Computers and Chemical Engineering 192 (2025) 108899

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Physics-informed neural networks for dynamic process operations with limited physical knowledge and data

Mehmet Velioglu [a,b], Song Zhai [e], Sophia Rupprecht [a,f], Alexander Mitsos [c,a,d], Andreas Jupke [e], Manuel Dahmen [a,*]

[a] Institute of Climate and Energy Systems, Energy Systems Engineering (ICE-1), Forschungszentrum Jülich GmbH, Jülich 52425, Germany
[b] RWTH Aachen University, Aachen 52062, Germany
[c] JARA-ENERGY, Jülich 52425, Germany
[d] RWTH Aachen University, Process Systems Engineering (AVT.SVT), Aachen, 52074, Germany
[e] RWTH Aachen University, Fluid Process Engineering (AVT.FVT), Aachen 52074, Germany
[f] Delft University of Technology, 2629 HZ, Delft, The Netherlands

ARTICLE INFO

Keywords:
Physics-informed neural networks
Chemical engineering
Dynamic process modeling
State estimation
Van de Vusse reaction
Liquid–liquid separator

ABSTRACT

In chemical engineering, process data are expensive to acquire, and complex phenomena are difficult to fully model. We explore the use of physics-informed neural networks (PINNs) for modeling dynamic processes with incomplete mechanistic semi-explicit differential–algebraic equation systems and scarce process data. In particular, we focus on estimating states for which neither direct observational data nor constitutive equations are available. We propose an easy-to-apply heuristic to assess whether estimation of such states may be possible. As numerical examples, we consider a continuously stirred tank reactor and a liquid–liquid separator. We find that PINNs can infer immeasurable states with reasonable accuracy, even if respective constitutive equations are unknown. We thus show that PINNs are capable of modeling processes when relatively few experimental data and only partially known mechanistic descriptions are available, and conclude that they constitute a promising avenue that warrants further investigation.

1. Introduction

Dynamic operation and control of chemical and biotechnological processes are essential for efficient and sustainable production. Mathematical models describing the behavior of such processes are often classified concerning their degree of reliance on physical/chemical knowledge or data into three categories: (1) white-box or first-principle or mechanistic models, (2) black-box or data-driven models, and (3) gray-box or hybrid models (Zendehboudi et al., 2018; Marquardt, 1996).

Black-box modeling relies on (measurement) data to establish a predictive relation between process inputs and outputs, thus avoiding the need for a mechanistic process description. In recent years, approaches involving deep neural networks (DNNs) have become particularly prominent data-driven models for process operations. DNNs can model nonlinear dependencies between multiple inputs and outputs (Goodfellow et al., 2016) but require extensive training data and often fail to make physically consistent predictions in scientific or engineering applications (Zendehboudi et al., 2018). In contrast, mechanistic process models are based on the governing physical and chemical laws of a system and suitable constitutive equations and comprise relatively few parameters that need to be estimated from data (von Stosch et al., 2014). They typically allow for physically consistent predictions. However, in chemical and biotechnological processes, complex phenomena such as reaction kinetics, coalescence, or sedimentation often lack a rigorous mathematical description, hindering the mechanistic modeling of such processes (Kahrs and Marquardt, 2008). Hybrid modeling combines mechanistic and data-driven modeling and aims to take advantage of the respective strengths and mitigate the respective weaknesses of the two approaches. Compared to purely data-driven models, suitably-designed hybrid models require less training data, make physically more consistent predictions, and (thus) extrapolate to a higher extent (Kahrs and Marquardt, 2007).

Hybrid models have been used extensively to model dynamic process operation problems if complete system knowledge is unavailable (Roffel and Betlem, 2006) and thus have become a crucial modeling tool for numerous tasks related to chemical process control (Asprion et al., 2019). Various types of hybrid model structures have been proposed over the years in the process systems engineering (PSE) community, with the sequential approach and the parallel approach being

* Corresponding author.
E-mail address: m.dahmen@fz-juelich.de (M. Dahmen).

https://doi.org/10.1016/j.compchemeng.2024.108899
Received 7 July 2024; Received in revised form 30 September 2024; Accepted 19 October 2024
Available online 30 October 2024

the most prominent structures. For instance, Psichogios and Ungar (1992) studied incorporating an artificial neural network to predict states lacking a constitutive description inside an otherwise mechanistic model for a fed-batch bioreactor (sequential approach). Su et al. (1992) proposed to correct the mismatch between a white-box model and process data from a polymer reaction system by a neural network (parallel approach). The parallel approach can also be combined with the sequential approach, i.e., a second mechanistic model is added after the parallel hybrid model to enforce physically consistent predictions, see, e.g., Thompson and Kramer (1994). Recently, the popularity of hybrid modeling in chemical engineering has been increasing again due to advancements in machine learning and the rise of digital twins in smart manufacturing (Yang et al., 2020). Some notable contemporary works on hybrid modeling are dedicated to the estimation of (spatio-)temporally varying parameters, which is related to the estimation of states with missing constitutive equations, the main topic of our article. Specifically, Shah et al. (2022) estimate time-varying parameters in fermentation processes, Pahari et al. (2024) estimate spatio-temporally varying diffusivity in a reaction–diffusion model, and Sitapure and Sang-Il Kwon (2023) estimate kinetic parameters in a batch crystallization process with a transformer architecture. For further applications of hybrid modeling in chemical engineering, we refer the reader to review papers by Sansana et al. (2021), Yang et al. (2020), Sharma and Liu (2022) and Schweidtmann et al. (2021).

Physics-based regularization of DNNs gives rise to so-called physics-informed neural networks (PINNs), which have some similarities to hybrid models but are better regarded as a special variant of a data-driven model that is trained with available physical laws as constraints (Bradley et al., 2022). Specifically, in a PINN, the DNN acts as the sole prediction model, but it is informed about governing physical laws during training through additional terms in the loss function (Nabian and Meidani, 2019; Karniadakis et al., 2021). In contrast, hybrid models have distinct mechanistic and data-driven sub-models which jointly produce a prediction (Bradley et al., 2022; Schweidtmann et al., 2024).

The origins of physics-based regularization date back to (at least) the works of Lagaris et al. (1998) on solving ordinary and partial differential equations using neural networks (NNs) as universal function approximators. This approach was originally not taken up widely, likely due to the general limitations of NN training at that time. However, Raissi et al. (2019) recently revisited the physics-based regularization approach using modern algorithms and tools for training and introduced the term PINN.

The original PINN architectures (Raissi et al., 2019; Nascimento et al., 2020) did not account for varying initial/boundary conditions or control inputs. However, Antonelo et al. (2021) showed that adding control inputs and initial conditions to the NN makes the PINN approach suitable for control applications. Another application of PINNs for control purposes was proposed by Arnold and King (2021), who pursued a state-space modeling approach based on PINNs, including initial conditions as inputs to the NN. However, separate networks are trained for each discretized control actuation instead of adding control inputs to the network.

Recently, PINNs have also seen a surge in chemical engineering applications, mainly in the form of physics-informed *recurrent* neural networks (Zheng et al., 2023). For instance, they have been applied in conjunction with model predictive control (MPC) to a continuously stirred tank reactor (CSTR) (Zheng et al., 2023) and a batch crystallization process (Wu et al., 2023), to control systems with noisy data (Alhajeri et al., 2022) and parametric uncertainty (Zheng and Wu, 2023), and to fluid flow problems, most notably flow field prediction in cyclone separators (Queiroz et al., 2021) and a Van de Vusse CSTR (Choi et al., 2022). Ji et al. (2021) developed PINNs that can address stiff chemical kinetic problems.

While studies have shown that PINNs are promising model candidates for chemical engineering applications, open questions remain about their utility for state estimation. In general, state estimation is concerned with estimating the state of a given process utilizing measurement data and a mathematical process model (Barfoot, 2017; Gelb et al., 1974). State estimation is often performed with filtering techniques, e.g., the Kalman filter (Kalman, 1960a), which have recently also been combined with PINNs, see, e.g., Tan et al. (2023) and Arnold and King (2021). PINNs have also been used to estimate unmeasured states directly, i.e., without the use of a state estimation technique. For instance, Raissi et al. (2020) estimated velocity and pressure fields from the concentration data of a passive scalar from flow field visualizations, using Navier Stokes equations as the physics knowledge. Recently, Wu et al. (2023) showed that PINNs with partial physics knowledge can estimate immeasurable states in a batch crystallization process by using the known governing equations of these states. The question, however, remains whether PINNs can estimate states for which neither direct observational data nor constitutive equations are available.

In the present work, we thus set out to answer the following two questions: (i) Can PINNs estimate immeasurable process states for which constitutive equations are not known? (ii) Under which conditions can we expect this to work? To this end, we will first conceptualize PINN-based dynamic process models in a setting of partially known mechanistic equations as well as measured and unmeasured process states. Specifically, we consider systems that (i) can be described by differential–algebraic equations (DAEs) in principle, (ii) for which only partial mechanistic knowledge in the form of some known equations is available, and (iii) for which process data for some states is available. Regarding the PINN modeling, we follow the standard approach, as it was first introduced by Raissi et al. (2019), but with the extensions to initial states and control inputs by Antonelo et al. (2021). We propose the use of an incidence matrix as an easy-to-apply heuristic to *a priori* evaluate whether estimation of unmeasured states with a PINN may be possible. We then perform extensive numerical studies by using two fully-known mechanistic models to emulate situations where some, but not full, mechanistic knowledge is available for modeling purposes. Specifically, we study a CSTR model with Van de Vusse reaction from the literature (van de Vusse, 1964) and a liquid–liquid separator for which we develop a model by extending the model from Backi et al. (2018, 2019). We follow an in-silico approach to generate process data, i.e., we use the full-order mechanistic model, which in a real situation would not be available, to generate synthetic observational data. Controlling the amount and diversity of training data allows us to run extensive numerical experiments on the fitting and generalization capabilities of PINNs as well as vanilla neural network benchmark models, i.e., multilayer perceptrons. Following the taxonomy of process quantities and model equations by Marquardt (1996), we distinguish balance equations and constitutive equations and emulate situations with different degrees of mechanistic knowledge available for PINN model development.

The paper is structured as follows: Section 2 presents the proposed approach for PINN-based dynamic process modeling with incomplete physical knowledge, and our heuristic for assessing the state estimation capabilities of a PINN. Section 3 provides numerical examples and results for the CSTR, focusing on the physics-informed part of the PINN by varying the amount of physical knowledge provided. Section 4 provides numerical examples and results for the liquid–liquid separator, focusing on the data-driven part of the PINN by varying the number of measured properties provided as NN inputs. In all examples, the empirical findings are related to the results from the heuristic. Section 5 discusses the conclusion and future work.

2. Methods

2.1. Preliminaries

Raissi et al. (2019) introduced PINNs to find data-driven solutions to partial differential equations (PDEs) utilizing DNNs. In their approach,

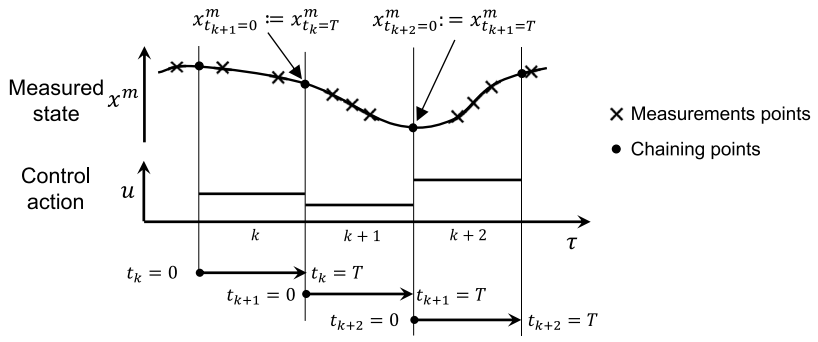


Fig. 1. Relationship between PINN time t and process time τ: The PINN time domain $[0, T]$ corresponds to the length of a step-wise constant control input. In general, PINN time t differs from process time τ and chaining of model predictions is required to simulate longer periods of time. Only if the control input is constant over the entire process duration, t and τ coincide. Measurements can come from an irregular grid.

they employ the NN to approximate the solution of a PDE problem. The inputs to the DNN are the spatio-temporal coordinates, and the DNN outputs are the states of the dynamic system. The DNN is trained in a semi-supervised manner, e.g., with small amounts of labeled data, i.e., process data with corresponding input/output relations, and large amounts of unlabeled data, i.e., collocation points in time and space where residuals of governing equations, i.e., the PDEs, are computed. These residuals constitute a loss term that penalizes the deviations of the DNN outputs from the governing equations. Thus, PINNs can learn to obey the physical laws of the system.

