Computers and Chemical Engineering 205 (2026) 109484

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

The incorporation of qualitative knowledge in hybrid modeling

Elia Arnese-Feffin [a], Nidhish Sagar [a], Luis A. Briceno-Mena [b], Birgit Braun [c], Ivan Castillo [c], Caterina Rizzo [d], Linh Bui [c], Jinsuo Xu [b], Leo H. Chiang [c], Richard D. Braatz [a,*]

[a] Department of Chemical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139, United States of America
[b] The Dow Chemical Company, Collegeville, PA 19426, United States of America
[c] The Dow Chemical Company, Lake Jackson, TX 77566, United States of America
[d] The Dow Chemical Company, Terneuzen, 4542 NM, The Netherlands

ARTICLE INFO

Dataset link: https://github.com/EliaAF/QualInfoHybridModelingPGNN

Keywords:
Hybrid modeling
Machine learning
Qualitative knowledge

ABSTRACT

Data-driven modeling enables the pursuit of model-based solutions when no or partial knowledge is available on the process of interest. The typically poor generalization performance of data-driven models can be improved by incorporating all available knowledge into the model development workflow, i.e. by hybrid modeling. However, how to incorporate this knowledge and what kind of knowledge to use have been strictly problem-dependent questions: whether a general framework for hybrid modeling can be devised remains an open research topic. In this article, we make the first step towards such an objective: we propose a method relying on theoretically sound modification of the objective function of data-driven models by incorporation of constraints derived from process knowledge, an approach that can be readily implemented in existing software packages with minimal overhead. We focus on an underutilized source of knowledge, i.e. *qualitative* knowledge. The proposed approach is demonstrated in two case studies, where we employ qualitative and quantitative process knowledge with varying degrees of complexity and discuss their effectiveness. Results show promising performance, paving the way for a truly general hybrid modeling framework.

1. Introduction

Mathematical modeling has long been a key asset in the chemical process industry. Mechanistic models (i.e. models derived from physical understanding of the phenomena of interest) are generally preferred because of their explainability and extrapolation capabilities, but can require significant investments of time and resources to be properly developed. On the other hand, data-driven models (i.e. derived from empirical observation of the phenomena) are increasingly used in the chemical process industry as they allow for fast development using plant data directly, at the cost of little to no explainability of the inner reasoning of the model and greater data requirements (Reis and Saraiva, 2021; Romagnoli et al., 2025; Schmidt and Wallace, 2024). The complementary strengths and weaknesses of mechanistic and data-driven models[1] sparked the field of *hybrid modeling*, in which both approaches are combined in a single model to facilitate model synthesis while promoting better interpretability and extrapolation performance, with lower data requirements (von Stosch et al., 2014; Schweidtmann et al., 2024; Narayanan et al., 2019).

Early explorations of hybrid modeling (Psichogios and Ungar, 1992; Thompson and Kramer, 1994; Schubert et al., 1994) combined mechanistic and data-driven models in a block-wise fashion: the material and energy balances of the process serve as the backbone of the model, while the data-driven element, most frequently an Artificial Neural Network (ANN), describes specific terms in the balance, e.g. biochemical kinetics. This hybrid modeling paradigm was systematized by Oliveira (2004) and von Stosch et al. (2014). Several alternative ways to incorporate mechanistic knowledge into the data-driven modeling framework have been explored, including the design of physically meaningful variables to be used in data-driven models (Severson et al., 2019; Arnese-Feffin et al., 2024b; Sansana et al., 2024), modification of the structure of data-driven models to reflect physical laws, such as conservation of mass (van Sprang et al., 2005; Beucler et al., 2021; Masi et al., 2021), and the inclusion of mechanistic models as constraints in

* Corresponding author.
E-mail address: braatz@mit.edu (R.D. Braatz).

[1] In this article, the term "mechanistic model" is intended to mean "a model derived from physical understanding of the phenomena of interest", whereas the term "data-driven model" is intended to mean "a model derived from data with the aid of machine learning algorithms" as those are the data-driven models of interest in this article.

https://doi.org/10.1016/j.compchemeng.2025.109484
Received 5 April 2025; Received in revised form 25 September 2025; Accepted 15 November 2025
Available online 18 November 2025
0098-1354/

the data-driven model training (Raissi et al., 2019; Daw et al., 2021; Koksal and Aydin, 2023).

