x}(0) + b\frac{\partial T}{\partial x}(h(0))\frac{\partial y}{\partial x}(0). \tag{18}$$

Notice that the elements of the Jacobian matrix $\frac{\partial T}{\partial x}(0)$ can be computed, by solving the system of Eqs. (18), if Φ and h are explicitly available.

2.1. Physics-informed neural networks approach

Inspired by the work in Lagaris et al. [21], physics-informed neural networks (PINNs) as introduced in Raissi & Karniadakis et al. [56,57] (for a review see also [58]), have the capability of integrating knowledge of the underlying physical laws into the learning process, thus facilitating the solution and development of models against incomplete or noisy datasets. Consider a partial differential equation (PDE) of the form:

$$\mathcal{D}(u(x,t)) = f(u(x,t),x,t), \tag{19}$$

subject to boundary conditions on the spatial domain Ω and initial conditions at time $t = 0$:

$$u(x,t) = g(x) \quad \text{for} \quad x \in \partial\Omega, \tag{20}$$

$$u(x,0) = h(x). \tag{21}$$

$\mathcal{D}$ is a (differential) operator, $u(x,t)$ is the unknown solution, $f(u(x,t),x,t)$ is a given functional, x is the spatial variable, t is the time variable.

PINNs approximate the solution $u(x,t)$ using a neural network $\hat{u}_\theta(x,t)$, where θ represents the network parameters. For a feedforward neural network with L hidden layers, the structure can be written as:

$$\hat{u}_\theta(x) = W^{(L)^T}S^{(L)}\left(W^{(L-1)^T}S^{(L-1)}\left(\dots S^{(2)}\left(W^{(2)^T}S^{(1)}\left(W^{(1)^T}z + \beta^{(1)}\right) + \beta^{(2)}\right) + \cdots\right) + \beta^{(L)}\right) + \beta^{(0)},$$

where: z is the input, in this context $z = (\mathbf{x},t)$, $W^{(1)}, W^{(2)}, \dots, W^{(L)}, W^{(0)}$ are the weight matrices, $\beta^{(1)}, \beta^{(2)}, \dots, \beta^{(L)}, \beta^{(0)}$ are the bias vectors, $S^{(1)}, S^{(2)}, \dots, S^{(L)}$ are vector-valued functions associated with the activation functions, say $\sigma^{(\ell)}$, $\ell = 1, \dots L$ (e.g., ReLU, tanh, sigma).

Using a set of M collocation points for x in Ω, K points in $\partial\Omega$ and N collocation points for t, the total loss function is defined as

$$\mathcal{L}(\theta) = \sum_{i=1}^{M}\sum_{j=1}^{N}\left(\mathcal{R}_{ij}(\hat{u}_\theta(x_i,t_j),x_i,t_j)\right)^2 + \sum_{k=1}^{K}\left(\hat{u}_\theta(x_k) - g(x_k)\right)^2, \tag{22}$$

where

$$\mathcal{R}_{ij}(\hat{u}_\theta(x_i,t_j),x_i,t_j) = \mathcal{D}(\hat{u}_\theta(x_i,t_j)) - f(\hat{u}_\theta(x_i,t_j),x_i,t_j). \tag{23}$$

The optimization problem is to find the network parameters θ that minimize the total loss function:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta). \tag{24}$$

It is worth noting that PINNs can be used to solve both the forward [57,59–65] and inverse problems [57,66–74] as well as, inspired by the work of Chen & Chen in [24], for learning nonlinear functional operators [39,75–77].

Here, exploiting the properties of universal approximation [24], we use a PINN to learn an approximation $\hat{T}(x;\theta)$ of the transformation operator $T(x)$ in Eq. (7). The proposed PINN architecture consists of one feedforward neural network two hidden layers, namely a feedforward neural network with say, N_1, N_2 neurons for the first and second hidden layers respectively, along with a linear output (regression) layer. In our context, the unknown operator $\hat{T}(x;\theta)$ can be approximated as:

$$\hat{T}(x;\theta) = W^{(0)^T}S^{(2)}(W^{(2)^T}S^{(1)}(W^{(1)^T}x + \beta^{(1)}) + \beta^{(2)}) + \beta^{(0)},$$
$$\hat{T} : \mathbb{R}^n \longrightarrow \mathbb{R}^n, \tag{25}$$

where $W^{(0)}$ is a $N_2 \times n$ matrix that encompasses the weights $w_{ji}^{(0)}$ linking the second hidden layer to the linear output layer. The multivariate vector-valued functions $S^{(1)} : \mathbb{R}^{N_1} \to \mathbb{R}^{N_1}$ and $S^{(2)} : \mathbb{R}^{N_2} \to \mathbb{R}^{N_2}$ are associated with the activation functions $\sigma^{(1)}$ and $\sigma^{(2)}$ in the first and second hidden layers, respectively. $W^{(1)}$ is a $n \times N_1$ matrix that encompasses the weights connecting the input to the first hidden layer, while $W^{(2)}$ is a $N_1 \times N_2$ matrix whose elements are the weights linking the first hidden layer to the second hidden layer. The biases of the nodes in the first and second layers are denoted by $\beta^{(1)} \in \mathbb{R}^{N_1}$ and $\beta^{(2)} \in \mathbb{R}^{N_2}$, respectively; $\beta^{(0)} \in \mathbb{R}^n$ encapsulates the bias(es) of the output node(s).

To learn $\hat{T}(x;\theta)$, we considered a certain domain $D \subset \mathbb{R}^n$ around the equilibrium point $(0,0)$ discretized in a grid of M points x_i with $i = 1, \dots, M$. Hence, finding $\hat{T}(x;\theta)$ reduces to the task of minimizing a given loss function. Such a function must contain physics-related constraints which enforce the network (25) to converge to the unknown solution. Particularly the PINN in Eq. (25) must satisfy system (7), thus we add to the loss function the difference between the left-hand side and the right-hand side of the system (7), i.e.

$$r_{ij}^{(1)}\left(x_i,\hat{T}(x_i;\theta)\right) = \hat{T}_j(\Phi(x_i)) - \left(\alpha_j{}^T\hat{T}(x_i) + b_j(h(x_i))\right),$$
$$i = 1,\dots,M, \quad j = 1,2,\dots,n, \tag{26}$$

where $\alpha_j{}^T$ is the jth row of the matrix A, $\hat{T}_j$ is the jth component of $\hat{T}$, and $b_j(h(x_i))$ is the jth component of $b(h(x_i))$. Moreover, solution (25) must satisfy the condition in (7) $\hat{T}(0) = 0$, to render the equilibrium of the system invariant under the transformation $\hat{T}(x)$. The second term of the loss function reads:

$$r_j^{(2)}\left(\hat{T}(0;\theta)) = \hat{T}_j(0;\theta)\right), \quad j = 1,2,\dots,n. \tag{27}$$

Finally, the third term of the loss function is defined as follows:

$$r_{jk}^{(3)}\left(x_k, \frac{\partial \hat{T}_j}{\partial x_k}(0;\theta)\right) = \frac{\partial \hat{T}_j}{\partial x_k}(0;\theta) - \frac{\partial T_j}{\partial x_k}(0), \quad j,k = 1,2,\dots,n, \tag{28}$$

where $\frac{\partial \hat{T}_j}{\partial x_k}(0;\theta)$ is the (j,k)-th element of the Jacobian matrix of $\hat{T}(x;\theta)$ computed at the equilibrium and obtained by solving the system of equations in (18). This third term ensures that the derivatives of the network solution match those of the true solution at the equilibrium point. The sum of (26), (27), (28) is the loss function,

$$\mathcal{L}(\theta) = \sum_{i=1}^{M}\sum_{j=1}^{n}\left(r_{ij}^{(1)}\left(x_i,\hat{T}(x_i;\theta)\right)\right)^2 + \sum_{j=1}^{n}\left(r_j^{(2)}\left(\hat{T}_j(0;\theta)\right)\right)^2 + \sum_{j=1}^{n}\sum_{k=1}^{n}\left(r_{jk}^{(3)}\left(x_k,\frac{\partial \hat{T}_j}{\partial x_k}(0;\theta)\right)\right)^2, \tag{29}$$

to be minimized with respect to the unknown parameters $\theta = \left(W^{(0)}, W^{(2)}, W^{(1)}, \beta^{(0)}, \beta^{(2)}, \beta^{(1)}\right)$ of the FNN given by (25). It is worth noting that in the illustrative examples we consider below, we

assign equal weights to each of the three terms in the loss function expression. Furthermore, it is notable that the second and third terms bear a special role in the minimization process, functioning as pinning conditions. To train the network, we used a greedy zero-th order continuation approach presented in [39], in order to deal with steep gradients resembling singularities in the domain D of interest. In particular, the PINN structure is exploited to solve the system of Eq. (4) progressively using a *nested family* of n subdomains:

$$D_1 \subset D_2 \subset \cdots \subset D_k \subset \cdots \subset D_n = D. \tag{30}$$

The subdomain sizes are chosen to provide a level of sufficient accuracy.