In their original form, PINNs do not account for control variables. The extension to control applications is, however, straightforward: Antonelo et al. (2021) added the control variable(s) and initial states as NN inputs. Considering initial states as network inputs means that the PINN model can be trained for various samples of initial states and control variables, facilitating extensive coverage of the state and control action spaces. The time domain of the PINN can be chosen according to the needs of the control scheme, e.g., in MPC applications, step-wise constant control inputs are often used. Thus, if the PINN time domain $[0, T]$ corresponds to the length of a step-wise constant control input, the control inputs from the perspective of the NN are not functions of time but constants. It is therefore, in general, necessary to distinguish PINN time t from process time τ and to chain the PINN predictions in order to simulate longer periods involving changing control inputs (cf. Fig. 1). Note that in the numerical examples in Sections 3 and 4, for the sake of a simple implementation, we study varying control inputs which are however kept constant throughout the entire process duration, thus implying $t = \tau$. For further details on including control actions into PINNs, we refer the reader to Antonelo et al. (2021) for integrating PINNs into MPC.

2.2. PINN-based dynamic process modeling with partial physical knowledge

We consider the scenario where a partial mechanistic process model is available that can be used for physics-based regularization of a NN. We assume that this partial process model comes in the form of a semi-explicit differential–algebraic equation (DAE) system (Brenan et al., 1996):

$$\dot{\mathbf{x}}(t) = \boldsymbol{f}(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}), \tag{1a}$$

$$\mathbf{0} = \boldsymbol{g}(\mathbf{x}(t), \mathbf{y}(t), \mathbf{u}) \tag{1b}$$

Here, $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the differential states vector, $\mathbf{y}(t) \in \mathbb{R}^{n_y}$ is the algebraic states vector, and $\mathbf{u} \in \mathbb{R}^{n_u}$ is the control inputs vector. The dot symbol $\dot{(\cdot)}$ denotes a time derivative. $\boldsymbol{f}$ denotes the right-hand side (RHS) of the ordinary differential Eq. (1a), and $\boldsymbol{g}$ is the RHS of the algebraic Eq. (1b).

In a practical setting, some states might be impossible to measure (immeasurable), e.g., reaction rate constants, or some states might be impractical/expensive to measure, e.g., concentrations. The term *unmeasured states* covers both of these types and will be used throughout this work. We aim to estimate unmeasured process states with the available partial mechanistic knowledge and measurement data on other measured states. To this end, we sub-categorize the differential and algebraic states into measured and unmeasured states, using superscripts m and u, respectively. This is a special case of the more general output equations used in observability analysis and control, see, e.g., Lee and Markus (1967).

To predict the measured states $\mathbf{x}^m(t) \in \mathbb{R}^{n_{x^m}}$, $\mathbf{y}^m(t) \in \mathbb{R}^{n_{y^m}}$ and to estimate the unmeasured states $\mathbf{x}^u(t) \in \mathbb{R}^{n_{x^u}}$, $\mathbf{y}^u(t) \in \mathbb{R}^{n_{y^u}}$, we use the neural network $\mathbf{NN}_{\mathbf{w},\mathbf{b}}$ with weights $\mathbf{w}$ and biases $\mathbf{b}$, i.e., $[\hat{\mathbf{x}}(t), \hat{\mathbf{y}}(t)] = \mathbf{NN}_{\mathbf{w},\mathbf{b}}(t, \mathbf{x}^m(t_0), \mathbf{u})$, where $\hat{\mathbf{x}}(t)$ and $\hat{\mathbf{y}}(t)$ denote the NN predictions of the differential and algebraic states, respectively. The network inputs are the time t, the initial values of the measured differential states $\mathbf{x}^m(t_0)$, and the control inputs $\mathbf{u}$. The NN parameters $\mathbf{w}$ and $\mathbf{b}$ can be learned by minimizing the mean squared error loss, similar to Raissi et al. (2019) and Antonelo et al. (2021):

$$MSE_{total} = MSE_{data} + \lambda_1 MSE_{physics} + \lambda_2 MSE_{init}, \tag{2a}$$

$$MSE_{data} = \frac{1}{n_{x^m} N_d} \sum_{j=1}^{N_d} (\hat{\mathbf{x}}^m(t_j) - \mathbf{x}^m(t_j))^2 + \frac{1}{n_{y^m} N_d} \sum_{j=1}^{N_d} (\hat{\mathbf{y}}^m(t_j) - \mathbf{y}^m(t_j))^2, \tag{2b}$$

$$MSE_{physics} = \frac{1}{n_x N_e} \sum_{j=1}^{N_e} \left(\dot{\hat{\mathbf{x}}}(t_j) - \boldsymbol{f}(\hat{\mathbf{x}}(t_j), \hat{\mathbf{y}}(t_j), \mathbf{u}_j)\right)^2 + \frac{\lambda_g}{n_y N_e} \sum_{j=1}^{N_e} \left(\boldsymbol{g}(\hat{\mathbf{x}}(t_j), \hat{\mathbf{y}}(t_j), \mathbf{u}_j)\right)^2, \tag{2c}$$

$$MSE_{init} = \frac{1}{n_{x^m} N_i} \sum_{j=1}^{N_i} (\hat{\mathbf{x}}^m_j(t_0) - \mathbf{x}^m_j(t_0))^2 \tag{2d}$$

Here, MSE_{data} corresponds to the loss term accounting for the measurement data, $MSE_{physics}$ corresponds to the loss term that is computed with the available physics knowledge (Eqs. (1a) and (1b)), and MSE_{init} corresponds to a loss term that describes the mismatch between the NN predictions at $t = t_0$ and the initial values $\mathbf{x}^m_j(t_0)$. N denotes the number of data points. Note that the subscript j refers to finitely many samples taken at times t_j, with corresponding initial values $\mathbf{x}^m_j(t_0)$ and control actions $\mathbf{u}_j$. We omit the latter two from the notation for simplicity. The subscripts d, e, and i correspond to data points associated with MSE_{data}, $MSE_{physics}$, and MSE_{init}, respectively.

λ_1 and λ_2 denote the weights of the physics and initial condition loss terms, respectively, and λ_g establishes a weighting between the algebraic and the differential equations in the physics loss term.

Note that for the calculation of $MSE_{physics}$ and MSE_{init} no measurement data are needed. For $MSE_{physics}$, we calculate the physics residuals using Eqs. (1a) and (1b) at randomly sampled time points

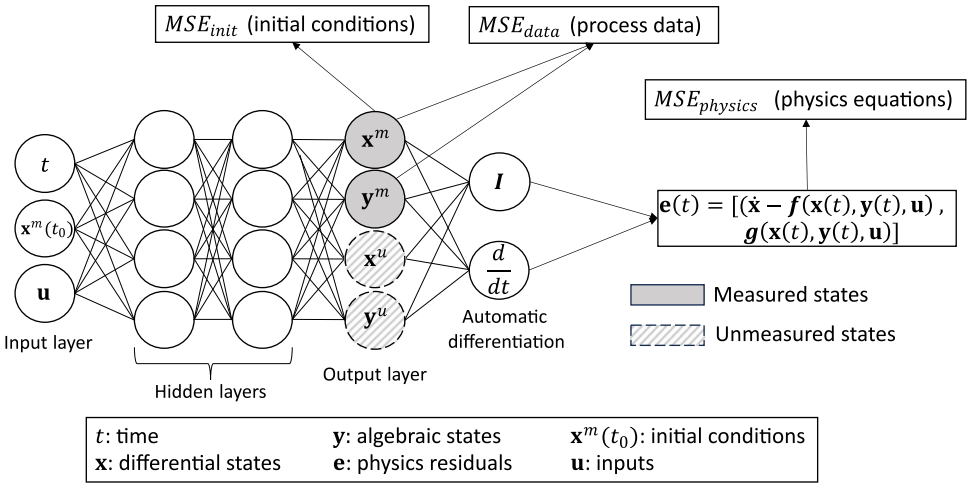


Fig. 2. PINN-based dynamic process model with semi-explicit DAE physics model.

$t = t_j$. For MSE_{init}, we train the NN predictions $\hat{\mathbf{x}}^m(t = t_0)$ to comply with the initial values $\mathbf{x}^m(t_0)$, again for randomly sampled values in a given range. A general network schematic of the PINN is given in Fig. 2.

2.3. Heuristic for assessing PINN state estimation capabilities

We propose a heuristic to a priori assess whether a PINN may be capable of estimating unmeasured process states by drawing inspirations from DAE solvability analysis, see, e.g., Brenan et al. (1996). Our conjecture is that the PINN can leverage training data, i.e., samples for $\mathbf{x}_m(t_j)$ and $\mathbf{y}_m(t_j)$, to "solve" the known Eqs. (1a) and (1b) for the unknown states $\mathbf{x}_u(t_j)$ and $\mathbf{y}_u(t_j)$ at a point t_j. Specifically, our heuristic mimics structural index analysis by means of an incidence matrix (Duff and Gear, 1986; Gani and Cameron, 1992; Unger et al., 1995). In our PINN incidence matrix, the rows represent the RHSs of the known physics equations, i.e., $\boldsymbol{f}$ and $\boldsymbol{g}$ (see Eqs. (1a) and (1b)), and the columns represent the unmeasured process states $\mathbf{x}_u$ and $\mathbf{y}_u$. Each occurrence of an unmeasured state in $\boldsymbol{f}$ and $\boldsymbol{g}$ is indicated by drawing a cross (×) in the corresponding entry of the matrix. Note that the PINN uses AD to compute $\dot{\hat{\mathbf{x}}}$, i.e., the derivative of the NN outputs $\hat{\mathbf{x}}(t)$ with respect to the NN input t. Moreover, the NN learns to assemble state trajectories from the data provided at distinct time points t_j, and thus, it implicitly learns time-derivatives of the states. Consequently, we do not consider $\dot{\mathbf{x}}$, i.e., the left-hand side (LHS) of Eqs. (1a), as unknowns but restrict our analysis to the RHSs $\boldsymbol{f}$ and $\boldsymbol{g}$ where no time-derivatives appear (see Eqs. (1a) and (1b)). This implies that we do not consider $\dot{x}_j$ as an occurrence of x_j when we assemble the incidence matrix.

We conjecture that the incidence matrix having a full-column rank, i.e., if exactly one cross in each column can be marked with a circle without marking more than one cross in a single row, constitutes an indicator for possible state estimation. A simple example of an incidence matrix for a PINN is given in Table 1. Note that an incidence matrix having more equations than unmeasured states, i.e., more rows than columns, is not a concern in itself. In fact, each additional equation may provide additional regularization to the NN and thus may be regarded as beneficial. We stress that the incidence matrix is a heuristic, i.e., it represents neither a necessary nor a sufficient condition for state estimation with a PINN (see Sections SM5 and SM6 of the Supplementary Materials), and thus, it can give wrong results. Note that for fully-specified dynamic systems, necessary and sufficient criteria for observability analysis exist, see, e.g., Lee and Markus (1967) and Kou et al. (1973), based on trajectory information. Since we have an incomplete physics model, we instead construct the heuristic with a point-wise analysis, similar to the solvability analysis of equation systems (Brenan et al., 1996; Duff and Gear, 1986; Gani and Cameron, 1992; Unger et al., 1995). The practical construction and interpretation of the incidence matrix are demonstrated extensively in Sections 3 and 4.