Physics-Informed Neural Networks (PINNs) (Raissi et al., 2019) are among the most successful hybrid modeling methods. In principle, PINNs use disagreement with a mechanistic model, namely a partial differential equation (PDE), as a penalty in the objective function of an ANN. As a result, the ANN becomes a surrogate of the PDE as it is forced to balance fitting the training data and respecting the mechanistic knowledge encoded by the PDE. PINNs were successfully applied to chemical engineering problems, e.g. turbulence modeling in fluid dynamic simulation (Yang et al., 2019; Eivazi et al., 2022), heat transfer modeling (Koric and Abueidda, 2023), and process control (Wu et al., 2023).

PINNs provide an efficient and intuitive way of incorporating mechanistic knowledge into data-driven models, but remain limited by their nature of "surrogate-model-based solvers". Also, PINNs require an *exact* mechanistic model of the system (Karpatne et al., 2024; Schweidtmann et al., 2024) and biased models could be obtained if only imperfect mechanistic models were available. Given that perfect mechanistic models are rarely available in practice, a generalizable approach to PINNs is of interest.

Approaches similar to PINNs but more general in nature were proposed in the literature under different names, such as Physics-Guided Neural Networks (PGNNs) (Karpatne et al., 2017b) and physics-guided machine learning (Willard et al., 2022). These approaches can employ any mechanistic model as a penalty term in the objective function of the data-driven model. In an early study, Pukrittayakamee et al. (2009) proposed to augment the objective function of an ANN to penalize deviations from a reference value of the gradient of the outputs with respect to the inputs to model interatomic potentials. Daw et al. (2021), building on the work of Karpatne et al. (2017b), structured PGNNs to use a generic mechanistic model as a penalty term in the objective function of the ANN, considering both equality and inequality constraints. PGNNs were later extended to use recurrent neural networks as the base ANN model (Jia et al., 2019), although the proposed formulation was quite tied to the specific application, rather than being general. A summary of applications of PGNNs is available (Karpatne et al., 2024).

In the context of chemical engineering, Koksal and Aydin (2023) proposed a method that uses a generic mechanistic element acting as a hard or soft constraint[2] in the ANN training process. The authors explored three cases: *(i)* the mechanistic model is incorporated as a soft constraint in the objective function of the ANN; *(ii)* the mechanistic model is incorporated as a hard constraint in the optimization problems solved to train the ANN; *(iii)* the ANN is first trained on the available data alone, then parameters obtained in such a way are used to initialize a second training cycle in which the mechanistic model acts as a hard constraint. The proposed method was applied to the modeling and optimization of distillation processes with varying degrees of complexity.

Previous reports on the formulation of PGNNs (Daw et al., 2021) and PINNs (Koksal and Aydin, 2023) show some inconsistencies in the formulation of the objective function, as mechanistic constraints are imposed in different ways without clear justification. Furthermore, the use of qualitative knowledge, such as causality among variables, is well known to aid the modeling exercise and has been extensively studied in the data-driven process monitoring/fault detection literature (Vedam and Venkatasubramanian, 1999; Chiang and Braatz, 2003; Reis et al., 2019; Paredes et al., 2023), but is quite overlooked in the general hybrid modeling literature (Karpatne et al., 2017a; Narayanan et al., 2019; Bikmukhametov and Jäschke, 2020; Yang et al., 2020b; Sansana et al., 2021; Aykol et al., 2021; Karniadakis et al., 2021; Bradley et al., 2022; Willard et al., 2022; Sharma and Liu, 2022; von Rueden et al., 2023; Schweidtmann et al., 2024; Karpatne et al., 2024; Shah et al., 2025). Finally, mechanistic knowledge can be used to constrain the data-driven modeling training process in different ways: by adding a soft constraint to the objective function (Daw et al., 2021; Mukherjee and Bhattacharyya, 2025), by using the mechanistic model as a hard constraint in the optimization problem (Koksal and Aydin, 2023), or by including an additional output to a data-driven model (Lagerquist et al., 2021). These methods are viewed as alternatives in the literature (Willard et al., 2022), yet no discussion on their equivalences and relative strengths and weaknesses has been carried out.