2.2. Inverse of the nonlinear transformation

As stated in Theorem 2.1, the nonlinear transformation $T : \mathbb{R}^n \to \mathbb{R}^n$, is locally invertible and the inverse function, denoted $T^{-1}(z) : \mathbb{R}^n \to \mathbb{R}^n$, has the property that $T^{-1}(T(x)) = x$ for all x in the neighborhood of the equilibrium point x_0. In order to reconstruct the trajectories of the system in the phase space, after learning an approximation of the operator $T(x)$ through the PINN approach, a second step is taken aiming at finding the inverse operator T^{-1}. In the next paragraph, we show how to compute the inverse operator using Newton iterations. Within the proposed framework, simulations are conducted for both the discrete system (1) and the corresponding state observer (10) dynamics since the observer dynamic equations are driven by the measurable states ("sensor measurements") of the discrete system over a specific time horizon. Within such a context, to determine the state estimate in the original coordinates $x(t)$ through the inverse transformation, given the simulated current value of the observer state vector in the transformed coordinates $z(t)$ at each time step, a standard Newton's method is employed by numerically solving w.r.t. $x(t)$ the following equations:

$$G(x) = T(x(t)) - z(t) = 0, \tag{31}$$

where $T(x)$ is the approximation of the operator calculated through the proposed PINN scheme. The objective is to compute the state vector x which satisfies Eqs. (31) at each time step of the simulation of the system's (1) and observer's (10) discrete-time dynamic equations. The iterative update formula for Newton's method in the multivariate case is given by:

$$x_{n+1} = x_n - \left(\frac{\partial G}{\partial x}\right)^{-1} G(x_n)$$

Here, x_n is the current estimate of the root, $\frac{\partial G}{\partial x}$ is the Jacobian matrix of G with respect to x, and $\left(\frac{\partial G}{\partial x}\right)^{-1}$ is its inverse matrix. Clearly, the solution x obtained through this process provides a numerical approximation of the inverse function $T^{-1}(z)$, allowing the reconstruction of the dynamic trajectories/profiles of the system's (1) unmeasurable states. For our computations, we have set the relative and absolute tolerance to be 1E−06.

3. Illustrative benchmark problems

The performance of the proposed method is evaluated in the following two benchmark case studies [15,16].

3.1. Benchmark problem 1

The following nonlinear discrete-time system is considered with a state space representation:

$$x_1(t+1) = exp\left(0.2\frac{x_2(t)}{1+x_2(t)}\right)\sqrt{(1+x_1(t)+x_2(t))} - 1 - 0.4x_2(t) - 0.5\ln(1+x_1(t)+x_2(t))$$

$$x_2(t+1) = 0.5\ln(1+x_1(t)+x_2(t)) + 0.4x_2(t)$$

$$y(t) = x_2(t), \tag{32}$$

where the measured output variable y coincides with the second state variable x_2. Notice that $(x_1, x_2) = (0, 0)$ is an equilibrium point of (32) and $1 + x_1 + x_2 > 0, \quad \forall x_1, x_2 \in \mathbb{R}$ (for well-defined right-hand side functions in the above system of difference equations). In addition, the Jacobian matrix F of (32) evaluated at the equilibrium point is:

$$F = \begin{bmatrix} 0.0 & -0.2 \\ 0.5 & 0.9 \end{bmatrix} \tag{33}$$

The eigenvalues of F are: $\lambda_1 = 0.1298$ and $\lambda_2 = 0.7702$. Consider now the following choice for matrix A:

$$A = \begin{bmatrix} 0.5 & 0.3 \\ 0.5 & 0.4 \end{bmatrix}, \tag{34}$$

with eigenvalues: $\mu_1 = 0.8405$ and $\mu_2 = 0.05915$. Moreover, the following output injection map was chosen:

$$b(y) = b(x_2) = \begin{bmatrix} 0.2\left(\frac{x_2}{1+x_2}\right) - 0.3x_2 \\ 0.0 \end{bmatrix} \tag{35}$$

such that the pair (A, B) is controllable. Under the above choice for A, all other assumptions of Theorem 2.1 are satisfied. Consequently, the associated system of functional Eqs. (7) takes the following form:

$$T_1(u_1, u_2) = 0.5T_1 + 0.3T_2 + 0.2\left(\frac{x_2}{1+x_2}\right) - 0.3x_2$$

$$T_2(u_1, u_2) = 0.5T_1 + 0.4T_2$$

$$T_1(0,0) = 0$$

$$T_2(0,0) = 0, \tag{36}$$

where $u_1 = exp\left(0.2\frac{x_2}{1+x_2}\right)\sqrt{(1+x_1+x_2)} - 1 - 0.4x_2 - 0.5\ln(1+x_1+x_2)$ and $u_2 = 0.5\ln(1+x_1+x_2) + 0.4x_2$; in addition (7) admits a unique real analytic and locally (around the equilibrium point of interest) invertible solution. In particular, the solution can be analytically derived and expressed in closed-form as shown below:

$$T_1(x_1, x_2) = \ln(1+x_1+x_2), \quad T_2(x_1, x_2) = x_2. \tag{37}$$

Since:

$$\frac{\partial T}{\partial x}(0,0) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \tag{38}$$

and $det[T] \neq 0$, the above solution $T(x)$ is indeed locally invertible around the equilibrium point $(x_1, x_2) = (0, 0)$. According to Theorem 2.1, the proposed discrete-time observer expressed in the transformed state variables z is given by:

$$\hat{z}_1(t+1) = 0.5\hat{z}_1(t) + 0.3\hat{z}_2(t) + 0.2\frac{y(t)}{1+y(t)} - 0.3y(t)$$

$$\hat{z}_2(t+1) = 0.5\hat{z}_1(t) + 0.4\hat{z}_2(t) \tag{39}$$

Therefore, through the inverse transformation, the state estimates expressed in the original state variables are given by the following equations:

$$\bar{x}_1(t) = exp(\hat{z}_1(t)) - \hat{z}_2(t) - 1$$

$$\bar{x}_2(t) = \hat{z}_2(t) \tag{40}$$

please notice that the nonlinear operator in the first benchmark example exhibits singularities when $x_1 + x_2 = -1$ in the system's state space. The primary objective is to learn an approximation of the transformation operator in the domain $[x_L, 0] \times [x_L, 0] = [-0.495, 0] \times [-0.495, 0]$.

Fig. 1 depicts the analytical solutions $T_1(x_1, x_2), T_2(x_1, x_2)$, of the associated system of functional Eqs. (36). We note that it encompasses the solution domain which has been deliberately chosen to be $x_1, x_2 \in [-0.495, 0]$ since $T_1(x_1, x_2)$ exhibits a singular point at $(x_1, x_2) = (-0.5, -0.5)$. Therefore, the first reasonable step would be to

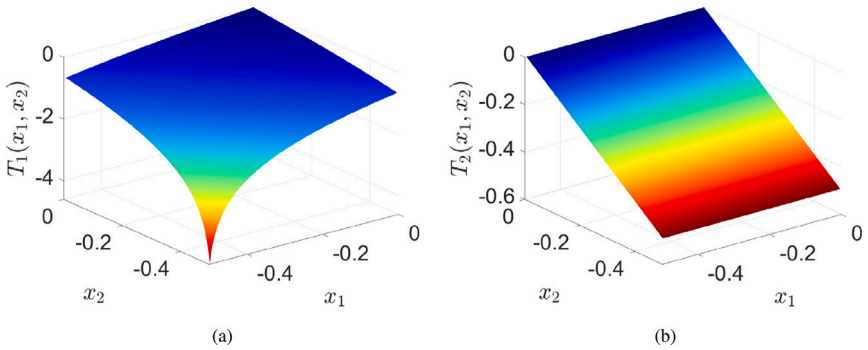


Fig. 1. Analytical solution of the NFEs (36) in $[-0.495, 0] \times [-0.495, 0]$. (a) $T_1(x_1, x_2) = \ln(1 + x_1 + x_2)$. A steep-gradient at $(-0.495, -0.495)$ is due to the presence of a singular point at $(x_1, x_2) = (-0.5, -0.5)$. (b) $T_2(x_1, x_2) = x_2$.

comparatively evaluate the numerical approximation accuracy of the proposed PINN scheme in this region against the existing analytical solution and assess its impact on the performance profile of the resulting discrete-time observer.

3.2. Benchmark problem 2

Consider also the following nonlinear discrete-time dynamical system [16]:

$$\begin{aligned} x_1(t+1) &= \frac{0.5\frac{x_1(t)}{1+x_1(t)} - 0.9x_2(t)}{1 - 0.5\frac{x_1(t)}{1+x_1(t)} + 0.9x_2(t)} \\ x_2(t+1) &= x_2(t) \\ y(t) &= x_1(t) \end{aligned} \tag{41}$$

where x_2 represents an unidentified constant parameter or disturbance term which needs to be estimated and x_1 the measured state variable. Please notice that $(x_1, x_2) = (0, 0)$ is an equilibrium point of the above system and the Jacobian matrix F evaluated at this point is:

$$F = \begin{bmatrix} 0.5 & -0.9 \\ 0.0 & 1 \end{bmatrix}. \tag{42}$$

The eigenvalues of F are: $\lambda_1 = 0.5$ and $\lambda_2 = 1$, and therefore, the original nonlinear discrete-time observer design method presented in [15] cannot be applied since the second eigenvalue is not located within the Poincaré domain; instead it lies on the unit circle. However, under the following choice for the matrix A

$$A = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.1 \end{bmatrix} \tag{43}$$

with eigenvalues: $\mu_1 = 0$ and $\mu_2 = 0.1$, and an output injection map:

$$b(y) = b(x_1) = \begin{bmatrix} 0.5\left(\frac{x_1}{1+x_1}\right) \\ \left(\frac{x_1}{1+x_1}\right) \end{bmatrix} \tag{44}$$

all assumptions in the method's extension presented in [16] (where the Poincaré domain assumption was relaxed and replaced by a more general one of the Siegel-type) are met, and therefore, the proposed nonlinear discrete-time observer exists and can be similarly designed. Indeed, the associated system of functional equations takes the following form:

$$\begin{aligned} T_1\left(\frac{0.5\frac{x_1}{1+x_1} - 0.9x_2}{1 - 0.5\frac{x_1}{1+x_1} + 0.9x_2}, x_2\right) &= 0.5\frac{x_1}{1+x_1} \\ T_2\left(\frac{0.5\frac{x_1}{1+x_1} - 0.9x_2}{1 - 0.5\frac{x_1}{1+x_1} + 0.9x_2}, x_2\right) &= 0.1T_2 + \frac{x_1}{1+x_1} \\ T_1(0,0) &= 0 \\ T_2(0,0) &= 0, \end{aligned} \tag{45}$$

The above system of functional Eqs. (45) admits a unique analytically derived closed-form solution shown below:

$$\begin{aligned} T_1(x_1, x_2) &= \frac{x_1}{1+x_1} + 0.9x_2 \\ T_2(x_1, x_2) &= \frac{5}{2}\left(\frac{x_1}{1+x_1} + x_2\right). \end{aligned} \tag{46}$$

Since:

$$\frac{\partial T}{\partial x}(0,0) = \begin{bmatrix} 1 & 0.9 \\ 2.5 & 2.5 \end{bmatrix}, \tag{47}$$

and $det[T] \neq 0$, the above solution $T(x)$ is locally invertible around the origin. The proposed discrete-time observer is then given by the following dynamic equations:

$$\begin{aligned} \hat{z}_1(t+1) &= 0.5\frac{y(t)}{1+y(t)} \\ \hat{z}_2(t+1) &= 0.1\hat{z}_2(t) + \frac{y(t)}{1+y(t)}. \end{aligned} \tag{48}$$

Therefore, using the inverse transformation map, the state estimates expressed in the original state variables are given by the following equations:

$$\begin{aligned} \bar{x}_1(t) &= \frac{10\hat{z}_1(t) - 3.6\hat{z}_2(t)}{1 - 10\hat{z}_1(t) + 3.6\hat{z}_2(t)} \\ \bar{x}_2(t) &= 4\hat{z}_2(t) - 10\hat{z}_1(t). \end{aligned} \tag{49}$$

In this second benchmark problem, the nonlinear transformation operator exhibits a singularity when $x_1 = -1$. Here, we aim to learn this map in the domain $[x_L, 0] \times [x_L, 0] = [-0.91, 0] \times [-0.91, 0]$.