2.4. Vanilla NN benchmark models

To compare the predictions of a PINN model with a purely data-driven benchmark, we choose a feed-forward artificial neural network (ANN), as ANNs are widely used and can have a similar network architecture as the PINN model, thus allowing us to study the effects of the physics-based regularization. To make the comparison as meaningful as possible, we use the same hyperparameters and training scheme for the PINN model and the vanilla ANN model. Still, the network architecture for the vanilla ANN is slightly different from that of the PINN in the sense that only the measured states can be network outputs, as no process data is available for the unmeasured states. We use the following loss function to train the vanilla ANN, omitting the physics-based regularization term in Eq. (2a) but keeping the loss term for the initial conditions:

$$MSE = MSE_{data} + \lambda_1 MSE_{init},$$

$$MSE_{data} = \frac{1}{n_{x^m} N_d} \sum_{j=1}^{N_d} (\hat{\mathbf{x}}^m(t_j) - \mathbf{x}^m(t_j))^2 + \frac{1}{n_{y^m} N_d} \sum_{j=1}^{N_d} (\hat{\mathbf{y}}^m(t_j) - \mathbf{y}^m(t_j))^2,$$

$$MSE_{init} = \frac{1}{n_{x^m} N_i} \sum_{j=1}^{N_i} (\hat{\mathbf{x}}_j^m(t_0) - \mathbf{x}_j^m(t_0))^2$$

3. Numerical example 1: Van de Vusse reactor

We use the Van de Vusse (van de Vusse, 1964) CSTR, a common benchmark problem in the literature on nonlinear control applications (Chen et al., 1995), to investigate generalization, state estimation, and extrapolation capabilities of the PINN models under varying amounts of physical knowledge provided through physics equations. Thus, we focus on the physics regularization aspect of the PINN.

The van de Vusse reaction scheme reads:

$$\mathrm{A} \xrightarrow{k_1} \mathrm{B} \xrightarrow{k_2} \mathrm{C},$$

$$2\,\mathrm{A} \xrightarrow{k_3} \mathrm{D}.$$

Substance A is fed to the reactor with concentration $c_{A,in}$ and temperature T_{in}. Substance B is the desired product, whereas substances C and D are unwanted byproducts. Heat is removed from the cooling jacket fluid with rate $\dot{Q}_K$ by an external heat exchanger. The schematic of the CSTR is given in Fig. 3. The dynamics of the reactor are given by the following nonlinear equations derived from component balances for

Table 1
Incidence matrix for a PINN with a semi-explicit DAE physics model: Measurement data for training is available for x_1^m only. The unmeasured states x_2^u and y^u shall be estimated from the data on x_1^m. The cross (×) denotes the occurrence of an unmeasured state in a physics equation. The incidence matrix has full-column rank, as it is possible to mark exactly one cross in each column without marking more than one cross in a single row.

Known physics model (semi-explicit DAE)

(a): $\dot{x}_1^m = x_1^m + x_2^u$
(b): $\dot{x}_2^u = 3x_1^m$
(c): $0 = x_1^m x_2^u + y^u$

Incidence matrix

$[\boldsymbol{f}, \boldsymbol{g}] \downarrow$ $[\mathbf{x}^u, \mathbf{y}^u] \rightarrow$	x_2^u	y^u
(a)	⊗	
(b)		
(c)	×	⊗

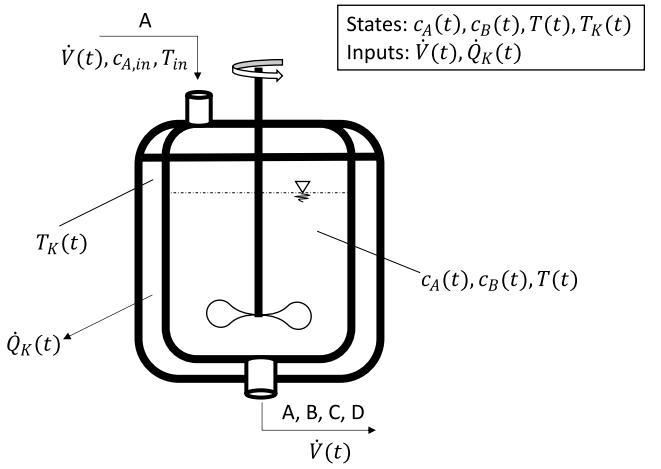


Fig. 3. Schematic representation of the van de Vusse CSTR.

Table 2
Parameters for the van de Vusse CSTR.
Source: Taken from Chen et al. (1995).

Parameter	Symbol	Value
inlet molar flow rate of substance A	$c_{A,\text{in}}$	5.10 mol/L
inlet temperature	T_{in}	378.1 K
collision factor for reaction 1	k_{10}	1.287×10^{12} 1/h
collision factor for reaction 2	k_{20}	1.287×10^{12} 1/h
collision factor for reaction 3	k_{30}	9.043×10^{9} L/(mol h)
activation energy for reaction 1	$E_{a,1}$	−9758.3 K
activation energy for reaction 2	$E_{a,2}$	−9758.3 K
activation energy for reaction 3	$E_{a,3}$	−8560 K
enthalpy of reaction 1	ΔH_{AB}	4.2 kJ/mol$_A$
enthalpy of reaction 2	ΔH_{BC}	−11.0 kJ/mol$_B$
enthalpy of reaction 3	ΔH_{AD}	−41.85 kJ/mol$_A$
density	ρ	0.9342 kg/L
heat capacity	C_P	3.01 kJ/(kg K)
heat capacity of coolant	C_{PK}	2.00 kJ/(kg K)
heat transfer coefficient of cooling jacket	k_w	4032 kJ/(h m^2 K)
surface area of cooling jacket	A_R	0.215 m^2
reactor volume	V_R	0.01 m^3
coolant mass	m_K	5.0 kg

substances A and B and energy balances for the reactor and the cooling jacket (Chen et al., 1995):

$$\dot{c}_A(t) = \frac{\dot{V}(t)}{V_R}(c_{A,in} - c_A(t)) - k_1(T)c_A(t) - k_3(T)c_A(t)^2, \tag{3a}$$

$$\dot{c}_B(t) = -\frac{\dot{V}(t)}{V_R}c_B(t) + k_1(T)c_A(t) - k_2(T)c_B(t), \tag{3b}$$

$$\dot{T}(t) = \frac{\dot{V}(t)}{V_R}(T_{in} - T(t)) - \frac{1}{\rho C_p}[k_1(T)c_A(t)\Delta H_{AB} + k_2(T)c_B(t)\Delta H_{BC} + k_3(T)c_A(t)^2\Delta H_{AD}] + \frac{k_w A_R}{\rho C_p V_R}(T_K(t) - T(t)), \tag{3c}$$

$$\dot{T}_K(t) = \frac{1}{m_K C_{pK}}[\dot{Q}_K(t) + k_w A_R(T(t) - T_K(t))] \tag{3d}$$

Here, $c_A(t)$ and $c_B(t)$ denote the concentrations of substances A and B, $T(t)$ is the reactor temperature, and $T_K(t)$ is the cooling jacket temperature, assumed to be uniform in space. The aforementioned quantities correspond to the differential states $\mathbf{x}$ of the Van de Vusse CSTR, i.e., $\mathbf{x} = [c_A, c_B, T, T_K]^T$. The flow rate $\dot{V}(t)$, and the heat transfer rate by the coolant $\dot{Q}_K(t)$ (heat removal) are the manipulated variables. Note that the dot notation in $\dot{V}(t)$ and $\dot{Q}_K(t)$ indicates flow rates (as opposed to time derivatives). The reaction rate constants $k_i(T)$ correspond to the algebraic states $\mathbf{y}$ and are calculated using the Arrhenius equation:

$$k_i(T) = k_{i0}\exp(\frac{E_{a,i}}{T}), \qquad i = 1, 2, 3 \tag{4}$$

All parameters listed in Eqs. (3) and (4) are given in Table 2.

During our preliminary tests, we observed that having values in a similar order of magnitude for the different PINN inputs and outputs improves the training stability and performance. However, when normalizing the outputs, the PINN physics equations must be scaled accordingly. Thus, we decided to make the time, states, and manipulated variables dimensionless and use dimensionless equations to calculate the physics loss. We give the dimensionless variables and equations in the Supplementary Materials. In addition, we normalize the PINN inputs, i.e., we scale the input features to values between −1 and 1.

To investigate the effects of varying physical knowledge, we create three different PINN models with increasing physics knowledge. Moreover, we investigate the performance of a vanilla ANN model to facilitate a comparison between the PINN model and a purely data-driven model. We list these models, the physics equations, the knowledge supplied to the PINN model, and the measured and unmeasured states in Table 3. Moreover, we give the network schematic of each model in the Supplementary Materials.

3.1. Data set generation, training, and hyperparameter selection

We assume the operating ranges presented in Table 4, with a selected time interval for step-wise control changes of $T = 60$ s, i.e., $t \in [0, 60]$ s. Data generation to calculate the physics loss term $MSE_{physics}$ and the initial condition loss term MSE_{init} in Eqs. (2) are done by selecting $N_e = 10{,}000$ collocation and $N_i = 100$ initial value points. This selection is done using Latin Hypercube sampling (Iman et al., 1981).

For the process data generation, we use the explicit Runge–Kutta method of order 5, utilizing *solve_ivp* solver from *scipy.integrate* module in Python (Virtanen et al., 2020; Dormand and Prince, 1980). We solve the full-order process model (Eqs. (3) and (4)) for time $t \in [0, 60]$ s with random inputs for $c_A(t_0)$, $c_B(t_0)$, $T(t_0)$, $T_K(t_0)$, $\frac{\dot{V}}{V_R}$, $\dot{Q}_K$ in the given ranges and keeping the manipulated variables $\frac{\dot{V}}{V_R}$ and $\dot{Q}_K$ constant throughout the investigated process duration of 60 s, with relative and absolute error of 1×10^{-13} and 1×10^{-16} respectively. We output each process trajectory on an equidistant time grid with step-size $\Delta t = 0.6$ s. Note that $\Delta t = 0.6$ s pertains to the granularity of the training/testing trajectories; the PINN at the prediction phase can make one-shot predictions for any time $t \in [0, 60]$ s. Moreover, the training/testing trajectories could also be obtained from an irregular,

Table 3
Physical knowledge and output configuration (measured and unmeasured process states) for the Van de Vusse CSTR PINN models. In PINN-C, T and T_K can also be unmeasured depending on the case study (cf. Section 3.4). The time dependence of the states is not shown explicitly for brevity. The manipulated variables $\frac{\dot{V}}{V_R}$ and $\dot{Q}_K$ are the step-wise constant controls which we, for the sake of a simple implementation, keep constant throughout the investigated process duration.