This article aims to address the aforementioned limitations. We propose an approach to hybrid modeling based on a mechanistic penalty term added to the objective function of the data-driven model, thus following architecture B.1 in the taxonomy proposed by Aykol et al. (2021). We argue that this approach constitutes a general way to merge mechanistic and data-driven elements while not entailing complex, case-based block structures typical of other hybrid modeling strategies (von Stosch et al., 2014). We propose a defined mathematical function to include the mechanistic knowledge as a constraint in the data-driven model training, and provide a clear justification for our choice. A discussion on what kind of knowledge can be used to drive the data-driven model towards physically meaningful results is included, paying particular attention to the use of qualitative knowledge and providing several examples of its use. We further explore the use of mechanistic knowledge with different degrees of complexity, and demonstrate the proposed approach in two case studies: a pH neutralization process and a catalyst deactivation study.

The rest of this article is structured as follows. We introduce the relevant mathematical methods and objects in Section 2; we also summarize fundamental concepts on hybrid modeling therein. In Section 3, we outline the proposed approach, discuss and motivate each design choice, and provide examples of qualitative knowledge that might be employed in hybrid modeling. We apply the proposed approach to two case studies in Section 4: a pH neutralization process and the tracking of the activity of a catalyst. Finally, we draw the conclusions of this study and outline future research directions in Section 5.

2. Mathematical methods

This Section introduces the fundamental concepts on mathematical modeling used throughout this article. We also provide a brief summary of hybrid modeling methods, with particular focus on the methods of interest for this study, i.e. those relying on the modification of the objective function of data-driven models.

2.1. Mechanistic modeling

Mechanistic models in (bio-)chemical engineering can be found in different forms, the common trait being the ability of describing the inner working mechanism of the process being modeled. This article considers Differential-Algebraic Equation (DAE) systems, which have the general implicit form

$$\mathbf{0} = \psi\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}, \mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}; \mathbf{p}_{\mathrm{kb}}\right), \tag{1}$$

where t is the time, or, more generally, the independent variable of the model; ψ is a vector-valued function defining the evolution of the process with t; $\mathbf{x}$ and $\mathbf{z}$ are the vectors of differential and algebraic states of the model, respectively; $\mathbf{y}$ and $\mathbf{u}$ are the vectors of model outputs and inputs; $\mathbf{p}_{\mathrm{kb}}$ is the vector of parameters of the model (the subscript "kb" indicating "knowledge-based"). All variables in Eq. (1) are understood to depend on the dependent variable t (e.g. time-dependent); the explicit dependence is omitted for ease of notation. The

[2] The term *soft constraint* refers to a penalty that is directly incorporated into the objective function of an optimization problem, possibly with a given weight factor. The term *hard constraint* is a constraint stated in the optimization problems that must be exactly satisfied for the solution to be considered feasible.

parameters $\mathbf{p}_{\text{kb}}$ and the function ψ may be independent on t, but cases in which $\mathbf{p}_{\text{kb}}$ and/or ψ depend upon t exist as well (e.g. time-varying processes).

We remark that Eq. (1) denotes a class of mechanistic models frequently found in chemical engineering, i.e. DAEs in implicit form. Other classes of models exists, such as the DAE in explicit form

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = \chi(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}; \mathbf{p}_{\text{kb}}), \tag{2}$$

$$\mathbf{0} = \zeta(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{y}; \mathbf{p}_{\text{kb}}), \tag{3}$$

partial differential equations, ordinary differential equations, and purely algebraic systems. Any mechanistic model could be used in hybrid modeling, in principle. The choice of limiting our discussion to the model in Eq. (1) is without loss of generality of the hybrid modeling framework described herein.

2.2. Data-driven modeling

Given a mechanistic model of a process such as in Eq. (1) and the values of the parameters $\mathbf{p}_{\text{kb}}$, the model inputs $\mathbf{u}$ are assigned, the state variables $\mathbf{x}$ and $\mathbf{z}$ are initialized, and the model is integrated up to a given final time t_{fin} to compute the trajectories of the states and outputs $\mathbf{y}$. In such a model, the function ψ is assumed to be known and derived from first principles and/or mechanistic understanding of the phenomena driving the process, which "hard-code" the input/output relationship. In contrast, the data-driven modeling paradigm can be conceptualized as the use of some automatic learning algorithm to identify a function f such that

$$\hat{\mathbf{y}} = f(\mathbf{u}, \mathbf{x}, \mathbf{z}; \mathbf{p}_{\text{ml}}), \tag{4}$$