Fig. 2 depicts the analytical solutions $T_1(x_1, x_2), T_2(x_1, x_2)$, of the associated system of functional Eqs. (45). Please notice that it includes the solution domain, which in this case is $(x_1, x_2), \in [-0.91, 0]$ because $T_1(x_1, x_2)$ displays a singular point at $(x_1, x_2) = (-1, -1)$. As in the previous benchmark example, the focus is placed on the evaluation of the impact of the proposed PINN scheme on the performance profile of the resulting nonlinear observer by comparing the identified transformation operator to the analytical one and hence, demonstrating the enhanced numerical approximation accuracy in this domain attained by PINN.

4. Numerical results

We utilized the Matlab deep-learning toolbox, with the aid of which a custom code was developed to implement the Physics-Informed Neural Network (PINN) strategy. In this process, we employed the Levenberg–Marquardt (LM) algorithm to optimize the learnable parameters. The scheme utilizes finite differences for computing the required derivatives. However, it is worth noting that these derivatives can also be computed analytically, numerically or through Automatic Differentiation (AD), as demonstrated in [39]. The simulation results obtained when the PINN was trained in the entire domain without the

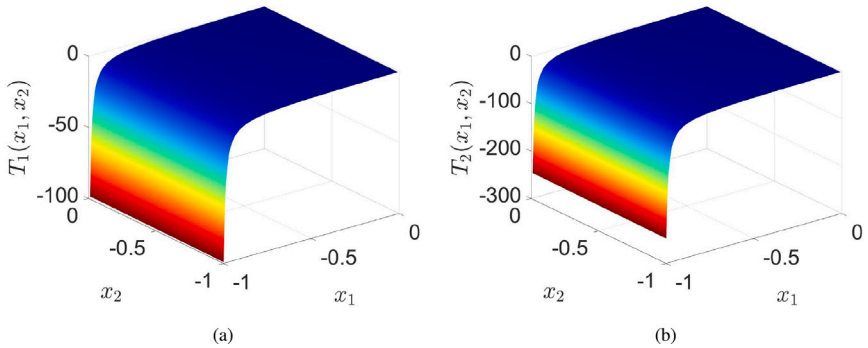


Fig. 2. Analytical solution of the functional Eqs. (45). (a) $T_1(x_1, x_2) = \frac{x_1(t)}{1+x_1(t)} + 0.9x_2$. (b) $T_2(x_1, x_2) = \frac{5}{2}\left(\frac{x_1(t)}{1+x_1(t)} + x_2\right)$. A steep-gradient at $(-0.91, -0.91)$ is due to the presence of a singular point at $x_1 = -1$.

greedy approach are also presented in order to evaluate the efficacy of the proposed greedy strategy. For training, we used 15×15 equispaced distributed points in the interval $[a, b]$: $a = -0.495$, $b = 0$, for the model (32), and $a = -0.91$, $b = 0$, for the model (41). For the test set, we employed Chebyshev-Lobatto nodes, specifically selecting the roots of the (second kind) Chebyshev-Lobatto polynomial of degree k, i.e.

$$x_n = \frac{1}{2}(a+b) - \frac{1}{2}(a-b)cos\left(\frac{2n-1}{2n}\pi\right) \quad n = 1, \dots, k \tag{50}$$

Power-series solution. In accordance with the theoretical framework outlined in Ref. [15], we also used the power series solution method delineated in Section 2, to find the nonlinear transformation. In particular, we used sixth-order multivariate Taylor polynomial approximations. The method produces a system of linear algebraic equations involving the unknown Taylor coefficients of the nonlinear transformation $T(x_1, x_2)$. The solution to this system can be computed using MATLAB's symbolic toolbox, as explained in Section 2. We opt for a sixth-order multivariate Taylor polynomial expansion for two reasons: firstly, the number of coefficients aligns well with the number of unknowns in our PINN, facilitating a fair comparison between the two schemes. Secondly, solving the system for an expansion of this order is already computationally challenging, highlighting the underlying issues of this traditional methodological approach usually employed to solve this type of problems for high order series expansions.

PINN-based solution. We have used feedfordward network (FNNs) with two hidden layers, with each layer containing five neurons. We have used sigmoid activation functions $\sigma(x)$. In the training process, we computed the derivatives for the optimization process using finite differences for the LM algorithm and we have set the maximum function evaluations to 500,000 while the maximum iteration parameter was set to 50,000. the finite difference step was set to $1e^{-05}$ and the step size tolerance was set to $1e^{-12}$. This realization prompted the exploration of alternative strategies to enhance the training process. Subsequently, the adoption of a greedy-wise procedure emerged as a pivotal step in refining the PINN's convergence behavior. By incorporating this approach, the greedy-wise procedure effectively addressed challenges related to convergence of the PINN close to the singular points.

Greedy-wise procedure. We initiated training within the region $[-0.1, 0] \times [-0.1, 0]$. Subsequently, we systematically expanded the grid size in the domain, with a varying step size in the grid depending on the specific benchmark problem at hand. In the first benchmark problem, we employed a grid step size of -0.1 in both directions, until reaching a domain size of $[-0.4, 0] \times [-0.4, 0]$, then the step size is decreased to a value of -0.01 until reaching $[-0.49, 0] \times [-0.49, 0]$. Finally, the step size is decreased again to a value of -0.001 until reaching the domain of interest, i.e. $[-0.495, 0] \times [-0.495, 0]$, while for the second benchmark problem, we initiated training within the region $[-0.1, 0] \times [-0.1, 0]$ and we adopted a step of -0.1 until reaching a grid of $[-0.8, 0] \times [-0.8, 0]$. After that we selected a step size of -0.01 until reaching the domain of interest $[-0.91, 0] \times [-0.91, 0]$. In both instances, following the completion of training within each subdomain, we utilized the learnable parameters specifically, the weights and biases obtained from the preceding subdomain as the initial guess for the optimization algorithm in the subsequent subdomain.

4.1. Benchmark problem 1

Fig. 3 depicts the numerical approximation accuracy, measured as the difference between computed and analytical solutions for the training set, which is built using a grid of 15×15 nodes uniformly distributed. The results using the 6th-order power series expansion of both the nonlinear transformation and the right-hand side of the discrete model are displayed in subplots Fig. 3(a) and (b), respectively. As expected, the power series expansion yields zero error for the $T_2(x_1, x_2)$ component and demonstrates good approximation accuracy for the $T_1(x_1, x_2)$ component, but primarily in the vicinity of the linearization-relevant point at $[0, 0]$. Beyond this point, the numerical approximation deteriorates significantly, reaching the order of 10^0 at the grid edge, particularly close to the singular point. This situation could potentially undermine the precision with which the observer estimates the unmeasured state.

Fig. 3(c),(d) depict the approximation accuracy of the PINN, when trained in the entire domain (without the greedy approach). As observed, the approximation accuracy of the PINN remains rather poor, of the order of 1 in the vicinity of the singularity. Fig. 3(e),(f) illustrate the approximation error using PINN with the greedy-wise training approach. As shown, the performance of the PINN is significantly better compared to the one involving training across the entire domain. Furthermore, Fig. 4 showcases the performance of the schemes for the test set, which consist of a grid of 20×20 Chebyshev-Lobatto distributed nodes. Please notice that these results are similar to those observed in the training set.

Table 1 presents a comparison of L_1, L_2, L_∞ error norms between the analytical and numerical solutions on the test set. Specifically, it summarizes the results obtained with the 6th order power series expansions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$, and also with PINN solution approximations both trained in the entire domain at once using the greedy approach. The reported errors include the median, 5th percentile, and 95th percentile over 100 runs.

Table 2, on the other hand, reports the results of uncertainty quantification carried out to investigate the convergence of the PINN scheme in approximating the nonlinear transformation operator $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ on a test set involving the use of a grid of 20×20 Chebyshev-Lobatto distributed points. The investigation encompassed 100, 200, 300, and 400 independent runs, providing a thorough evaluation of the resulting solution's variability.

Furthermore, Fig. 5 depicts the error characteristics between the analytical solution and the outcome generated by the PINN for $T_1(x_1, x_2)$.