Model name	Physics knowledge	Physics equations	Measured process states	Unmeasured process states
Vanilla ANN	None	None	c_A, c_B, T, T_K	None
PINN-A	Mole balances with net reaction rates	$\dot{c}_A = \frac{\dot{V}}{V_R}(c_{A,in} - c_A) + r_A$ $\dot{c}_B = -\frac{\dot{V}}{V_R}c_B + r_B$	c_A, c_B, T, T_K	r_A, r_B
PINN-B	Mole and energy balances with individual reaction rates	$\dot{c}_A = \frac{\dot{V}}{V_R}(c_{A,in} - c_A) - r_1 - r_3$ $\dot{c}_B = -\frac{\dot{V}}{V_R}c_B + r_1 - r_2$ $\dot{T} = \frac{\dot{V}}{V_R}(T_{in} - T) + \frac{k_w A_R}{\rho C_p V_R}(T_K - T) - \frac{1}{\rho C_p}(r_1 \Delta H_{AB} + r_2 \Delta H_{BC} + r_3 \Delta H_{AD})$ $\dot{T}_K = \frac{1}{m_K C_{pK}}(\dot{Q}_K + k_w A_R (T - T_K))$	c_A, c_B, T, T_K	r_1, r_2, r_3
PINN-C	Mole and energy balances with reaction rate expressions (without Arrhenius' law)	$\dot{c}_A = \frac{\dot{V}}{V_R}(c_{A,in} - c_A) - k_1 c_A - k_3 c_A^2$ $\dot{c}_B = -\frac{\dot{V}}{V_R}c_B + k_1 c_A - k_2 c_B$ $\dot{T} = \frac{\dot{V}}{V_R}(T_{in} - T) + \frac{k_w A_R}{\rho C_p V_R}(T_K - T) - \frac{1}{\rho C_p}(k_1 c_A \Delta H_{AB} + k_2 c_B \Delta H_{BC} + k_3 c_A^2 \Delta H_{AD})$ $\dot{T}_K = \frac{1}{m_K C_{pK}}(\dot{Q}_K + k_w A_R (T - T_K))$	c_A, c_B, T, T_K	k_1, k_2, k_3

Table 4
Operating ranges for states and inputs in the Van de Vusse CSTR example. The lower bound is denoted by lb, and the upper bound is denoted by ub. These values are chosen to remain in the vicinity of a steady state. Extreme values refer to the minimum and maximum values appearing in a generated trajectory.

Variable	Unit	Initial value		Extreme value	
		lb	ub	min	max
c_A	mol/L	2.14	2.57	1.74	2.74
c_B	mol/L	0.87	1.09	0.87	1.28
T	K	387	403	385	403
T_K	K	371	386	371	395
$\frac{\dot{V}}{V_R}$	(1/h)	5	28.4	5	28.4
$\dot{Q}_K$	kJ/h	−2227	0	−2227	0

i.e., non-equidistant, time grid. We create $N_{total} = 100$ trajectories from which we select $N_{test} = 20$ trajectories for testing. For training, we use N_{train} trajectories, each one having $N_m = 101$ data points. The total number of measurement points is thus $N_d = N_{train} N_m$. Specifically, we create two training sets from the 80 trajectories that are not used for the testing: First, we create a training set representing a *low-data regime* consisting of only $N_{train} = 20$ training trajectories. Second, we create a training set representing a *high-data regime* consisting of $N_{train} = 80$ training trajectories.

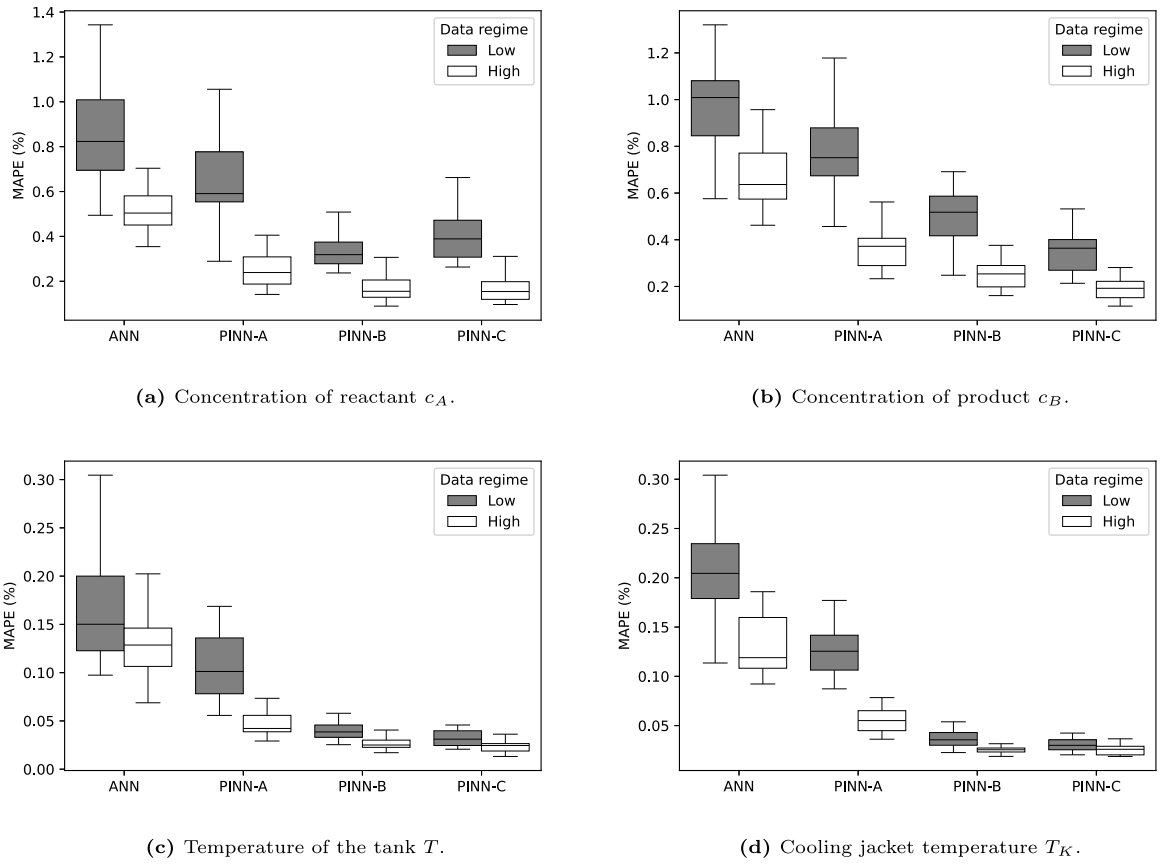
For the training, we use a hybrid strategy; we first start with the Adam optimizer (Kingma and Ba, 2017) and then switch to the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm (Liu and Nocedal, 1989). L-BFGS typically provides more accurate results for PINNs (Markidis, 2021); however, it tends to get stuck in a local minimum if used directly (Markidis, 2021). Thus, Adam is first used to avoid local minima, and then L-BFGS is used for fine-tuning following the approach presented by Markidis (2021) and Jin et al. (2021) since we could confirm their observation during our preliminary studies. We use a dynamic weighting scheme to decide on the weights λ_i in Eq. (2), called inverse Dirichlet weighting (IDW) (Maddu et al., 2022). For this purpose, we used code snippets from the GitHub repository of Maddu et al. (2022). In preliminary studies, we found IDW to yield decent results but did not perform a systematic comparison of different weighting schemes. As evidenced by the results stated below, the PINNs consistently outperform the corresponding vanilla NN benchmark models. Thus, we refrained from further investigations into different weighting schemes. As IDW only works with first-order optimizers, we apply it only during the Adam optimization step and then keep the final weights constant for the L-BFGS optimization step. We start the training process with the Adam optimizer for 1000 epochs with a learning rate of 0.001. After that, we utilize the L-BFGS optimizer for 300 epochs. Mean squared error (MSE) is the metric used for minimization.

To determine the architecture parameters of the PINN, we utilize a grid search varying the following hyperparameters: activation function $\in \{\tanh, \text{sigmoid}\}$, depth of the hidden layers $\in \{1, 2, 3, 4\}$ and width of hidden layers (number of nodes) $\in \{16, 32, 64, 128\}$. We investigate all four models for both data regimes. The tanh activation function performs best in all cases. Moreover, we find that the best-performing width and depth of the hidden layers do not change across models but with the amount of training data. For the low-data regime, we find that a network with 2 hidden layers and 32 nodes performs the best. A network with 2 hidden layers and 64 nodes performs best for the high-data regime. The grid search is done with 5 randomly drawn data sets and 5 runs for each data set to account for variations in training/test split and weight initialization. Moreover, all the upcoming studies are also done using 5 data sets and 5 runs for each data set. The result of a run is reported as the average error over $N_{test} = 20$ trajectories.

3.2. Prediction of measured states

In this subsection, we investigate the generalization capabilities of the different PINN models and the vanilla ANN model listed in Table 3.

As can be seen from Fig. 4, the prediction error for all states decreases with increasing physical knowledge supplied to the models, except for the reactant concentration c_A in the low-data regime. Moreover, all PINN models perform better than the vanilla ANN model in predicting measured states for both data regimes. A particularly interesting result is that PINN-A performs better at estimating the measured



(a) Concentration of reactant c_A.

(b) Concentration of product c_B.

(c) Temperature of the tank T.

(d) Cooling jacket temperature T_K.

Fig. 4. Test set error for the measured states for all models and data regimes. Boxplots show the results of 25 models (5 runs each for 5 data sets), averaged over the test set of each model. The error metric is the mean absolute percentage error (MAPE).

states T and T_K than the vanilla ANN, even though both models predict these states only based on data, i.e., PINN-A does not have energy balances, and T and T_K do not appear in the mole balances. A possible explanation could be that, since PINN-A has physics knowledge on c_A and c_B, it reaches a lower loss value on c_A and c_B than the vanilla ANN and thus has more room to optimize for T and T_K.

We conclude that the PINN models show strong generalization capabilities, better than the purely data-driven model, especially in the low-data regime.

3.3. Algebraic state estimation

We now investigate if the PINN models can predict unmeasured algebraic states $\mathbf{y}^u$ with reasonable accuracy. First, we conduct an incidence matrix analysis for each PINN model. shows that all PINN models have a full-column rank incidence matrix, suggesting that state estimation is possible in all cases.

Table 6 reports test errors for the unmeasured algebraic states. Since the compared quantities are different for the different models, a direct comparison between the models is not justified. However, we can conclude that all models can predict the unmeasured algebraic states with acceptable accuracy (less than 10% mean absolute percentage error), except r_3 in PINN-B. We also observe that the accuracy gap between the low and high data regimes decreases as the provided physics knowledge increases.

Note that the estimated algebraic states, i.e., the net reaction rates (PINN-A), the individual reaction rates (PINN-B), and the reaction rate constants (PINN-C), were not only unmeasured, i.e., no process data was used for training, but also their corresponding constitutive equations were not provided. This example thus shows that PINNs can, in certain situations, infer immeasurable states, even if respective constitutive equations are unknown.

Table 5
Incidence matrices of PINN-A, PINN-B, and PINN-C for the Van de Vusse reactor example. If an unmeasured state appears in an equation, it is marked with a cross. Encircled crosses show feasible assignments of states to equations.

(a) Incidence matrix of PINN-A. Matrix has full-column rank.

$[\boldsymbol{f}, \boldsymbol{g}]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	r_A	r_B
Eqn. for $\dot{c}_A$	⊗	
Eqn. for $\dot{c}_B$		⊗

(b) Incidence matrix of PINN-B. Matrix has full-column rank.

$[\boldsymbol{f}, \boldsymbol{g}]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	r_1	r_2	r_3
Eqn. for $\dot{c}_A$	⊗		×
Eqn. for $\dot{c}_B$	×	⊗	
Eqn. for $\dot{T}$	×	×	⊗
Eqn. for $\dot{T}_K$			

(c) Incidence matrix of PINN-C. Matrix has full-column rank.

$[\boldsymbol{f}, \boldsymbol{g}]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	k_1	k_2	k_3
Eqn. for $\dot{c}_A$	⊗		×
Eqn. for $\dot{c}_B$	×	⊗	
Eqn. for $\dot{T}$	×	×	⊗
Eqn. for $\dot{T}_K$			

Table 6
Estimation accuracy for the unmeasured algebraic states $\mathbf{y}^u$ on the test set for all PINN models and data regimes. Results are averaged over 25 models (5 runs each for 5 data sets). The error metric is the mean absolute percentage error (MAPE).