where $\hat{\mathbf{y}}$ is the prediction of $\mathbf{y}$ obtained by the model,[3] and $\mathbf{p}_{\text{ml}}$ is the vector of parameters of the function f, which are also identified in the learning process. The learning algorithm does not leverage any knowledge on the process determining the input/output relationship: the availability of a dataset $D = \{\mathbf{y}_n, \mathbf{u}_n, \mathbf{x}_n, \mathbf{z}_n\}_{n\in\{1,\dots,N\}}$ gathering N joint observations of the process variables is assumed, and then the function f is constructed and the parameters $\mathbf{p}_{\text{ml}}$ identified to match the predicted and observed outputs. As such, data-driven modeling is essentially an inductive approach leveraging data, whereas mechanistic modeling is a deductive workflow based on prior knowledge (Reis and Saraiva, 2021).

In practice, the form of the function f is assumed *a priori* as some parametrized machine learning model, and only the parameters $\mathbf{p}_{\text{ml}}$ are identified based on the dataset D. In regression problems (i.e. $\mathbf{y} \in \mathbb{R}^{N_{\text{o}}}$, with N_{o} being the number of output variables), this training amounts to solving the optimization

$$\mathbf{p}^*_{\text{ml}} = \arg\min_{\mathbf{p}_{\text{ml}}} L(\mathbf{p}_{\text{ml}}; f, D), \tag{5}$$

where $L(\mathbf{p}_{\text{ml}}; f, D)$ is some objective function (i.e. cost function or loss function) measuring the performance of the data-driven model f on dataset D when the parameters are set to $\mathbf{p}_{\text{ml}}$. The objective function is typically chosen as the Mean-Squared Error (MSE)

$$\begin{aligned} L(\mathbf{p}_{\text{ml}}; f, D) &= \frac{1}{N N_{\text{o}}} \sum_{n=1}^{N} \|\mathbf{y}_n - \hat{\mathbf{y}}_n\|_2^2 \\ &= \frac{1}{N N_{\text{o}}} \sum_{n=1}^{N} \|\mathbf{y}_n - f(\mathbf{u}_n, \mathbf{x}_n, \mathbf{z}_n; \mathbf{p}_{\text{ml}})\|_2^2, \end{aligned} \tag{6}$$

where $\|\cdot\|_2^2$ denotes the squared of the Euclidean norm (in $\mathbb{R}^{N_{\text{o}}}$, in this case). In other words, the optimal values of the parameters $\mathbf{p}_{\text{ml}}$ are chosen to minimize the MSE between the observed outputs and the predictions of the model. While Eq. (5) represents a general optimization problem to train a data-driven model, the specific choice of f may entail some slight variations in the objective function (6) and/or in the solution algorithm. For example, the simple linear regression model is typically based on the sum of squared errors and has an analytical solution, LASSO regression incorporates an L_1 penalty on the model parameters into the objective function and is trained by the least-angle regression algorithm, and ANNs may include weight-decay or weight elimination penalties in the objective function and are trained by the back-propagation algorithm or by fancier algorithms meant to reduced the likelihood of the optimizer to get trapped into one of the many local minima in the complex landscape of the ANN objective function (Hastie et al., 2009).

Recent reviews (Venkatasubramanian, 2018; Rendall et al., 2019; Reis and Saraiva, 2021; Mowbray et al., 2022) highlight many accomplishments of machine learning in chemical engineering and outline future research directions, signaling some room for improvement in an already very active research field. One of these directions, which is mentioned by nearly all of the reviews, is the incorporation of mechanistic knowledge by hybrid modeling. Therefore, before delving into the description of the data-driven model of interest for this article (i.e. ANNs), we provide a brief introduction to hybrid modeling in the next section. Then ANNs are summarized.

2.3. Hybrid modeling

Hybrid modeling stands at the interface of mechanistic modeling and data-driven modeling, drawing elements from both worlds. In contrast to the two aforementioned modeling paradigms, there is no single, commonly accepted description for hybrid modeling. There is significant variability even in the nomenclature used: see Sansana et al. (2021) and Schweidtmann et al. (2024) for a discussion on this topic. In this article, the term "hybrid modeling" is used to refer in general to "the integration of prior knowledge on the process being modeled and knowledge that can be extracted from process data".