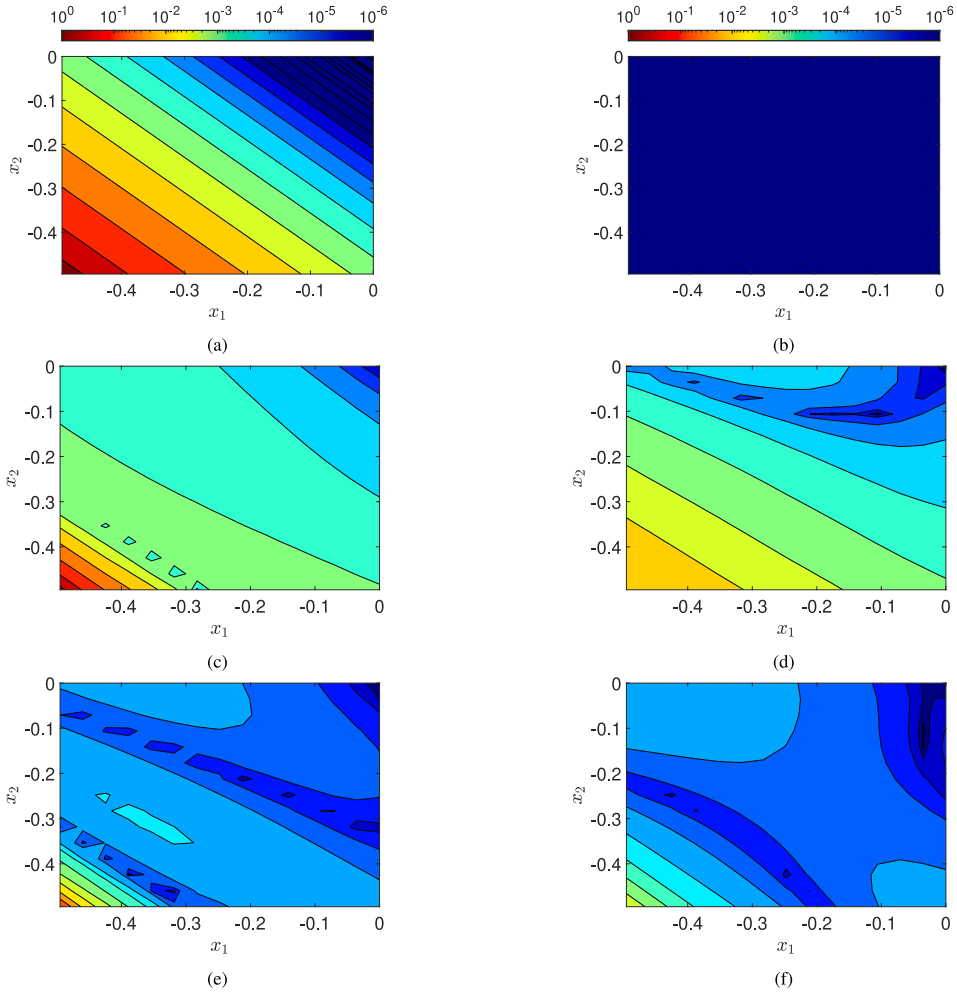


Fig. 3. Benchmark problem 1. Training sets (grids of 15 × 15 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) 6th order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (36) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PINN trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Table 1
Benchmark problem 1. Test set using 20 × 20 Chebyshev-Lobatto distributed points L_1, L_2, L_∞ error norms (median, 5th-95th percentiles) between the analytical and computed solutions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ of the various schemes trained in the entire domain and with the greedy approach, over 100 runs.

$T(x_1, x_2)$	Error norm	Power-series 6th order –	PINN entire domain Median (5th, 95th)	PINN greedy Median (5th, 95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.89E+01	1.186E+01,(1.048E+01, 1.372E+01)	6.051E−01,(6.94E−02, 6.00E+00)
	$\|\cdot\|_2$	4.55E+00	1.062E+01,(9.94E+00, 1.243E+01)	1.528E−01,(1.62E−02, 1.33E+00)
	$\|\cdot\|_\infty$	2.88E+00	8.28E+00,(7.00E+00, 9.63E+00)	9.15E−02,(7.80E−03, 5.87E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	0	2.44E+00,(7.40E−01, 4.00E+00)	3.595E−01,(4.39E−02, 2.90E+00)
	$\|\cdot\|_2$	0	2.34E+00,(6.68E−01, 3.75E+00)	6.31E−02,(1.02E−02, 4.988E−01)
	$\|\cdot\|_\infty$	0	1.65E+00,(4.55E−01, 2.68E+00)	2.94E−02,(3.90E−03, 1.85E−01)

In this benchmark problem, the transformation operator is quite important as it encompasses the singular point and relates to the state assumed to be unmeasurable. In Fig. 5(a), we juxtapose the solution approximations obtained through training across the entire domain and in a greedy-wise manner. The L_1 norm for both PINN approximations is presented; the blue line signifies the median of the L_1 error evaluated at specific points in the domain. Additionally, the green band represents a confidence interval spanning from the 5th to the 95th percentile, reflecting a PINN trained in a greedy-wise fashion. On the other hand, the black line depicts the median of the L_1 error when the network is trained across the entire domain at once, and the green band denotes a confidence interval ranging from the 5th to the 95th percentile. Fig. 5(b), depicts the L_2 norm; the blue line depicts the median of the L_2 error with a confidence interval ranging from the 5th to the 95th percentile represented by the green band; the black line shows again the median of the L_2 error when the network is trained across the entire domain at once, considering the blue band as a confidence interval in the same sense as before. Fig. 5(c) displays results for the L_∞ norm. It can be inferred that the greedy-wise training significantly outperforms the one obtained when training the network across the entire domain in a single step. The numerical approximation from the single training exhibits a poorer performance, particularly in regions proximate to the singular point, as is also evident in Fig. 4.

Finally, Fig. 5(d) depicts the difference between the analytical inverse operator $T_1^{-1}(x_1, x_2)$ (49) of the observed variable (48) and the numerical approximation of the inverse $\hat{T}_1^{-1}(x_1, x_2)$ as computed by

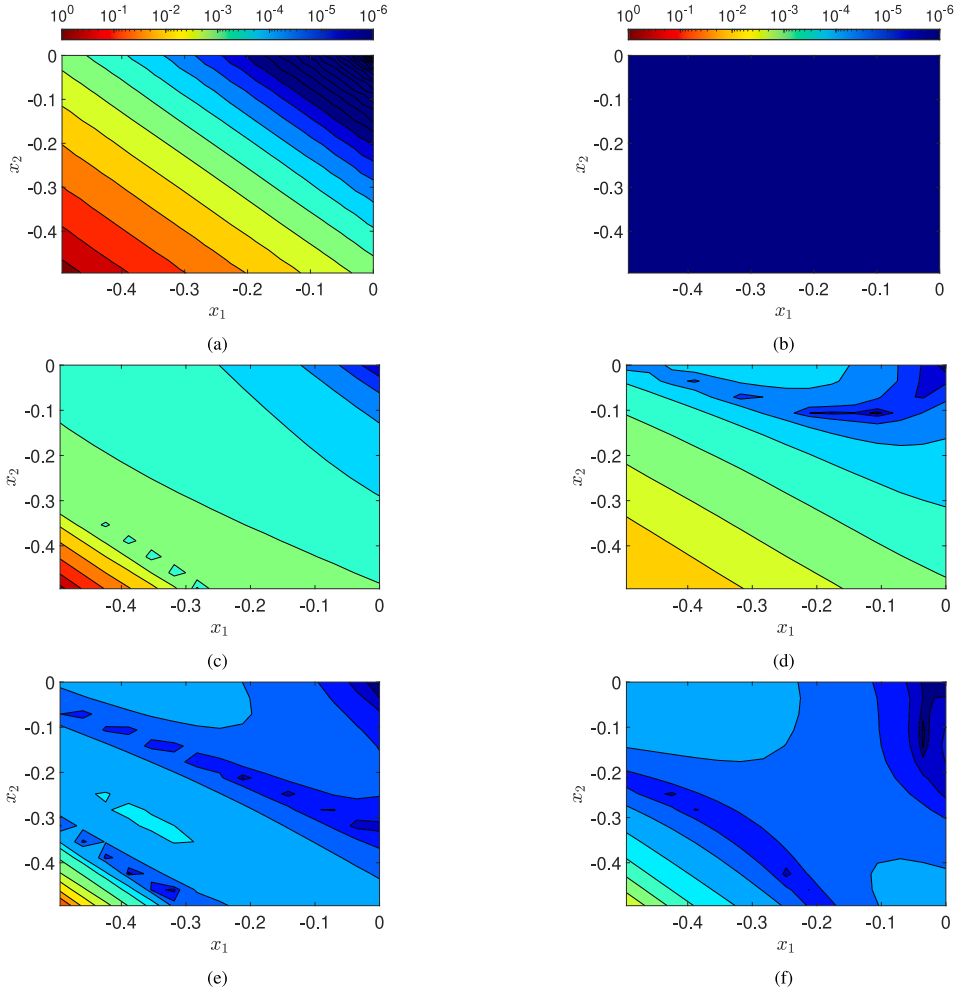
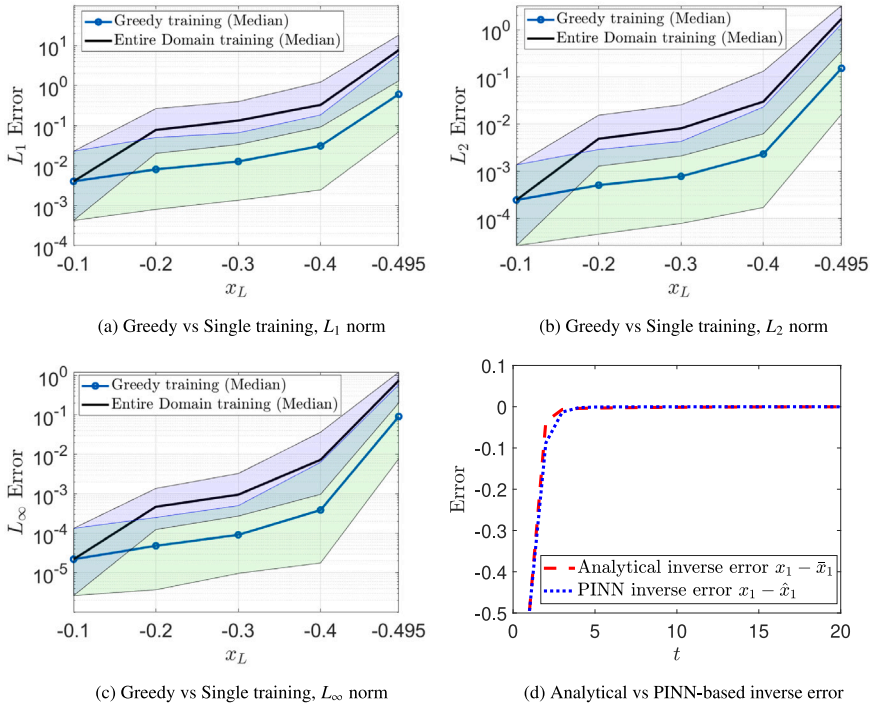


Fig. 4. Benchmark problem 1. Test sets (grids of 20 × 20 Chebyshev-distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) 6th order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (36) in $[-0.495, 0] \times [-0.495, 0]$. (c),(d) PINN trained in the entire domain $[-0.495, 0] \times [-0.495, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Table 2
Benchmark problem 1. Uncertainty quantification systematically conducted on a test set employing a grid of 20 × 20 Chebyshev-Lobatto distributed points to investigate the convergence behavior of the Physics-Informed Neural Networks (PINN). The study involves performing 100, 200, 300, and 400 independent runs to thoroughly assess the variability in the results. The analysis focused on capturing the central tendency by calculating the median and examining the dispersion of outcomes through the 5th and 95th percentile confidence intervals.