Model	Unmeasured algebraic state	Low data regime	High data regime
PINN – A	r_A	4.71%	2.61%
	r_B	9.31%	5.12%
PINN – B	r_1	4.33%	3.43%
	r_2	9.15%	7.27%
	r_3	11.99%	10.42%
PINN – C	k_1	3.59%	2.90%
	k_2	6.84%	6.13%
	k_3	7.14%	6.98%

Table 7
Incidence matrices of PINN-C with $\mathbf{x}^u = [c_A]^T$, $\mathbf{x}^u = [T]^T$ and $\mathbf{x}^u = [T_K]^T$ for Van de Vusse reactor example. If an unmeasured state appears in an equation, it is marked with a cross. Encircled crosses show feasible assignments of states to equations.

(a) Incidence matrix of PINN-C with $\mathbf{x}^u = [c_A]^T$ (setting 1). Matrix does *not* have full-column rank.

$[f, g]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	c_A	k_1	k_2	k_3
Eqn. for $\dot{c}_A$	×	×		×
Eqn. for $\dot{c}_B$	×	×	×	
Eqn. for $\dot{T}$	×	×	×	×
Eqn. for $\dot{T}_K$				

(b) Incidence matrix of PINN-C with $\mathbf{x}^u = [T]^T$ (setting 2). Matrix has full-column rank.

$[f, g]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	T	k_1	k_2	k_3
Eqn. for $\dot{c}_A$		⊗		×
Eqn. for $\dot{c}_B$		×	⊗	
Eqn. for $\dot{T}$	×	×	×	⊗
Eqn. for $\dot{T}_K$	⊗			

(c) Incidence matrix of PINN-C with $\mathbf{x}^u = [T_K]^T$ (setting 3). Matrix has full-column rank.

$[f, g]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	T_K	k_1	k_2	k_3
Eqn. for $\dot{c}_A$		⊗		×
Eqn. for $\dot{c}_B$		×	⊗	
Eqn. for $\dot{T}$	×	×	×	⊗
Eqn. for $\dot{T}_K$	⊗			

3.4. Differential state estimation

We study PINN-C and create three different settings to empirically gauge the differential state estimation capabilities. In the first setting, we assume that state c_A is unmeasured i.e., $\mathbf{x}^u = [c_A]^T$. In the second setting, T is unmeasured, i.e., $\mathbf{x}^u = [T]^T$. In the third setting, T_K is unmeasured, i.e., $\mathbf{x}^u = [T_K]^T$. For all settings, the algebraic states k_1, k_2, and k_3 are also unmeasured, i.e., $\mathbf{y}^u = [k_1, k_2, k_3]^T$.

In the first setting, $\mathbf{x}^u = [c_A]^T$, we do not obtain a full-column rank incidence matrix, as can be seen from Table 7a, whereas in the other two settings we do (Tables 7b and 7c).

In Fig. 5, we see that the PINN model with $\mathbf{x}^u = [c_A]^T$ (setting 1) indeed fails to estimate c_A, as indicated by the incidence matrix (Table 7). In contrast, the MAPE values suggest that the PINN models with $\mathbf{x}^u = [T]^T$ (setting 2) and $\mathbf{x}^u = [T_K]^T$ (setting 3) yield good results for the estimation of the respective unmeasured differential states T and T_K. However, when we compare the results to the case where all differential states were measured (cf. Fig. 4), we see that the MAPE values are about 20 times higher in case of T_K, and around 5 times higher in case of T. More importantly, as the ranges of T and T_K are quite low compared to the actual values (cf. Table 4), the MAPE values can be deceptively low. Thus, as a more reliable measure of goodness of fit, we evaluate the coefficient of determination (R^2). As can be seen from Fig. 6, the PINN-C model with $\mathbf{x}^u = [T]^T$ can successfully predict the unmeasured differential state T, with R^2 scores above 0.90. However, the PINN-C model with $\mathbf{x}^u = [T_K]^T$ essentially fails to estimate T_K, with R^2 scores ranging between 0.15 and 0.85.

In state estimation theory (Kalman, 1960b; Lee and Markus, 1967), a system is called *observable* if the initial values of unmeasured states can be estimated uniquely using the information on measured states and a mathematical process model. In the particular example considered here, the initial state (and thus the trajectory) of T can be estimated uniquely by the PINN using the data on the measured states and the built-in physical knowledge, whereas this is not the case for T_K. Transferring observability conditions for nonlinear dynamic models, see, e.g., Lee and Markus (1967) and Kou et al. (1973), to PINNs is not straightforward and thus considered beyond the scope of this paper.

Table 8
Ranges of the initial state c_{A0} for the trajectories used in the training, test, and extrapolation sets.

Set	Initial State	Lower bound	Upper bound
Training set	c_{A0}	$2.14\,\frac{mol}{L}$	$2.57\,\frac{mol}{L}$
Test set	c_{A0}	$2.14\,\frac{mol}{L}$	$2.57\,\frac{mol}{L}$
Extrapolation set	c_{A0}	$1.71\,\frac{mol}{L}$	$2.14\,\frac{mol}{L}$

3.5. Extrapolation capabilities

We now explore if the PINN can extrapolate beyond the bounds of the process data supplied for training. For this purpose, we create a set of test trajectories with the initial value of c_{A0} out of the bounds of c_{A0} in the training trajectories. We term this set *extrapolation set*. In Table 8, respective ranges for the inputs c_{A0} can be seen for training, test, and extrapolation sets. In contrast to purely data-driven models, the PINNs may also learn the system dynamics from the physics residuals. Nevertheless, we still expect a lower accuracy in the extrapolation regime since we do not provide measurement data about that regime during training.

As can be seen from Fig. 7, the test errors on both the test and the extrapolation sets are much lower for the PINN models compared to the vanilla ANN model. We also observe that PINN-C, the model with the most physics knowledge, has the lowest difference in accuracy between test and extrapolation sets. Thus, we conclude that the PINN models can extrapolate better than the non-informed NN, and the extrapolation accuracies tend to increase when more physics knowledge is incorporated into the PINN.

4. Numerical example 2: Liquid-liquid separator

With this second example, we aim to investigate the generalization and state estimation capabilities of the PINN models under varying amounts of measured physical property data supplied as additional inputs to the NN. Thus, we now focus on the data-driven part of the PINN.

The dynamic liquid–liquid separator model shown below is based on the work of Backi et al. (2018, 2019). We included extensions for swarm sedimentation in the aqueous phase, convection terms for the drop size distribution (DSD) in the dense-packed zone (DPZ) analogously to Backi et al. (2018), and a state-of-the-art coalescence model (Henschke, 1995). The chosen swarm model (Mersmann, 1980) was also used to model liquid–liquid columns (Kampwerth et al., 2020) and takes the form of Stokes' law (Stokes, 2009) for diminishing hold-ups. Stokes' law was experimentally confirmed to model the outlet hold-up of liquid–liquid separator accurately (Ye et al., 2023).

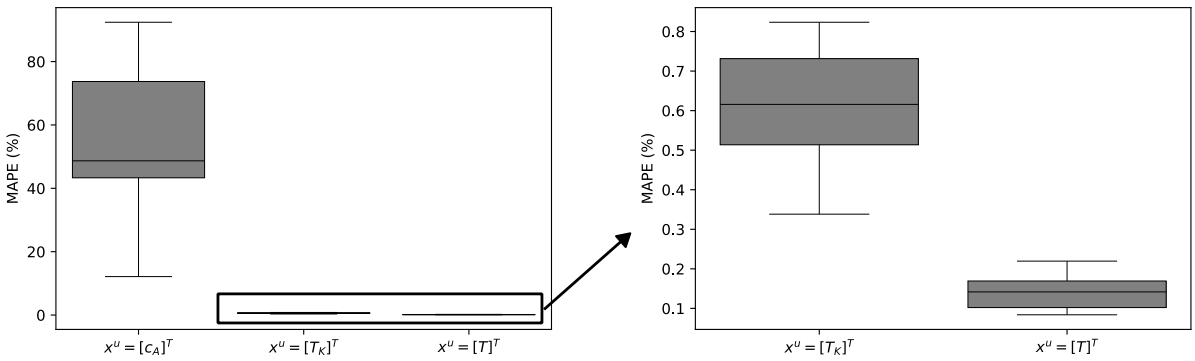


Fig. 5. Test set errors for the unmeasured differential states of PINN-C with $\mathbf{x}^u = [c_A]^T$, $\mathbf{x}^u = [T]^T$ and $\mathbf{x}^u = [T_K]^T$ for the Van de Vusse reactor example. All error values correspond to the respective unmeasured differential state, e.g., the value for the model with $\mathbf{x}^u = [c_A]^T$ shows the error of c_A. Boxplots show the results of 25 models (5 runs each for 5 data sets), averaged over the test set of each model. The error metric is the mean absolute percentage error (MAPE).

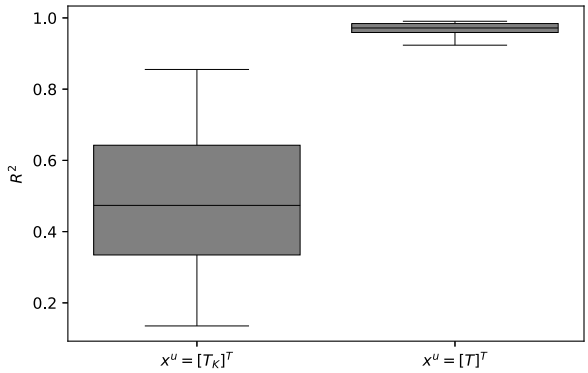


Fig. 6. R^2 values for PINN-C with $\mathbf{x}^u = [T_K]^T$ and $\mathbf{x}^u = [T]^T$ (test set goodness of fit).

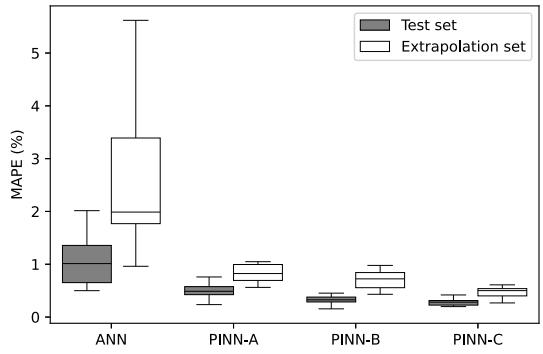


Fig. 7. Test and extrapolation set errors of the reactant concentration c_A for all models. The results are on the low-data regime. The error metric is the mean absolute percentage error (MAPE).

The considered liquid–liquid separator shown in Fig. 8 is divided into three subsystems: light (organic) phase, dense-packed zone (DPZ), and heavy (aqueous) phase. The light phase is assumed to be free of the dispersed phase; the DPZ is assumed to have a constant hold-up $\bar{\epsilon}_p$ (volume phase fraction of dispersed phase) of 0.9, and the heavy phase contains dispersed organic droplets but mostly water. The total volume flow $\dot{V}_{\text{in}}$ enters the separator in the heavy phase with the dispersed light phase described by the Sauter mean diameter d_{32} and phase fraction of organic phase ϵ_{in}. In the heavy phase, the dispersed droplets sediment upwards as a droplet swarm, resulting in the volume flow of organic droplets to the DPZ $\dot{V}_s$. In the DPZ, drop–drop coalescence is assumed to be negligible, and only droplet–interface coalescence occurs, giving the volume flow of coalesced drops $\dot{V}_c$ to the light phase. The volume flow of water $\dot{V}_w$ from the aqueous phase to the DPZ stems from trapped water between the sedimented droplets and coalesced drops at the interface of the organic phase. By applying a volume balance to the DPZ and assuming a constant hold-up, the volume flow of water can be expressed by the sedimentation and coalescence rate. The outlet volume flow of the aqueous $\dot{V}_{\text{aq,out}}$ and organic phase $\dot{V}_{\text{org,out}}$ are the manipulated variables of the settler, $\mathbf{u} = [\dot{V}_{\text{aq,out}}, \dot{V}_{\text{org,out}}]^T$.