Recent literature reviews from diverse fields (von Stosch et al., 2014; Karpatne et al., 2017a; Narayanan et al., 2019; Bikmukhametov and Jäschke, 2020; Yang et al., 2020b; Sansana et al., 2021; Aykol et al., 2021; Karniadakis et al., 2021; Bradley et al., 2022; Willard et al., 2022; Sharma and Liu, 2022; von Rueden et al., 2023; Schweidtmann et al., 2024; Karpatne et al., 2024) reveal a multitude of approaches to hybrid modeling, indicating the strong heterogeneity of the literature on the topic. Different scientific communities tend to prefer different hybridization approaches.

The most intuitive hybrid modeling approach is herein referred to as sub-modeling: i.e. the explicit combination of mechanistic and data-driven elements within the same overall model (Thompson and Kramer, 1994; von Stosch et al., 2014). The mechanistic component is typically seen as the backbone of the model, while the data-driven component can be used for residual modeling (Picabea et al., 2021), intermediate variables modeling (Arnese-Feffin et al., 2024a) or correction (Marques et al., 2017), or modeling the parameters of the mechanistic component (Rogers et al., 2023; Bui et al., 2022). On the other hand, the data-driven element can be viewed as the core component of the hybrid model, while mechanistic knowledge is used to enhance its performance. Examples are dataset augmentation (Destro et al., 2020), feature engineering (Wold et al., 2009; Severson et al., 2019; Arnese-Feffin et al., 2024b; Sansana et al., 2024), or structural modification (van Sprang et al., 2005; Beucler et al., 2021; Masi et al., 2021). Interested readers are referred to the cited literature for details on each approach.

Another approach to hybrid modeling is discussed in Section 1: the use of a mechanistic model as a penalty term in training a data-driven

[3] In this example, we explicitly pass the model states as input to the function f. However, a close correspondence with the rationale of the aforementioned mechanistic model is achieve by data-driven models with hidden states, where the states need to be just initialized and are internally evolved by the function f.

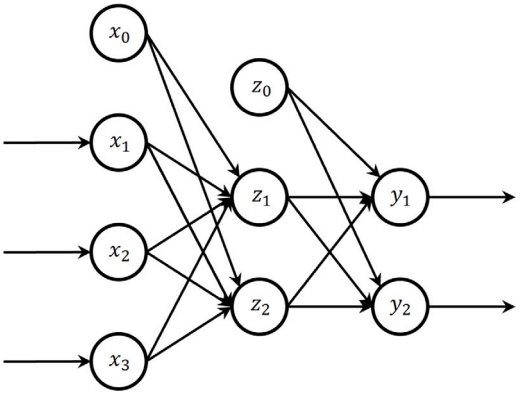

Fig. 1. Example of an ANN with three inputs ($\{x_1, x_2, x_2\}$), two outputs ($\{y_1, y_2\}$), and two neurons in the hidden layer ($\{z_1, z_2\}$). x_0 and z_0 are the unitary variables whose weights are the biases of the input and bout layers, respectively.

model. The most well-known method in this category is PINNs (Raissi et al., 2019), where a partial differential equation of the process is used as a regularization term during the training a deep neural network which parametrizes the solution to the differential equation itself. The rationale of PINNs can be generalized to use any mechanistic model as penalty term in the training of any data-driven model; with focus on ANNs, this idea has been explored by some studies (Lagerquist et al., 2021; Daw et al., 2021; Koksal and Aydin, 2023). In particular, Daw et al. (2021) and Koksal and Aydin (2023) proposed using generic, possibly imperfect mechanistic models to regularize the training process of an ANN with generic inputs and outputs. In this article, we refer to such an approach as PGNN, after the name proposed by Daw et al. (2021).

The next Section provides some background on the general rationale of ANNs. Then the principle of PGNNs is briefly described.

2.4. Artificial Neural Networks

ANNs are among the most impactful data-driven methods in process systems engineering (Venkatasubramanian, 2018). While there exist countless variations of ANNs to tackle a diversity of problems (Bishop and Bishop, 2024), here we limit our description to the simple feed-forward, fully connected architecture with one hidden layer for the sake of clarity. Interested readers can find further details and generalization to deep neural networks in reference textbooks (Bishop, 2006; Hastie et al., 2009; Bishop and Bishop, 2024).