$T(x_1, x_2)$	Error norm	100 runs Median (5th, 95th)	200 runs Median (5th, 95th)	300 runs Median (5th, 95th)	400 runs Median (5th, 95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	6.051E−01, (6.94E−02, 6.00E+00)	6.34E−01, (6.20E−02, 5.60E+00)	6.27E−01, (6.188E−02, 5.45E+00)	6.12E−01, (6.12E−02, 5.32E+00)
	$\|\cdot\|_2$	1.528E−01, (1.62E−02, 1.33E+00)	1.77E−01, (1.95E−02, 1.23E+00)	1.72E−01, (1.88E−02, 1.21E+00)	1.705E−01, (1.76E−02, 1.18E+00)
	$\|\cdot\|_\infty$	9.15E−02, (7.80E−03, 5.87E−01)	8.87E−02, (6.58E−03, 5.12E−01)	8.75E−02, (6.55E−03, 5.10E−01)	8.66E−02, (6.502E−03, 5.108E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	3.595E−01, (4.39E−02, 2.90E+00)	3.90E−01, (4.98E−02, 3.25E+00)	3.84E−01, (4.85E−02, 3.21E+00)	3.78E−01, (4.78E−02, 3.16E+00)
	$\|\cdot\|_2$	6.31E−02, (1.02E−02, 4.988E−01)	6.21E−02, (1.80E−02, 6.01E−01)	6.19E−02, (1.75E−02, 6.00E−01)	6.15E−02, (1.69E−02, 5.98E−01)
	$\|\cdot\|_\infty$	2.94E−02, (3.90E−03, 1.85E−01)	2.81E−02, (3.28E−03, 2.10E−01)	2.75E−02, (3.22E−03, 2.03E−01)	2.701E−02, (3.19E−03, 1,98E−01)

Newton's method (presented in Section 2). As shown, the difference between the true state and the inverse transformation value generated by the PINN converges to zero. For this particular simulation, we have initialized the z-states as follows: $z_1(1) = 0$ and $z_2(1) = 0$, and we have used as initial guess for Newton's method the following values for the x-states: $x_0 = [0.1, 0.1]$. Moreover, we have initialized the analytical inverse expression using: $\hat{x}_1(1) = 0$ and $\hat{x}_2(1) = 0$; the actual states were: $x_1 = -0.495$ and $x_2 = 0.35$. Regarding the Newton's method, we have set the relative and absolute tolerances at: $tol < 1E\text{-}06$ (the scheme converged in 5 iterations).



(a) Greedy vs Single training, L_1 norm

(b) Greedy vs Single training, L_2 norm

(c) Greedy vs Single training, L_∞ norm

(d) Analytical vs PINN-based inverse error

Fig. 5. Benchmark problem 1: (a)–(c) Panels involve a comprehensive comparison between the performance of Physics-Informed Neural Networks (PINN) trained using two distinct approaches: one trained on the entire domain at once (Black line) and another trained using a greedy approach (blue line). The evaluation is conducted on a test set comprising 20×20 Chebyshev nodes of the second type. The reported performance metrics include the L_1, L_2, and L_∞ error norms, presented in terms of the median and the 5th to 95th percentiles. The evaluation is performed over 100 independent runs for the transformation $T_1(x_1, x_2)$. Panel (d) depicts a comparison between observation errors. The red line represents the difference between the analytical inverse $\bar{x}_1$ of the transformation $T_1(x_1, x_2)$ and the real state x_1, whereas the blue line depicts the difference between the true state x_1 and its approximation obtained through Newton's method, denoted as $\hat{x}_1$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Benchmark problem 2

This particular benchmark problem introduces the challenge of unavailability of the state x_2, with only the state x_1 being measurable. Consequently, the focus of this problem is on the reconstruction of the second state. As highlighted earlier, the unmeasured state x_2 is viewed as either an unidentified system parameter or unknown disturbance, thus adding complexity to the estimation task. As a consequence and in alignment with the methodology outlined so far, the task is to address the set of functional Eqs. (45) utilizing the established PINN scheme. Fig. 6 depicts the numerical approximation accuracy, as the difference between computed and analytical solutions (46), obtained by various schemes for the training set. This particular benchmark problem is more challenging as the gradients it contains (see Fig. 2) are notably bigger compared to the previous one.

Fig. 6(a),(b) demonstrate the results achieved with a 6th-order truncation in the power series expansion of the nonlinear transformation and the right-hand side of the discrete model. In this particular problem, the steep gradient emanating from the singular point at $x_1 = -1$ poses a significant challenge as mentioned before. The numerical approximation provided by the series expansion method is notably poor across the domain, particularly of the order of 10^1 at the grid's edge, as mentioned previously, near the singular point.

Furthermore, Fig. 6(c),(d) depict the outcomes obtained with the PINN approach to learn the transformation operator (46) across the entire domain without the greedy approach. The approximation accuracy of this scheme is relatively modest, especially in the vicinity of the singularity, approximately of the order of 10^0 for $T_1(x_1, x_2)$ and of the order of 10^1 for $T_2(x_1, x_2)$. Nevertheless, it outperforms the conventional Taylor series expansion method in terms of numerical error accuracy across the entire domain in general.

Fig. 6(e),(f) present the results obtained from the PINN implementation with the greedy approach. Notably, the adoption of the greedy-wise training method substantially enhances the numerical approximation accuracy when compared to PINN schemes trained across the entire domain in a single training step. Here, the numerical approximation error is of the order of 10^{-2} in the region close to the edge point $(-0.91, -0.91)$. Additionally, Fig. 7 portrays the performance of the schemes when tested on a Chebyshev-Lobatto grid, with similar results observed for both the training and test sets.

Table 3 allows a comparison of the approximation errors between the analytical and numerical solutions. Specifically, the 6th order power series expansions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ with the two PINN solution approximations (with and without the greedy approach) are compared. To test the convergence of the schemes, we used a Chebyshev-Lobatto grid with 20×20 points. The reported errors include the median, 5th percentile, and 95th percentile aggregated over 100 runs.

Table 4, illustrates the results of uncertainty quantification performed to investigate the convergence of the PINN scheme in approximating the nonlinear transformation maps $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ on a test set (a grid of 20×20 Chebyshev-Lobatto distributed points). In our computations, we considered 100, 200, 300, and 400 runs. As shown, the medians and variances, are for any practical means of the same order, thus indicating convergence of the training process.

Fig. 8 depicts the approximation error behavior between the analytical solution and the one produced by PINN for $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$. Please notice that $T_2(x_1, x_2)$ is the transformation map of interest in this problem because it is related to the state we seek to observe as mentioned earlier. The approximation accuracy of the PINN trained both across the entire domain and in a greedy-wise manner is assessed. Fig. 8(a) shows the L_1 norm of the transformation map $T_1(x_1, x_2)$. The blue line signifies the median, and the green band represents a confidence interval spanning the 5th to the 95th percentile range for the PINN model trained in a greedy-wise fashion. The black