The following volume balance equations are obtained after transforming the volume of a cylindrical segment to the height of each segment (Backi et al., 2018):

$$\dot{h}_L(t) = \frac{\dot{V}_{in}(t) - \dot{V}_{aq,out}(t) - \dot{V}_{org,out}(t)}{2L\sqrt{h_L(t)(2r - h_L(t))}}, \tag{5a}$$

$$\dot{h}_{\text{DPZ}}(t) = \frac{\dot{V}_{in}(t) - \dot{V}_{aq,out}(t) - \dot{V}_c(t)}{2L\sqrt{h_{\text{DPZ}}(t)(2r - h_{\text{DPZ}}(t))}}, \tag{5b}$$

$$\dot{h}_{\text{aq}}(t) = \frac{\dot{V}_{in}(t) - \dot{V}_{aq,out}(t) - \dot{V}_s(t)\frac{1}{\bar{\epsilon}_p} + \dot{V}_c(t)\frac{1-\bar{\epsilon}_p}{\bar{\epsilon}_p}}{2L\sqrt{h_{\text{aq}}(t)(2r - h_{\text{aq}}(t))}} \tag{5c}$$

Here, h_L, h_{DPZ}, and h_{aq} are the heights of the total liquid, the DPZ, and the aqueous phase, respectively, each measured from the bottom of the separator. They constitute the differential states $\mathbf{x}$ of the system. Note that the volume flow rates $\dot{V}_{in}$, $\dot{V}_{aq,out}$, $\dot{V}_{org,out}$, $\dot{V}_c$, and $\dot{V}_s$ are algebraic quantities. Similar to the CSTR case (Section 3), use of the dot notation to indicate flow rates is motivated by standard practice in engineering. In contrast, the dot symbols on the LHS of Eqs. (5a)–(5c) denote derivatives with respect to time. In the full-order mechanistic model (see Section SM4 of the Supporting Materials), the coalescence and sedimentation rates $\dot{V}_c$ and $\dot{V}_s$ are functions of h_{aq} and h_{DPZ}, boundary conditions at the entrance and physical properties such as the Sauter mean diameter d_{32} and the coalescence parameter r_{v}. As they cannot be measured, we aim to estimate $\dot{V}_c$ and $\dot{V}_s$ with a PINN model that uses only Eqs. (5a)–(5c) as available physical knowledge, i.e., the constitutive equations for the coalescence and sedimentation rates $\dot{V}_c$ and $\dot{V}_s$ are assumed to be unknown.

We assume that the total liquid height in the separator is constant, as this is the usual mode of operation. Then, the differential balance Eq. (5a) becomes an algebraic relation, serving as a closure condition for the flows in and out of the separator:

$$\dot{V}_{\text{in}}(t) - \dot{V}_{\text{aq,out}}(t) - \dot{V}_{\text{org,out}}(t) = 0$$

We also aim to investigate whether the PINN can take advantage of measurement data on d_{32} and r_{v} that are provided as input to the NN although these quantities do not appear in the physics Eqs. (5a)–(5c). Thus, we create three different PINN models with an increasing number of physical properties added as inputs to the NN, along with a vanilla NN for comparison (cf. Table 9). Moreover, we show the network structure of the models in the Supplementary Materials. As in Section 3, we make the time, states, and manipulated variables dimensionless for better performance and stability during NN training. We give the dimensionless variables and equations in the Supplementary Materials.

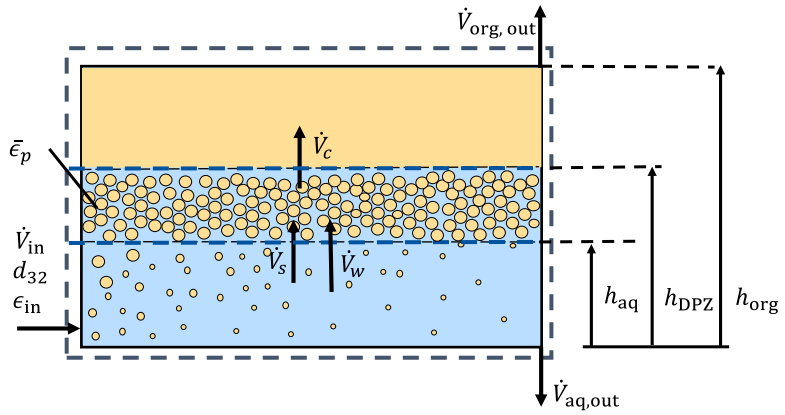


Fig. 8. Separator with the light phase (top), dense-packed zone (center), heavy phase (bottom), and flows. The dispersion with the properties phase fraction of dispersed phase ϵ_{in}, Sauter mean diameter d_{32}, and total volume flow rate $\dot{V}_{\text{in}}$ enters the heavy phase from the left. The heavy phase has the following outgoing flows: sedimentation rate $\dot{V}_s$, water flow rate $\dot{V}_w$ and outlet flow $\dot{V}_{\text{aq,out}}$. The dense-packed zone is modeled with a constant hold-up $\bar{\epsilon}_p = 0.9$ and a coalescence rate $\dot{V}_c$. The light phase has the outlet $\dot{V}_{\text{org,out}}$.

Table 9
Inputs of the models for the liquid–liquid separator.

Model name	Network inputs
Vanilla ANN	$t, h_{\text{aq}}(t_0), h_{\text{DPZ}}(t_0), \dot{V}_{\text{aq,out}}, \dot{V}_{\text{org,out}}$
Base PINN	$t, h_{\text{aq}}(t_0), h_{\text{DPZ}}(t_0), \dot{V}_{\text{aq,out}}, \dot{V}_{\text{org,out}}$
PINN-d_{32}	$t, h_{\text{aq}}(t_0), h_{\text{DPZ}}(t_0), \dot{V}_{\text{aq,out}}, \dot{V}_{\text{org,out}}, d_{32}$
PINN-d_{32}-r_{v}	$t, h_{\text{aq}}(t_0), h_{\text{DPZ}}(t_0), \dot{V}_{\text{aq,out}}, \dot{V}_{\text{org,out}}, d_{32}, r_{\text{v}}$

Table 10
Ranges for initial states and inputs for the liquid–liquid separator example.

Variable	Lower bound	Upper bound
$h_{\text{aq},0}$	0.090 m	0.110 m
$h_{\text{DPZ},0}$	0.108 m	0.132 m
$\dot{V}_{\text{aq,out}}$	4.5×10^{-4} m^3/s	5.5×10^{-4} m^3/s
$\dot{V}_{\text{org,out}}$	2.0×10^{-4} m^3/s	5.0×10^{-4} m^3/s
d_{32}	9.0×10^{-4} m	1.1×10^{-3} m
r_{v}	0.033	0.043

4.1. Data set generation, training, and hyperparameter selection

We investigate the phase separation of n-butyl acetate dispersed in water in a pilot-scale separator. The radius and length of the separator are $R = 0.1$ m and $L = 1.8$ m. We take the operating ranges presented in Table 10, with a selected time interval for step-wise control changes and thus process time of 20 s, i.e., $t \in [0, 20]$ s. We keep the manipulated variables constant throughout the process time for implementation reasons, as done in Section 3. Data generation to calculate the physics loss term $MSE_{physics}$ and the initial condition loss term MSE_{init} in Eqs. (2) are done by selecting $N_e = 10,000$ collocation and $N_i = 100$ initial value points. Again, the selection is done using Latin Hypercube sampling. We choose the bounds for the initial states $\mathbf{x}(t_0)$ corresponding to the minimum and maximum values of the states $\mathbf{x}$ in the operating range of the process (see Table 10), and perform similarly for the control variables $\mathbf{u}$. We use the explicit Runge–Kutta method of order 5 for the process data generation, utilizing *solve_ivp* solver from *scipy.integrate* module in Python (Virtanen et al., 2020; Dormand and Prince, 1980), with a relative and absolute error of 1×10^{-12}. We output the trajectories on a time grid $t \in [0, 20]$ s with $\Delta t = 0.1$ s. Nonphysical states, such as flooding of the separator with the DPZ, are addressed by early termination. The resulting shorter trajectories are kept in the data set; however, the step size Δt is adjusted to keep a constant number of grid points among all trajectories. We create $N_{total} = 200$ trajectories. From these, we select $N_{test} = 40$ trajectories for testing. For training, we use N_{train} trajectories, each having $N_m = 201$ data points. The total number of measurement points are $N_d = N_{train}N_m$. Again, we create two training sets from the remaining 160 trajectories not used for testing: a training set representing a *low-data regime* consisting of only $N_{train} = 20$ training trajectories, and a training set representing a *high-data regime* consisting of $N_{train} = 160$ training trajectories.

We use the strategy described in Section 3.1 for the training and hyperparameter optimization. For the low-data regime, we find that a network with two hidden layers and 32 nodes performs the best. A network with two hidden layers and 128 nodes performs best for the high-data regime. The tanh activation function performs best in all cases. The grid search is done with 5 data sets and 5 runs for each data set to account for variations in training/test split and weight initialization. Moreover, we use a sigmoid activation function for the output layer to bound the output values between 0 and 1 to prevent the square root in the denominator of Eqs. (5b) and (5c) from attaining negative values during PINN training. The following numerical studies are done with 5 data sets and 5 runs for each data set. The results of the runs are reported as the average error over $N_{test} = 20$ trajectories.

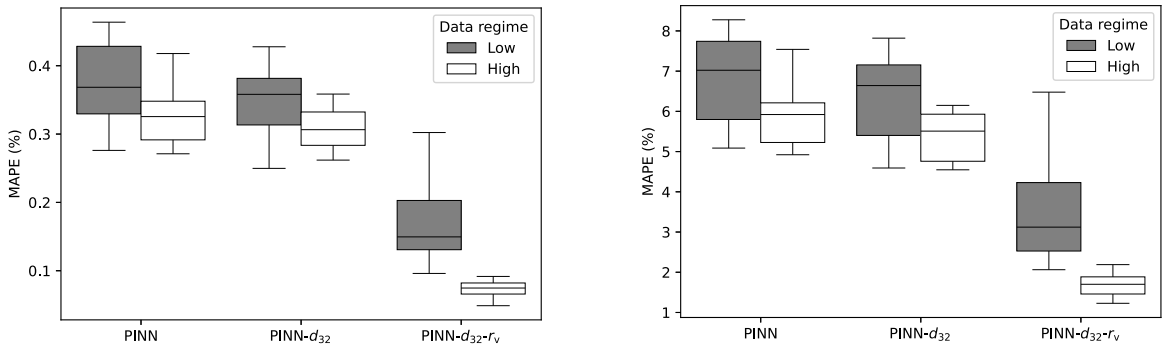
Table 11
Incidence matrix of the PINN models for the liquid–liquid separator. The matrix is identical for all three PINN models. If an unmeasured state appears in an equation, it is marked with a cross. Encircled crosses show the feasible assignment of states to equations. The matrix has full column rank.