Given a vector of input variables $\mathbf{u} \in \mathbb{R}^{N_\mathrm{i}}$ (N_i being the number of input variables) and a vector of output variables, the aforementioned ANN is described by

$$\hat{\mathbf{y}} = h_\mathrm{o}\big(\mathbf{b}_\mathrm{o} + \mathbf{W}_\mathrm{o} h_\mathrm{h}(\mathbf{b}_\mathrm{h} + \mathbf{W}_\mathrm{h}\mathbf{u})\big), \tag{7}$$

where $\hat{\mathbf{y}}$ is the ANN approximation of the (possibly unknown) true output $\mathbf{y} \in \mathbb{R}^{N_\mathrm{o}}$. In this equation, h_o, $\mathbf{b}_\mathrm{o} \in \mathbb{R}^{N_\mathrm{o}}$, and $\mathbf{W}_\mathrm{o} \in \mathbb{R}^{N_\mathrm{o}} \times \mathbb{R}^{N_\mathrm{h}}$ are the activation function, bias vector, and weight matrix of the output layer, respectively; and h_h, $\mathbf{b}_\mathrm{h} \in \mathbb{R}^{N_\mathrm{h}}$, and $\mathbf{W}_\mathrm{h} \in \mathbb{R}^{N_\mathrm{h}} \times \mathbb{R}^{N_\mathrm{i}}$ are the corresponding entities of the hidden layer. The ANN model (7) can be written in the compact notation $\hat{\mathbf{y}} = \mathrm{ANN}(\mathbf{u}; \mathbf{p}_\mathrm{ml})$, where $\mathbf{p}_\mathrm{ml} = \{\mathbf{b}_\mathrm{h}, \mathbf{W}_\mathrm{h}, \mathbf{b}_\mathrm{o}, \mathbf{W}_\mathrm{o}\}$. The typical representation of an example ANN is reported in Fig. 1.

The input variables are first linearly combined by the weights and biases of the hidden layer to produce a vector of hidden variables, then transformed element-wise by function h_h. This operation can be interpreted as an explicit transformation of the input variables by a set of basis functions defined by the parameters and activation function of the hidden layer. These transformed variables are then linearly combined and transformed by the output layer parameters and activation function to compute the outputs of the ANN. The ANN in Eq. (7) only uses the input variables of the process, while disregarding the state vectors. Therefore, the dataset required for training is $\mathcal{D} = \{\mathbf{y}_n, \mathbf{u}_n\}_{n\in\{1,\dots,N\}}$.

The activation functions (h_h and h_o) and number of neurons in the hidden layer (N_h) are degrees of freedom in the ANN design, and must be set prior to training. Sigmoidal functions, such as the hyperbolic tangent or the logistic function, are popular choices for the activation function of the hidden layer. Combined with a linear output layer (i.e. the activation function h_o is the identity function), this forms a function basis that endows the ANN with the universal approximation property: the ANN can approximate any continuous function with arbitrary accuracy, provided that a sufficiently large function basis is available (i.e. N_h is high enough) (Cybenko, 1989). Once N_h, h_h, and h_o are fixed, the parameters of the ANN, i.e. $\mathbf{p}_\mathrm{ml}$, are identified by the back-propagation algorithm in the ANN training (Bishop, 2006).

Various regularization techniques can be used to reduce the over-fitting tendency of ANNs (Bishop and Bishop, 2024). The simplest form is known as weight-decay, which is based on the continuous shrinkage properties of the L_2 regularization of ridge regression (Hoerl and Kennard, 1970). A regularization term[4]

$$R(\mathbf{p}_\mathrm{ml}) = \|\mathbf{p}_\mathrm{ml}\|_2^2 \tag{8}$$

is considered in the optimization (5), which minimizes the objective

$$J(\mathbf{p}_\mathrm{ml}; f, \mathcal{D}) = L(\mathbf{p}_\mathrm{ml}; f, \mathcal{D}) + \alpha R(\mathbf{p}_\mathrm{ml}) \tag{9}$$

rather than considering only $L(\mathbf{p}_\mathrm{ml}; f, \mathcal{D})$ defined in Eq. (6). The regularization term in Eq. (8) shrinks the weights and biases of the ANN towards zero, inducing a more uniform weight distribution and improving the generalization performance of the model. In Eq. (9), α is the weight of the regularization term in the optimization: the shrinkage increases as α increases. An optimal value of α is typically determined by cross-validation (Hastie et al., 2009). Generalized weight-decay functions use norms other than the Euclidean exist (Bishop and Bishop, 2024). Finally, we remark that $R(\mathbf{p}_\mathrm{ml})$ in Eq. (9) can be interpreted as a penalty on the complexity of the model. Therefore, minimizing Eq. (9) rather than Eq. (6) yields a model that balances the fit to the available data and the complexity of the model.