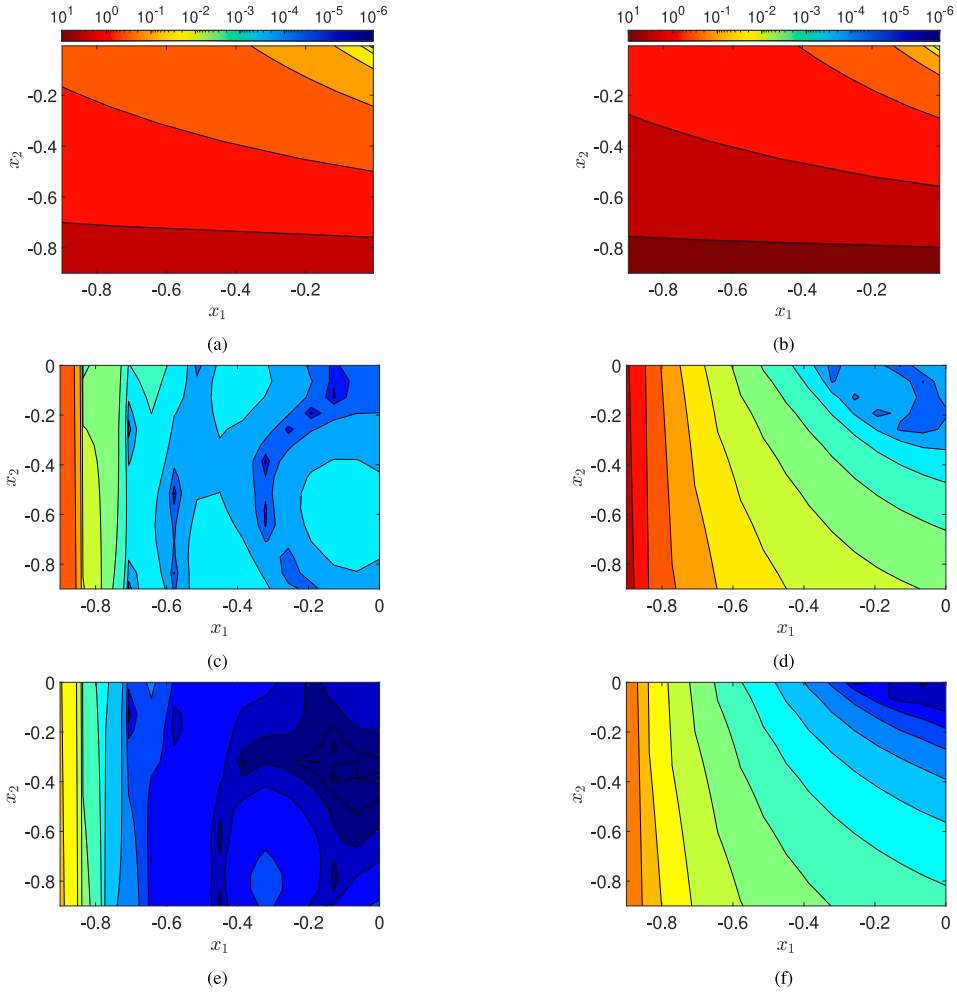


Fig. 6. Training sets (grids of 15 × 15 equispaced distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) 6th order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (45) in $[-0.91, 0] \times [-0.91, 0]$. (c),(d) PINN trained over the entire domain $[-0.91, 0] \times [-0.91, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Table 3
Test sets consist of grids with dimensions of 20 × 20 Chebyshev-distributed points. Error norms, including L_1, L_2, and L_∞, quantify the disparities between the analytical and computed solutions of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ using various training schemes. These schemes encompass training both with a greedy-wise approach and across the entire domain $[-0.91, 0] \times [-0.91, 0]$. The analysis of error norms includes statistics such as the median and 95th percentiles, aggregated over 400 runs, for each norm corresponding to both $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$.

$T(x_1, x_2)$	Error norm	Power-series 6th order	PINN entire domain Median(5th, 95th)	PINN greedy Median(5th, 95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	5.19E+02	7.03E+01,(5.75E+01, 8.31E+01)	2.17E+00,(3.43E−01, 2.60E+01)
	$\|\cdot\|_2$	4.13E+01	2.16E+01,(1.73E+01, 2.62E+01)	3.55E−01,(4.52E−02, 3.48E+00)
	$\|\cdot\|_\infty$	1.71E+01	9.39E+00,(7.32E+00, 1.16E+01)	3.37E−01,(1.78E−02, 1.058E+00)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	5.87E+01	2.14E+02,(1.82E+02, 2.15E+02)	1.72E+01,(8.25E−01, 7.50E+01)
	$\|\cdot\|_2$	5.30E+01	6.92E+01,(5.84E+01, 8.04E+01)	2.02E+00,(1.11E−01, 1.92E+01)
	$\|\cdot\|_\infty$	4.40E+01	3.40E+01,(2.89E+01, 4.02E+01)	1.90E+00,(5.36E−02, 3.99E+00)

line and the blue band corresponds to the PINN model trained across the entire domain without the greedy approach. Fig. 8(b) depicts the corresponding L_2 error norms, and Fig. 8(c) depicts the corresponding L_∞ error norms. Fig. 8(d–f) depict similar results for $T_2(x_1, x_2)$.

It is evident, that the greedy-wise training procedure outperforms the single training one for the approximation of $T_2(x_1, x_2)$. Finally, Fig. 9 depicts the error between the trajectories computed by applying the analytically derived inverse $T^{-1}(x_1, x_2)$ (49) to the observer (48) and the numerical approximation of the inverse $\hat{T}^{-1}(x_1, x_2)$ computed by Newton's method. For this task, we have initialized the z-states as: $z_1(1) = 0$ and $z_2(1) = 0$, and we have used as an initial guess the values $x_0 = [0.1, 0.1]$. The analytical inverse expression was initialized using the values: $\hat{x}_1(1) = 1$ and $\hat{x}_2(1) = -0.4$. Moreover, as mentioned earlier, the tolerance level for Newton's method was set at *tol* <1E−06.

5. Conclusions

In the present research study, a novel Physics-Informed Machine Learning (PINN) methodological and simulation framework is introduced to solve the nonlinear state observation problem in the discrete-time domain. This procedure stands in contrast to conventional methodologies [1,78], as it is executed in a single step. The proposed PINN

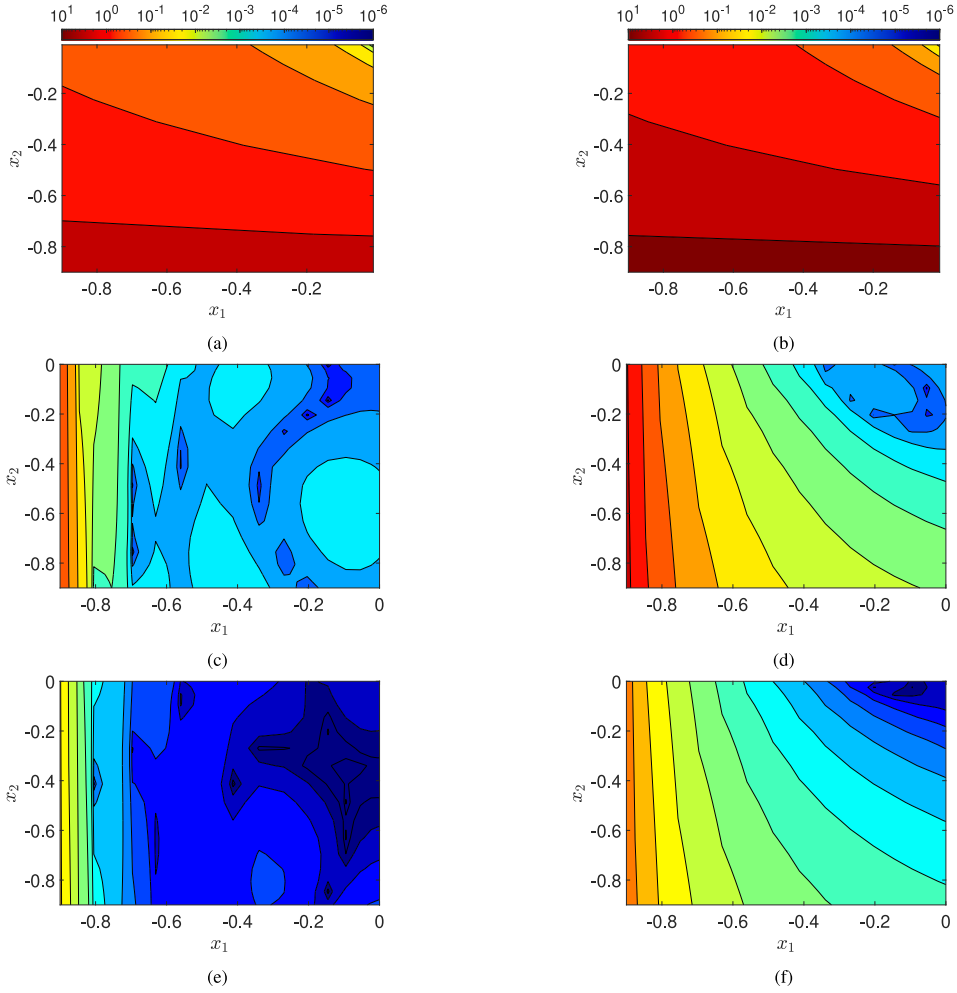
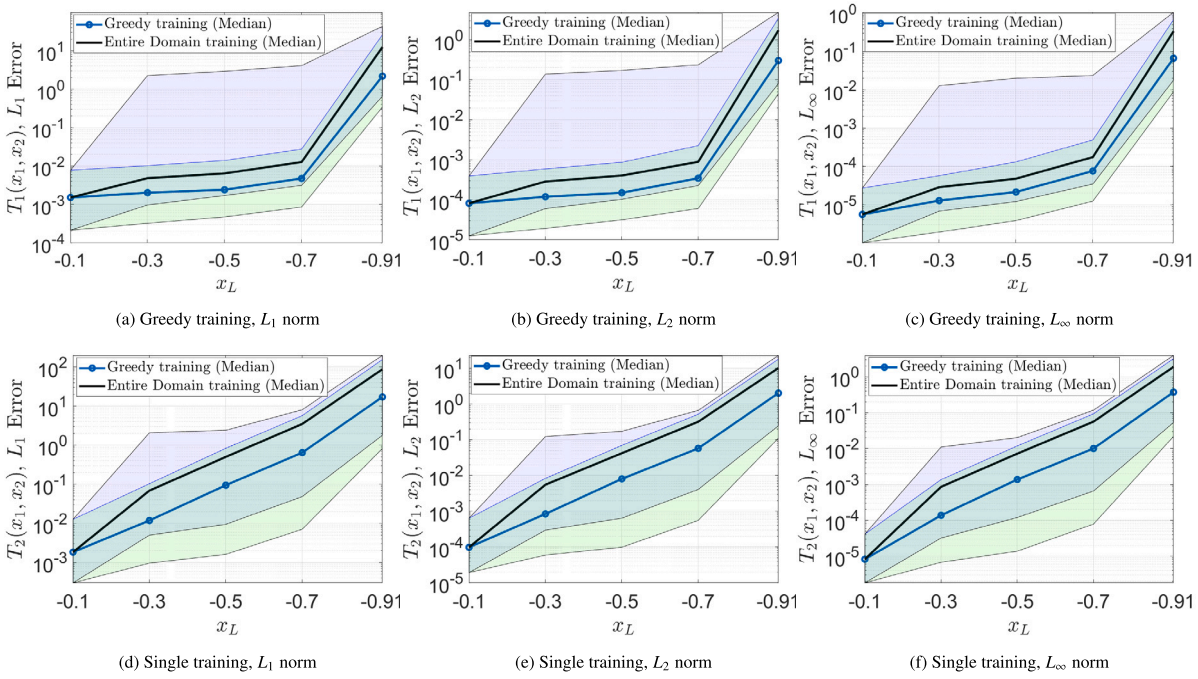


Fig. 7. Test sets (grids of 20 × 20 Chebyshev-Lobatto distributed points). Numerical approximation accuracy (difference between the computed and analytical solution) of $T_1(x_1, x_2)$ (left column) and $T_2(x_1, x_2)$ (right column) using the various schemes. (a),(b) 6th order power-series expansion of $T_1(x_1, x_2)$ and $T_2(x_1, x_2)$ and the right-hand side of the model (45) in $[-0.91, 0] \times [-0.91, 0]$. (c),(d) PINN trained over the entire domain $[-0.91, 0] \times [-0.91, 0]$. (e),(f) PINN trained via the greedy-wise approach.