$[f, g]\downarrow$ $[\mathbf{x}^u, \mathbf{y}^u]\rightarrow$	$\dot{V}_c$	$\dot{V}_s$
Eq. (5a)		
Eq. (5b)	⊗	
Eq. (5c)	×	⊗

4.2. Results

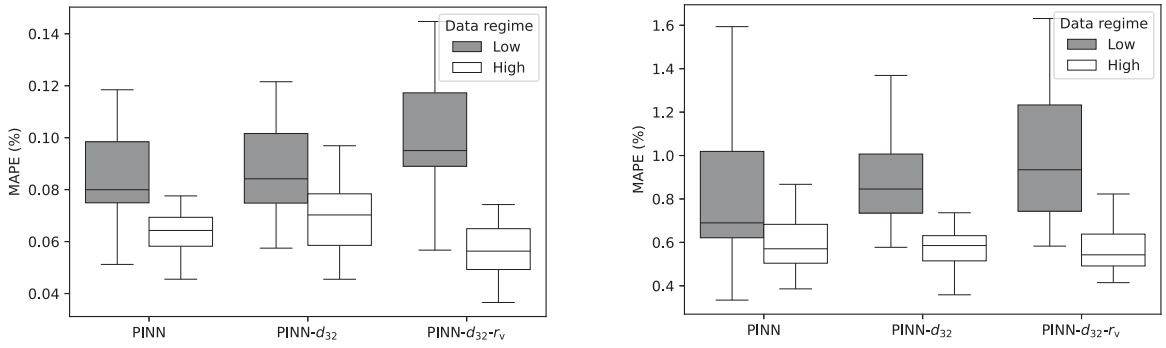
We show the incidence matrix of the liquid–liquid separator PINN model in Table 11. The unmeasured NN outputs are the algebraic states $\mathbf{y}^u = [\dot{V}_c, \dot{V}_s]$. The total liquid height is known and constant, i.e., $\dot{h}_L = 0$. The incidence matrix shows a feasible assignment and thus indicates possible state estimation.

We compare the prediction error of the states of the liquid–liquid separator model based on test set data. As can be seen from Fig. 9(a), the prediction accuracy of the DPZ height h_{DPZ} increases slightly with the addition of the Sauter mean diameter at the inlet d_{32} as a NN input. However, a more drastic increase can be noted if the coalescence parameter r_{v} is added as NN input. We see a similar trend with the estimation of the coalescence rate $\dot{V}_c$ in Fig. 9(b), although the accuracy increase is more apparent in the high-data regime. As explained in the Supplementary Materials, the coalescence parameter r_{v} plays a more direct role in the determination of the coalescence rate $\dot{V}_c$, which in turn



(a) Dense-packed zone height h_{DPZ}. (b) Coalescence rate $\dot{V}_c$.

Fig. 9. Test set error for the DPZ height h_{DPZ} and the coalescence rate $\dot{V}_c$ for all PINN models and data regimes. The error metric is the mean absolute percentage error (MAPE).



(a) Water height h_{aq}. (b) Sedimentation rate $\dot{V}_s$.

Fig. 10. Test set error for the water height h_{aq} and the sedimentation rate $\dot{V}_s$ for all PINN models and data regimes. The error metric is the mean absolute percentage error (MAPE).

has a high impact on h_{DPZ} (Eq. (5b)). The Sauter mean diameter at the inlet d_{32} has only an indirect role since the sub-model for coalescence and sedimentation in the full-order mechanistic model (see Section SM4 of the Supporting Materials) divides the separator into segments through the axial length. Thus, the Sauter mean diameter at each segment $d_{32,i}$ determines the coalescence rate rather than the value at the inlet. Moreover, since the PINN models are not trained with the data of the coalescence rate $\dot{V}_c$, and the sub-model for the coalescence rate is not provided as physics knowledge, the estimation accuracy of the coalescence rate $\dot{V}_c$ highly depends on the prediction accuracy of h_{DPZ}.

In Fig. 10(a), we observe that the prediction accuracy of the water height h_{aq} does not change notably with the addition of d_{32} and r_v as further NN inputs. We note a similar trend for the prediction of the sedimentation rate $\dot{V}_s$ in Fig. 10(b). These findings are not unexpected since the added physical properties play a negligible role in the sub-model for sedimentation rate $\dot{V}_s$ in the full-order mechanistic model and consequently for the water height h_{aq}.

The vanilla ANN performs considerably worse: For the DPZ height h_{DPZ}, the mean error of 25 models is 2.06% (MAPE) for the low-data regime and 1.78% (MAPE) for the high-data regime. For the water phase height h_{aq}, the mean error of 25 models is 1.33% (MAPE) for the low-data regime and 1.13% (MAPE) for the high-data regime. Note that the vanilla ANN cannot estimate the coalescence and sedimentation rates, $\dot{V}_c$ and $\dot{V}_s$, as no measurement data were available for training.

Overall, all PINN models show great generalization capabilities in the low-data regime for the prediction of h_{DPZ} which is a significant performance indicator for separation efficiency, with a maximum value of 0.46% for the mean absolute percentage error (MAPE). Moreover, the PINN models can estimate the unmeasured states, for which constitutive equations were assumed to be unknown, with a maximum error value of 8.28% for the coalescence rate $\dot{V}_c$, and with a maximum error value of 1.62% for the sedimentation rate $\dot{V}_s$. As a final remark, we observe that adding d_{32} and r_v as inputs to the PINN significantly improves the prediction of h_{DPZ} and the estimation of $\dot{V}_c$.

5. Conclusion and outlook

This paper investigates the PINN-based dynamic modeling of chemical engineering processes that are characterized by limited physical knowledge and limited data availability. Recognizing that certain process states, e.g., reaction rates or coalescence rates, often lack descriptive constitutive equations and cannot be measured, we set out to see if PINNs can infer such unmeasured states by leveraging known physical equations and data on measured states. To this end, we conducted numerical studies using two fully-known mechanistic process models and mimicking real-world modeling situations that are characterized by limited physical knowledge and data availability. Specifically, we assumed that certain equations would be unknown and thus unavailable for PINN development and that only a subset of process states would be measurable.

In both the Van de Vusse continuously stirred tank reactor (CSTR) example and the liquid–liquid separator example, we found that PINN models vastly outperform vanilla NNs of equal size, show superior generalization with respect to different initial states and control actions as well as superior extrapolation capabilities in regions of the state space without training data. Importantly, we observed that PINN models indeed may be capable of estimating unmeasured states, even if the corresponding constitutive relations are unknown. We provided a

heuristic for when the estimation of such unmeasured states might be successful. Although representing neither a necessary nor a sufficient condition for state estimation, the heuristic is easy to use and can be applied even before data collection is initiated.

Future work should concern the investigation of implicit DAE models in PINNs and whether the heuristic can be improved based on theory for observability of nonlinear dynamic systems, see, e.g., Lee and Markus (1967) and Kou et al. (1973). The feed-forward PINNs used in our work could also be compared to physics-informed *recurrent* neural networks that rely on time discretization, see, e.g., Zheng et al. (2023), or transformer-based PINN architectures, see, e.g., Zhao et al. (2023). Furthermore, PINN modeling and control of actual plant operations should be considered.

We conclude that PINN models with partial physical knowledge constitute a promising alternative to hybrid mechanistic/data-driven models in chemical engineering applications and warrant further investigation by the PSE community due to their potential to estimate states for which neither constitutive equations nor training data are available. Such further investigation should include performance comparisons between PINNs and hybrid models on identical tasks. For instance, for the estimation of immeasurable states $\boldsymbol{y}$ that lack constitutive equations, a DNN predicting measurable differential states $\boldsymbol{x}$ followed by a mechanistic model that uses (i) known balance equations, (ii) the predictions of $\boldsymbol{x}$, and (iii) estimates of $\dot{\boldsymbol{x}}$ obtained through automatic differentiation of the DNN to compute immeasurable algebraic states $\boldsymbol{y}$ would constitute a sequential hybrid model that could be compared to a PINN. Similarly, hybrid models recently proposed by Pahari et al. (2024) and Sitapure and Sang-Il Kwon (2023) use DNNs and time-series transformers, respectively, to estimate (spatio-)temporally varying quantities as inputs to mechanistic sub-models.

CRediT authorship contribution statement

Mehmet Velioglu: Writing – review & editing, Writing – original draft, Visualization, Supervision, Software, Methodology, Investigation, Conceptualization. **Song Zhai:** Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Sophia Rupprecht:** Writing – review & editing, Software, Methodology, Investigation. **Alexander Mitsos:** Writing – review & editing, Supervision, Methodology, Conceptualization. **Andreas Jupke:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Manuel Dahmen:** Writing – review & editing, Supervision, Methodology, Funding acquisition, 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.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 466656378 – within the Priority Programme “SPP 2331:Machine Learning in Chemical Engineering”. This work was performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE). We acknowledge financial support by the Helmholtz Association of German Research Centers, Germany through program-oriented funding. We would like to give special thanks to Lukas Polte, Fabian Mausbeck and Lukas Thiel (Aachener Verfahrenstechnik, Fluid Process Engineering, RWTH Aachen University) for their valuable suggestions and dedicated help on the separator model. We would like to give special thanks to Adel Mhamdi (Aachener Verfahrenstechnik, Process Systems Engineering, RWTH Aachen University) for providing valuable insight into the concepts of state estimation and observability.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.compchemeng.2024.108899.

Data availability

Data sharing not applicable to this article.

References

Alhajeri, M.S., Abdullah, F., Wu, Z., Christofides, P.D., 2022. Physics-informed machine learning modeling for predictive control using noisy data. Chem. Eng. Res. Des. 186, 34–49. http://dx.doi.org/10.1016/j.cherd.2022.07.035.

Antonelo, E.A., Camponogara, E., Seman, L.O., de Souza, E.R., Jordanou, J.P., Hübner, J.F., 2021. Physics-informed neural nets-based control. arXiv preprint, arXiv:2104.02556.

Arnold, F., King, R., 2021. State–space modeling for control based on physics-informed neural networks. Eng. Appl. Artif. Intell. 101, http://dx.doi.org/10.1016/j.engappai.2021.104195.

Asprion, N., Böttcher, R., Pack, R., Stavrou, M.E., Höller, J., Schwientek, J., Bortz, M., 2019. Gray-box modeling for the optimization of chemical processes. Chemie-Ingenieur-Technik 91 (3), 305–313. http://dx.doi.org/10.1002/CITE.201800086.

Backi, C.J., Emebu, S., Skogestad, S., Grimes, B.A., 2019. A simple modeling approach to control emulsion layers in gravity separators. In: 29th European Symposium on Computer Aided Process Engineering. In: Computer Aided Chemical Engineering, Vol. 46, Elsevier, pp. 1159–1164. http://dx.doi.org/10.1016/B978-0-12-818634-3.50194-6.

Backi, C.J., Grimes, B.A., Skogestad, S., 2018. A control- and estimation-oriented gravity separator model for oil and gas applications based upon first-principles. Ind. Eng. Chem. Res. 57 (21), 7201–7217. http://dx.doi.org/10.1021/acs.iecr.7b04297.

Barfoot, T.D., 2017. State Estimation for Robotics. Cambridge University Press.

Bradley, W., Kim, J., Kilwein, Z., Blakely, L., Eydenberg, M., Jalvin, J., Laird, C., Boukouvala, F., 2022. Perspectives on the integration between first-principles and data-driven modeling. Comput. Chem. Eng. 166, 107898. http://dx.doi.org/10.1016/j.compchemeng.2022.107898.

Brenan, K., Campbell, S., Petzold, L., 1996. Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. Vol. 14, SIAM.

Chen, H., Kremling, H., Allgöwer, F., 1995. Nonlinear predictive control of a benchmark CSTR. In: Proceedings of the 3rd European Control Conference, Rome-Italy. pp. 3247–3252.

Choi, S., Jung, I., Kim, H., Na, J., Lee, J.M., 2022. Physics-informed deep learning for data-driven solutions of computational fluid dynamics. Korean J. Chem. Eng. 39 (3), 515–528. http://dx.doi.org/10.1007/s11814-021-0979-x.

Dormand, J., Prince, P., 1980. A family of embedded runge-kutta formulae. J. Comput. Appl. Math. 6 (1), 19–26. http://dx.doi.org/10.1016/0771-050X(80)90013-3.