2.5. Physics-Guided Neural Networks

The weight-decay regularization of ANNs and, more in general, fancier regularization techniques (Bishop and Bishop, 2024) still make no use of the available knowledge on the process being modeled. On the other hand, PGNNs define the regularization term based on (possibly imperfect) mechanistic models to drive the input/output relationship encoded by the underlying ANN towards physically meaningful results (Lagerquist et al., 2021; Daw et al., 2021; Koksal and Aydin, 2023).

In the PGNN model (Daw et al., 2021), the ANN is trained by minimizing the objective

$$J(\mathbf{p}_\mathrm{ml}; f, \psi, \phi, \mathcal{D}, C, \mathbf{p}_\mathrm{kb}) = L(\mathbf{p}_\mathrm{ml}; f, \mathcal{D}) + \alpha R(\mathbf{p}_\mathrm{ml}) + \alpha_\mathrm{p} P(\mathbf{p}_\mathrm{ml}; f, \psi, \phi, C, \mathbf{p}_\mathrm{kb}), \tag{10}$$

where $P(\mathbf{p}_\mathrm{ml}; f, \psi, \phi, C)$ is a physics-based regularization term weighted in the objective function by α_p and is generally formulated as

$$\begin{aligned} P(\mathbf{p}_\mathrm{ml}; f, \psi, \phi, C, \mathbf{p}_\mathrm{kb}) = {} & \frac{1}{N_\mathrm{c}} \sum_{n=1}^{N_\mathrm{c}} \Big\| \psi(C_n; f, \mathbf{p}_\mathrm{ml}, \mathbf{p}_\mathrm{kb}) \Big\|_2^2 \\ & + \frac{1}{N_\mathrm{c}} \sum_{n=1}^{N_\mathrm{c}} \max\big\{0, \phi(C_n; f, \mathbf{p}_\mathrm{ml}, \mathbf{p}_\mathrm{kb})\big\}, \end{aligned} \tag{11}$$

[4] We acknowledge a mild abuse of notation in Eq. (8). By using the Euclidean norm, we have implicitly assumed that all the entities in the ANN parameter set $\mathbf{p}_\mathrm{ml}$ as defined above have been unfolded and gathered in a single vector.

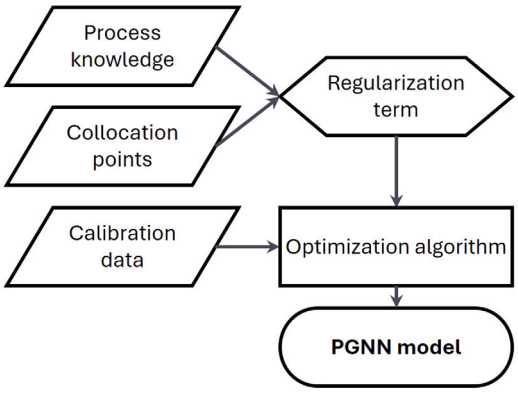

Fig. 2. Schematic representation of the PGNN model calibration workflow.

which uses the short-hand notation

$$\psi(C_n, f, \mathbf{p}_{\mathrm{ml}}; \mathbf{p}_{\mathrm{kb}}) = \psi\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_n, \mathbf{x}_n, \mathbf{z}_n, \mathbf{u}_n, f(\mathbf{u}_n; \mathbf{p}_{\mathrm{ml}}); \mathbf{p}_{\mathrm{kb}}\right) \tag{12}$$

$$\phi(C_n, f, \mathbf{p}_{\mathrm{ml}}; \mathbf{p}_{\mathrm{kb}}) = \phi\left(\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}_n, \mathbf{x}_n, \mathbf{z}_n, \mathbf{u}_n, f(\mathbf{u}_n; \mathbf{p}_{\mathrm{ml}}); \mathbf{p}_{\mathrm{kb}}\right). \tag{13}$$