Table 4
Benchmark problem 1. Uncertainty quantification systematically conducted on a test set employing a grid of 20 × 20 Chebyshev-Lobatto distributed points to investigate the convergence behavior of the Physics-Informed Neural Networks (PINN). The study involves performing 100, 200, 300, and 400 independent runs to thoroughly assess the variability in the results. The analysis focused on capturing the central tendency by calculating the median and examining the dispersion of outcomes through the 5th and 95th percentile confidence intervals.

$T(x_1, x_2)$	Error norm	100 runs Median (5th, 95th)	200 runs Median (5th, 95th)	300 runs Median (5th, 95th)	400 runs Median (5th, 95th)
$T_1(x_1, x_2)$	$\|\cdot\|_1$	2.17E+00, (3.43E−01, 2.60E+01)	2.88E+00, (3.97E−01, 2.81E+01)	2.83E+00, (3.86E−01, 2.721E+01)	2.72E+00, (3.81E−01, 2.715E+01)
	$\|\cdot\|_2$	3.55E−01, (4.52E−02, 3.48E+00)	4.23E−01, (5.12E−02, 3.93E+00)	4.01E−01, (5.076E−02, 3.66E+00)	3.98E−01, (4.93E−02, 3.563E+00)
	$\|\cdot\|_\infty$	3.37E−01, (1.78E−02, 1.0581E+00)	3.90E−01, (1.69E−02, 1.210E+00)	3.84E−01, (1.03E−02, 7.01E−01)	3.81E−01, (1.00E−02, 6.91E−01)
$T_2(x_1, x_2)$	$\|\cdot\|_1$	1.72E+01, (8.25E−01, 7.50E+01)	1.95E+01, (8.76E−01, 7.85E+01)	1.49E+01, (8.39E−01, 6.55E+01)	1.20E+01, (8.32E−01, 6.13E+01)
	$\|\cdot\|_2$	2.02E+00, (1.11E−01, 1.92E+01)	2.32E+00, (1.52E−01, 2.05E+01)	1.787E+00, (1.23E−01, 1.786E+01)	1.59E+00, (1.09E−01, 1.778E+01)
	$\|\cdot\|_\infty$	1.90E+00, (5.36E−02, 3.99E+00)	1.92E+00, (5.37E−02, 4.00E+00)	1.72E+00, (5.64E−02, 3.21E+00)	1.49E+00, (5.35E−02, 3.20E+00)

approach facilitates the learning of a nonlinear transformation operator, effectively reconstructing the dynamics of the nonlinear system through linearization up to an additional output injection term in the transformed system. This transformation map is required to satisfy a system of first-order functional equations. The principal advantages of this approach lie in its robustness in reconstructing system states, given a reasonably plausible set of assumptions. Regarding practical applications, it is worth noting that such transformation maps may



(a) Greedy training, L_1 norm (b) Greedy training, L_2 norm (c) Greedy training, L_∞ norm

(d) Single training, L_1 norm (e) Single training, L_2 norm (f) Single training, L_∞ norm

Fig. 8. Benchmark problem 2. (a)–(c) Panels involve a comprehensive comparison between the performance of Physics-Informed Neural Networks (PINN) trained using two distinct approaches: one trained on the entire domain at once (Black line) and another trained using a greedy approach (blue line). The evaluation is conducted on a test set comprising 20×20 Chebyshev nodes of the second type. The reported performance metrics include the L_1, L_2, and L_∞ error norms, presented in terms of the median and the 5th to 95th percentiles. The evaluation is performed over 100 independent runs for the transformation $T_1(x_1, x_2)$. Furthermore Panels (d)–(f) depicts the same error norms with the same configuration for the transformation $T_2(x_1, x_2)$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

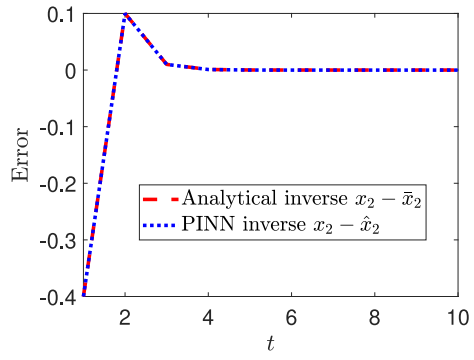


Fig. 9. Comparison between observation errors. The red line represents the difference between the analytical inverse $\bar{x}_2$ of the transformation $T_2(x_1, x_2)$ and the real state x_2, whereas the blue line depicts the difference between the true state x_2 and its approximation obtained through Newton's method, denoted as $\hat{x}_2$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exhibit abrupt gradients due to inherent singularities. Consequently, a greedy-wise training procedure has been implemented to refine our Physics-Informed Machine Learning (PINN) scheme's convergence characteristics. In this context, we employ a rudimentary zeroth-order continuation method to furnish improved initial estimations for the unknown PINN scheme parameters in proximity to singular regions. The incorporation of continuation techniques bears significant potential in simplifying computational experimentation, advancing the comprehension of these singularities, and possibly suggesting strategies for their characterization and alleviation. Moreover, it has been demonstrated that the proposed PINN scheme exhibits enhanced numerical precision capabilities, outperforming established numerical solutions such as power-series expansion.

CRediT authorship contribution statement

Hector Vargas Alvarez: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation. **Gianluca Fabiani:** Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Data curation. **Nikolaos Kazantzis:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization. **Ioannis G. Kevrekidis:** Writing – review & editing, Validation, Methodology, Conceptualization. **Constantinos Siettos:** Writing – review & editing, Writing – original draft, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The US National Science Foundation (ECCS award no. 2223987) partially supported I.G.K. The PNRR MUR, projects PE0000013-Future Artificial Intelligence Research-FAIR & CN0000013 CN HPC - National Centre for HPC, Big Data and Quantum Computing, Gruppo Nazionale Calcolo Scientifico-Istituto Nazionale di Alta Matematica (GNCS-INdAM) partially supported C.S.