Duff, I., Gear, C., 1986. Computing the structural index. SIAM J. Algebr. Discrete Methods 7 (4), 594–603.

Gani, R., Cameron, I.T., 1992. Modelling for dynamic simulation of chemical processes: the index problem. Chem. Eng. Sci. 47 (5), 1311–1315. http://dx.doi.org/10.1016/0009-2509(92)80252-8.

Gelb, A., et al., 1974. Applied Optimal Estimation. MIT Press.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Henschke, M., 1995. Dimensionierung liegender Flüssig-flüssig-Abscheider anhand diskontinuierlicher Absetzversuche (Ph.D. thesis). In: Fortschritt-Berichte VDI : Reihe 3, Verfahrenstechnik, Vol. 379, RWTH Aachen University.

Iman, R.L., Helton, J.C., Campbell, J.E., 1981. An approach to sensitivity analysis of computer models: Part I—Introduction, input variable selection and preliminary variable assessment. J. Qual. Technol. 13 (3), 174–183. http://dx.doi.org/10.1080/00224065.1981.11978748.

Ji, W., Qiu, W., Shi, Z., Pan, S., Deng, S., 2021. Stiff-PINN: Physics-informed neural network for stiff chemical kinetics. J. Phys. Chem. A 125 (36), 8098–8106. http://dx.doi.org/10.1021/acs.jpca.1c05102.

Jin, X., Cai, S., Li, H., Karniadakis, G.E., 2021. Nsfnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations. J. Comput. Phys. 426, 109951. http://dx.doi.org/10.1016/J.JCP.2020.109951.

Kahrs, O., Marquardt, W., 2007. The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process. Process Intensif. 46 (11), 1054–1066. http://dx.doi.org/10.1016/J.CEP.2007.02.031.

Kahrs, O., Marquardt, W., 2008. Incremental identification of hybrid process models. Comput. Chem. Eng. 32 (4–5), 694–705. http://dx.doi.org/10.1016/J.COMPCHEMENG.2007.02.014.

Kalman, R.E., 1960a. A new approach to linear filtering and prediction problems. J. Basic Eng. 82 (1), 35–45. http://dx.doi.org/10.1115/1.3662552.

Kalman, R., 1960b. On the general theory of control systems. IFAC Proc. Vol. 1 (1), 491–502.

Kampwerth, J., Weber, B., Rußkamp, J., Kaminski, S., Jupke, A., 2020. Towards a holistic solvent screening: On the importance of fluid dynamics in a rate-based extraction model. Chem. Eng. Sci. 227, http://dx.doi.org/10.1016/j.ces.2020.115905.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3 (6), 422–440. http://dx.doi.org/10.1038/s42254-021-00314-5.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv preprint, arXiv:1412.6980.

Kou, S.R., Elliott, D.L., Tarn, T.J., 1973. Observability of nonlinear systems. Inf. Control 22 (1), 89–99.

Lagaris, I.E., Likas, A., Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. 9 (5), 987–1000. http://dx.doi.org/10.1109/72.712178.

Lee, E., Markus, L., 1967. Foundations of Optimal Control Theory. SIAM series in applied mathematics, Wiley.

Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. Math. Program. 45 (1–3), 503–528. http://dx.doi.org/10.1007/BF01589116/METRICS.

Maddu, S., Sturm, D., Müller, C.L., Sbalzarini, I.F., 2022. Inverse Dirichlet weighting enables reliable training of physics informed neural networks. Mach. Learn.: Sci. Technol. 3 (1), 015026. http://dx.doi.org/10.1088/2632-2153/ac3712.

Markidis, S., 2021. The old and the new: Can physics-informed deep-learning replace traditional linear solvers? Front. Big Data 4, http://dx.doi.org/10.3389/fdata.2021.669097.

Marquardt, W., 1996. Trends in computer-aided process modeling. Comput. Chem. Eng. 20 (6), 591–609. http://dx.doi.org/10.1016/0098-1354(95)00195-6, Fifth International Symposium on Process Systems Engineering.

Mersmann, A., 1980. Zum flutpunkt in flüssig/flüssig–Gegenstromkolonnen. Chem. Ing. Tech. 52 (12), 933–942. http://dx.doi.org/10.1002/cite.330521203.

Nabian, M.A., Meidani, H., 2019. Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. J. Comput. Inf. Sci. Eng. 20 (1), http://dx.doi.org/10.1115/1.4044507.

Nascimento, R.G., Fricke, K., Viana, F.A., 2020. A tutorial on solving ordinary differential equations using python and hybrid physics-informed neural network. Eng. Appl. Artif. Intell. 96, http://dx.doi.org/10.1016/j.engappai.2020.103996.

Pahari, S., Shah, P., Sang-Il Kwon, J., 2024. Unveiling latent chemical mechanisms: Hybrid modeling for estimating spatiotemporally varying parameters in moving boundary problems. Ind. Eng. Chem. Res. 63 (3), 1501–1514. http://dx.doi.org/10.1021/acs.iecr.3c03531.

Psichogios, D.C., Ungar, L.H., 1992. A hybrid neural network-first principles approach to process modeling. AIChE J. 38 (10), 1499–1511. http://dx.doi.org/10.1002/aic.690381003.

Queiroz, L., Santos, F., Oliveira, J., Souza, M., 2021. Physics-informed deep learning to predict flow fields in cyclone separators. Digit. Chem. Eng. 1, 100002. http://dx.doi.org/10.1016/j.dche.2021.100002.

Raissi, M., Perdikaris, P., Karniadakis, G.E., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. 378, 686–707. http://dx.doi.org/10.1016/j.jcp.2018.10.045.

Raissi, M., Yazdani, A., Karniadakis, G.E., 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science 367 (6481), 1026–1030. http://dx.doi.org/10.1126/science.aaw4741.

Roffel, B., Betlem, B., 2006. Process Dynamics and Control: Modeling for Control and Prediction. Process Dynamics and Control: Modeling for Control and Prediction, Wiley.

Sansana, J., Joswiak, M.N., Castillo, I., Wang, Z., Rendall, R., Chiang, L.H., Reis, M.S., 2021. Recent trends on hybrid modeling for industry 4.0. Comput. Chem. Eng. 151, 107365. http://dx.doi.org/10.1016/j.compchemeng.2021.107365.

Schweidtmann, A.M., Esche, E., Fischer, A., Kloft, M., Repke, J.-U., Sager, S., Mitsos, A., 2021. Machine learning in chemical engineering: A perspective. Chem. Ing. Tech. 93 (12), 2029–2039. http://dx.doi.org/10.1002/cite.202100083.

Schweidtmann, A.M., Zhang, D., von Stosch, M., 2024. A review and perspective on hybrid modeling methodologies. Digit. Chem. Eng. 10, 100136. http://dx.doi.org/10.1016/j.dche.2023.100136.

Shah, P., Sheriff, M.Z., Bangi, M.S.F., Kravaris, C., Kwon, J.S.-I., Botre, C., Hirota, J., 2022. Deep neural network-based hybrid modeling and experimental validation for an industry-scale fermentation process: Identification of time-varying dependencies among parameters. Chem. Eng. J. 441, 135643. http://dx.doi.org/10.1016/j.cej.2022.135643.

Sharma, N., Liu, Y.A., 2022. A hybrid science-guided machine learning approach for modeling chemical processes: A review. AIChE J. 68 (5), http://dx.doi.org/10.1002/aic.17609.

Sitapure, N., Sang-Il Kwon, J., 2023. Introducing hybrid modeling with time-series-transformers: A comparative study of series and parallel approach in batch crystallization. Ind. Eng. Chem. Res. 62 (49), 21278–21291. http://dx.doi.org/10.1021/acs.iecr.3c02624.

Stokes, G.G., 2009. On the effect of the internal friction of fluids on the motion of pendulums. In: Mathematical and Physical Papers. In: Cambridge Library Collection - Mathematics, Cambridge University Press, pp. 1–10.

von Stosch, M., Oliveira, R., Peres, J., Feyo de Azevedo, S., 2014. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. Computers and Chemical Engineering 60, 86–101. http://dx.doi.org/10.1016/j.compchemeng.2013.08.008.

Su, H.-T., Bhat, N., Minderman, P., McAvoy, T., 1992. Integrating neural networks with first principles models for dynamic modeling. IFAC Proc. Vol. 25 (5), 327–332. http://dx.doi.org/10.1016/S1474-6670(17)51013-7.

Tan, C., Cai, Y., Wang, H., Sun, X., Chen, L., 2023. Vehicle state estimation combining physics-informed neural network and unscented Kalman filtering on manifolds. Sensors 23 (15), 6665. http://dx.doi.org/10.3390/s23156665.

Thompson, M.L., Kramer, M.A., 1994. Modeling chemical processes using prior knowledge and neural networks. AIChE J. 40 (8), 1328–1340. http://dx.doi.org/10.1002/aic.690400806.

Unger, J., Kröner, A., Marquardt, W., 1995. Structural analysis of differential-algebraic equation systems—theory and applications. Comput. Chem. Eng. 19 (8), 867–882. http://dx.doi.org/10.1016/0098-1354(94)00094-5.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, I., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., Vijaykumar, A., Bardelli, A.P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C.N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D.A., Hagen, D.R., Pasechnik, D.V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G.A., Ingold, G.-L., Allen, G.E., Lee, G.R., Audren, H., Probst, I., Dietrich, J.P., Silterra, J., Webber, J.T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J.L., de Miranda Cardoso, J.V., Reimer, J., Harrington, J., Rodríguez, J.L.C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N.J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P.A., Lee, P., McGibbon, R.T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T.J., Robitaille, T.P., Spura, T., Jones, T.R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y.O., Vázquez-Baeza, Y., Contributors, S.., 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. Nature Methods 17 (3), 261–272. http://dx.doi.org/10.1038/s41592-019-0686-2.

van de Vusse, J.G., 1964. Plug-flow type reactor versus tank reactor. Chem. Eng. Sci. 19 (12), 994–996. http://dx.doi.org/10.1016/0009-2509(64)85109-5.

Wu, G., Yion, W.T.G., Dang, K.L.N.Q., Zhe, W., 2023. Physics-informed machine learning for MPC: Application to a batch crystallization process. Chem. Eng. Res. Des. 192, 556–569. http://dx.doi.org/10.1016/j.cherd.2023.02.048.

Yang, S., Navarathna, P., Ghosh, S., Bequette, B.W., 2020. Hybrid modeling in the era of smart manufacturing. Comput. Chem. Eng. 140, 106874. http://dx.doi.org/10.1016/j.compchemeng.2020.106874.

Ye, S., Hohl, L., Kraume, M., 2023. Impact of feeding conditions on continuous liquid-liquid gravity separation, part I: Inlet and outlet drop size, dense-packed zone and separation efficiency. Chem. Eng. Sci. 282, http://dx.doi.org/10.1016/j.ces.2023.119237.

Zendehboudi, S., Rezaei, N., Lohi, A., 2018. Applications of hybrid models in chemical, petroleum, and energy systems: A systematic review. Appl. Energy 228, 2539–2566. http://dx.doi.org/10.1016/J.APENERGY.2018.06.051.

Zhao, Z., Ding, X., Prakash, B.A., 2023. Pinnsformer: A transformer-based framework for physics-informed neural networks. arXiv preprint, arXiv:2307.11833.

Zheng, Y., Hu, C., Wang, X., Wu, Z., 2023. Physics-informed recurrent neural network modeling for predictive control of nonlinear processes. J. Process Control 128, 103005. http://dx.doi.org/10.1016/j.jprocont.2023.103005.

Zheng, Y., Wu, Z., 2023. Physics-informed online machine learning and predictive control of nonlinear processes with parameter uncertainty. Ind. Eng. Chem. Res. 62 (6), 2804–2818. http://dx.doi.org/10.1021/acs.iecr.2c03691.