References

[1] Isidori A. Nonlinear control systems. Communications and control engineering, Springer London; 1995.
[2] Chen CT. Linear system theory and design. NY: Oxford University Press; 2013.
[3] Sepulchre A, Jankovic M. Constructive nonlinear control. Communications and control engineering, Springer London; 2011.
[4] Gelb A, et al. Applied optimal estimation. MIT Press; 1974.
[5] Bernard P. Observer design for nonlinear systems. Springer International Publishing; 2019.
[6] Wu Z, Rincon D, Christofides PD. Real-time adaptive machine-learning-based predictive control of nonlinear processes. Ind Eng Chem Res 2020;59(6):2275–90.
[7] Chung ST, Grizzle JW. Sampled-data observer error linearization. Automatica 1990;26:997.
[8] Lee W, Nam K. Observer design for autonomous discrete-time nonlinear systems. Systems Control Lett 1991;17:49.
[9] Lin W, Byrnes CI. Remarks on linearization of discrete-time autonomous systems and nonlinear observer design. Systems Control Lett 1995;25:31.
[10] Krener AJ, Isidori A. Linearization by output injection and nonlinear observers. Systems Control Lett 1983;3:47.
[11] Ciccarela G, Mora MD, Germani A. Observers for discrete-time nonlinear systems. Systems Control Lett 1993;20:373.
[12] Boutayeb M, Darouach M. A reduced-order observer for nonlinear discrete-time systems. Systems Control Lett 2000;39:141.
[13] Lilge T. On observer design for nonlinear discrete-time systems. Eur J Control 1998;4:306–19.
[14] Califano C, Monaco S, Normand-Cyrot D. Canonical observer forms for multi-output systems up to coordinate and output transformations in discrete-time. Automatica 2009;45:2483–90.
[15] Kazantzis N, Kravaris C. Discrete-time observer design using functional equations. Systems Control Lett 2001;42:81–94.
[16] Xiao M, Kazantzis N, Kravaris C, Krener AJ. Nonlinear discrete-time observer design with linearizable error dynamics. IEEE Trans Autom Control 2003;48:622–6.
[17] Brivadis L, Andrieu V, Serres U. Luenberger observers for discrete-time nonlinear systems. In: 2019 IEEE 58th annual conference on decision and control. CDC, IEEE; 2019, p. 3435–40.
[18] Venkateswaran S, Kravaris C. Functional observers with linear error dynamics for discrete-time nonlinear systems. Automatica 2002;143:1–7.
[19] Nguyen DH, Widrow B. Neural networks for self-learning control systems. IEEE Control Syst Mag 1990;10(3):18–23.
[20] Hunt KJ, Sbarbaro D, Żbikowski R, Gawthrop PJ. Neural networks for control systems—a survey. Automatica 1992;28(6):1083–112.
[21] Lagaris IE, Likas A, Fotiadis DI. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans Neural Netw 1998;9(5):987–1000.
[22] Miller WT, Sutton RS, Werbos PJ. Neural networks for control. MIT Press; 1995.
[23] Chen F-C, Khalil H. Adaptive control of a class of nonlinear discrete-time systems using neural networks. IEEE Trans Autom Control 1995;40(5):791–801.
[24] Chen T, Chen H. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 1995;6(4):911–7.
[25] Taprantzis A, Siettos C, Bafas G. Fuzzy control of a fluidized bed dryer. Dry Technol 1997;15(2):511–37.
[26] Lee C, Kim J, Babcock D, Goodman R. Application of neural networks to turbulence control for drag reduction. Phys Fluids 1997;9(6):1740–7.
[27] Siettos C, Kiranoudis C, Bafas G. Advanced control strategies for fluidized bed dryers. Drying Technol 1999;17(10):2271–91.
[28] González-García R, Rico-Martínez R, Kevrekidis IG. Identification of distributed parameter systems: A neural net based approach. Comput Chem Eng 1998;22. European Symposium on Computer Aided Process Engineering-8.
[29] Kim YH, Lewis FL. Neural network output feedback control of robot manipulators. IEEE Trans Robot Autom 1999;15(2):301–9.
[30] Leu Y, Lee T, Wang W. Observer-based adaptive fuzzy-neural control for unknown nonlinear dynamical systems. IEEE Trans Syst Man Cybern B 1999;29(5):583–91.
[31] Alexandridis A, Siettos C, Sarimveis H, Boudouvis A, Bafas G. Modelling of nonlinear process dynamics using Kohonen's neural networks, fuzzy systems and Chebyshev series. Comput Chem Eng 2002;26(4–5):479–86.
[32] Siettos C, Bafas G, Boudouvis A. Truncated Chebyshev series approximation of fuzzy systems for control and nonlinear system identification. Fuzzy Sets and Systems 2002;126(1):89–104.
[33] Siettos C, Bafas G. Semiglobal stabilization of nonlinear systems using fuzzy control and singular perturbation methods. Fuzzy Sets and Systems 2002;129(3):275–94.
[34] Wu Z, Tran A, Rincon D, Christofides PD. Machine learning-based predictive control of nonlinear processes. Part I: Theory. AICHE J 2019;65:1–14.
[35] Wu Z, Tran A, Rincon D, Christofides PD. Machine-learning-based predictive control of nonlinear processes. Part II: Computational implementation. AIChE J 2019;65(11):e16734.
[36] Wu Z, Alnajd A, Gu Q, Christofides PD. Statistical machine-learning-based predictive control of uncertain nonlinear processes. AICHE J 2022;68:17642.
[37] Lombardi M, Liuzza D, di Bernardo M. Using learning to control artificial avatars in human motor coordination tasks. IEEE Trans Robot 2021;37(6):2067–82.
[38] Lombardi M, Liuzza D, Di Bernardo M. Dynamic input deep learning control of artificial avatars in a multi-agent joint motor task. Front Robot AI 2021;8:665301.
[39] Vargas-Alvarez H, Fabiani G, Kazantzis N, Siettos C, Kevrekidis IG. Discrete-time nonlinear feedback linearization via physics-informed machine learning. J Comput Phys 2023;492:12408.
[40] Patsatzis DG, Russo L, Kevrekidis IG, Siettos C. Data-driven control of agent-based models: An equation/variable-free machine learning approach. J Comput Phys 2023;111953.
[41] Alhajeri MS, Wu Z, Rincon D, Albalawi F, Christofides PD. Machine-learning-based state estimation and predictive control of nonlinear processes. Chem Eng Res Des 2021;167:268–80.
[42] Abdollahi F, Talebi H, Patel R. A stable neural network-based observer with application to flexible-joint manipulators. IEEE Trans Neural Netw 2006;17(1):118–29.
[43] Karanayil B, Rahman F, Grantham C. Stator and rotor resistance observers for induction motor drive using fuzzy logic and artificial neural networks. IEEE Trans Energy Convers 2006;20:771–80.
[44] Kazantzis N, Kravaris C. Nonlinear observer design using Lyapunov's auxiliary theorem. Systems Control Lett 1998;34(5).
[45] Ramos LdC, Di Meglio F, Morgenthaler V, da Silva LFF, Bernard P. Numerical design of Luenberger observers for nonlinear systems. In: IEEE annual conference on decision and control. CDC, 2020, p. 1–9.
[46] Janny S, Andrieu V, Wolf MN, Wolf C. Deep KKL: Data-driven output prediction for non-linear systems. In: 2021 60th IEEE conference on decision and control. CDC, 2021, p. 4376–81.
[47] Trommer L, Oksuz HY. Adaptive meta-learning-based observer design for nonlinear dynamical systems. In: 2022 European control conference. ECC, 2022.
[48] Niazi MUB, Cao J, Sun X, Das A, Johansson KH. Learning-based design of Luenberger observers for autonomous nonlinear systems. In: 2023 American control conference. ACC, 2023, p. 3048–55.
[49] Miao K, Gatsis K. Learning robust state observers using neural ODEs. In: Learning for dynamics and control conference. PMLR; 2023, p. 208–19.
[50] Peralez J, Nadri M. Deep learning-based Luenberger observer design for discrete-tine nonlinear systems. In: 2021 IEEE 69th annual conference on decision and control. CDC, 2019, p. 4370–5.
[51] Lusch B, Kutz JN, Brunton SL. Deep learning for universal linear embeddings of nonlinear dynamics. Nature Commun 2018;9:4950.
[52] Peralez J, Nadri M, Astolfi D. Neural network-based KKL observer for nonlinear discrete-time systems. In: 2022 IEEE 61st conference on decision and control. CDC, 2022, p. 2105–10.
[53] Kazantzis N. A functional equations approach to nonlinear discrete-time feedback stabilization through pole-placement. Systems Control Lett 2001;43(5):361–9.
[54] Krener AJ. Feedback linearization. In: Mathematical control theory. New York, NY: Springer New York; 1999, p. 66–98, Ch. 1.
[55] Siettos C, Kevrekidis IG, Kazantzis N. An equation-free approach to nonlinear control: Coarse feedback linearization with pole-placement. Int J Bifurcation Chaos 2006;16(07):2029–41.
[56] Raissi M, Karniadakis GE. Hidden physics models: Machine learning of nonlinear partial differential equations. J Comput Phys 2018;357:125–41.
[57] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys 2019;378:686–707.
[58] Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. Nat Rev Phys 2021;3(6):422–40.
[59] Fabiani G, Calabrò F, Russo L, Siettos C. Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines. J Sci Comput 2021;89:1–35.
[60] Calabrò F, Fabiani G, Siettos C. Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients. Comput Methods Appl Mech Engrg 2021;387:114188.
[61] De Florio M, Schiassi E, Furfaro R. Physics-informed neural networks and functional interpolation for stiff chemical kinetics. Chaos 2022;32(6).
[62] Fabiani G, Galaris E, Russo L, Siettos C. Parsimonious physics-informed random projection neural networks for initial value problems of ODEs and index-1 DAEs. Chaos 2023;33(4).
[63] Lu L, Meng X, Mao Z, Karniadakis GE. DeepXDE: A deep learning library for solving differential equations. SIAM Rev 2021;63(1):208–28.
[64] Dong S, Wang Y. A method for computing inverse parametric PDE problems with random-weight neural networks. J Comput Phys 2023;489:112263.
[65] Qian Y, Zhang Y, Huang Y, Dong S. Physics-informed neural networks for approximating dynamic (hyperbolic) PDEs of second order in time: Error analysis and algorithms. J Comput Phys 2023;495:112527.
[66] Schiassi E, De Florio M, D'Ambrosio A, Mortari D, Furfaro R. Physics-informed neural networks and functional interpolation for data-driven parameters discovery of epidemiological compartmental models. Mathematics 2021;9(17):2069.

[67] Dietrich F, Makeev A, Kevrekidis G, Evangelou N, Bertalan T, Reich S, Kevrekidis IG. Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning. Chaos 2023;33(2):023121.

[68] Kičić I, Vlachas PR, Arampatzis G, Chatzimanolakis M, Guibas L, Koumoutsakos P. Adaptive learning of effective dynamics for online modeling of complex systems. Comput Methods Appl Mech Engrg 2023;415:116204.

[69] Lee S, Psarellis YM, Siettos CI, Kevrekidis IG. Learning black-and gray-box chemotactic PDEs/closures from agent based Monte Carlo simulation data. J Math Biol 2023;87(1):15.

[70] Champenois B, Sapsis T. Machine learning framework for the real-time reconstruction of regional 4D ocean temperature fields from historical reanalysis data and real-time satellite and buoy surface measurements. Physica D 2024;459:134026.

[71] Kovachki N, Li Z, Liu B, Azizzadenesheli K, Bhattacharya K, Stuart A, Anandkumar A. Neural operator: Learning maps between function spaces with applications to pdes. J Mach Learn Res 2023;24(89):1–97.

[72] Karnakov P, Litvinov S, Koumoutsakos P. Solving inverse problems in physics by optimizing a discrete loss: Fast and accurate learning without neural networks. PNAS Nexus 2024;3(1):page005.

[73] Daryakenari NA, De Florio M, Shukla K, Karniadakis GE. AI-Aristotle: A physics-informed framework for systems biology gray-box identification. PLoS Comput Biol 2024;20(3).

[74] Fabiani G, Evangelou N, Cui T, Bello-Rivas JM, Martin-Linares CP, Siettos C, Kevrekidis IG. Task-oriented machine learning surrogates for tipping points of agent-based models. Nature Commun 2024;15(4117):1–13.

[75] Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nat Mach Intell 2021;3(3):218–29.

[76] Tipireddy R, Perdikaris P, Stinis P, Tartakovsky AM. Multistep and continuous physics-informed neural network methods for learning governing equations and constitutive relations. J Mach Learn Model Comput 2022;3(2).

[77] Patsatzis DG, Fabiani G, Russo L, Siettos C. Slow invariant manifolds of singularly perturbed systems via physics-informed machine learning. SIAM J Sci Comput 2023;46(4):C297–322.

[78] Luenberger D. Observing the state of a linear system. IEEE Trans Mil Electron 1963;8(2):74–80.