In Eq. (11), ψ is the mechanistic model defined in Eq. (1), which is evaluated at a collection of collocation points, defined by the dataset $C = \{\mathbf{u}_n, \mathbf{x}_n, \mathbf{z}_n\}_{n\in\{1,\dots,N_c\}}$, and at the corresponding ANN predictions $\{\hat{\mathbf{y}}_n = f(\mathbf{u}_n; \mathbf{p}_{\mathrm{ml}})\}_{n\in\{1,\dots,N_c\}}$. Therefore, the first addendum in Eq. (11) drives the ANN training towards results consistent with the mechanistic model (1), which acts as a soft equality constraint in the optimization. On the other hand, the second addendum acts as a soft inequality constraint, being "active" (i.e. greater than 0) only when the second term in the max operator is positive. The function ϕ can be designed based on mechanistic knowledge to impose soft inequality constraints on the ANN training process. Note that the collocation points can be chosen arbitrarily to cover the portion of the input space of interest in a given application, and that the mechanistic constraints are enforced (in a soft way) only at locations defined by the elements in C. The equality constraint in Eq. (11) is stated by means of a squared Euclidean norm (i.e. L_2 regularization), while the inequality constraint is formulated as a positive part function (i.e. a form of L_1 regularization). The rationale of PGNN is schematically shown in Fig. 2.

The approach proposed by Lagerquist et al. (2021) is based on the incorporating auxiliary outputs into the model. The mechanistic model ψ is solved on the training dataset to obtain the corresponding outputs. The auxiliary outputs of the data-driven model are then trained to "predict" the outputs of the mechanistic model. Such an approach leads to an objective function similar to Eq. (11): the loss associated to the auxiliary outputs corresponds to an L_2 penalty between the predictions of the data-driven and mechanistic models (provided that the MSE is used as the loss function), thus acting as a regularization directly added to the overall loss. We refer the reader to Lagerquist et al. (2021) for details on the mathematical formulation.

Koksal and Aydin (2023) proposed three strategies to use mechanistic knowledge in ANNs, which lead to different formulation of PGNNs. A first formulation is closely related to that proposed by Daw et al. (2021): Eq. (10) is minimized in ANN training. However, Koksal and Aydin (2023) do not include the term $R(\mathbf{p}_{\mathrm{ml}})$ in their framework, and use the training dataset D both in the loss function and in the physics-based regularization term, rather than using collocation points in the latter. Furthermore, the authors do not specify any particular form for the physics-based regularization term, but only define it as a «physics function term». Indeed they used different forms of the term in their case studies, namely the squared Euclidean norm, the simple average of the deviations, and the square root of the simple average. The second and third formulations are based on hard constraints enforced while training the data-driven model. We refer the reader to Koksal and Aydin (2023) for additional details on such formulations.

3. Incorporation of qualitative knowledge

While PGNNs allow the incorporation of mechanistic knowledge in the data-driven modeling framework in an apparently seamless way, they are not free from drawbacks in their current form. In this Section, we propose a general way to include mechanistic knowledge in the data-driven modeling workflow. We pay particular attention to the inclusion of qualitative knowledge, and provide several examples of this valuable form of knowledge. Some of the examples are demonstrated in Section 4.

3.1. Proposed approach

In this article, we propose a general method to integrate mechanistic knowledge, either quantitative or qualitative, in data-driven modeling. Specifically, we incorporate mechanistic knowledge as a penalty term in the objective function to be minimized to train the data-driven model

$$J(\mathbf{p}_{\mathrm{ml}}; f, \psi, \phi, D, C, \mathbf{p}_{\mathrm{kb}}) = L(\mathbf{p}_{\mathrm{ml}}; f, D) + P(\mathbf{p}_{\mathrm{ml}}; f, \psi, \phi, C, \mathbf{p}_{\mathrm{kb}}), \tag{14}$$

and formulate the mechanistic penalty as

$$\begin{aligned} P(\mathbf{p}_{\mathrm{ml}}; f, \psi, \phi, C, \mathbf{p}_{\mathrm{kb}}) = \boldsymbol{\beta}^{\top}\left(\frac{1}{N_c}\sum_{n=1}^{N_c}|\psi(C_n; f, \mathbf{p}_{\mathrm{ml}}, \mathbf{p}_{\mathrm{kb}})|\right) + \\ + \boldsymbol{\gamma}^{\top}\left(\frac{1}{N_c}\sum_{n=1}^{N_c}\max\{0, \phi(C_n; f, \mathbf{p}_{\mathrm{ml}}, \mathbf{p}_{\mathrm{kb}})\}\right), \end{aligned} \tag{15}$$

where the same shorthand notation defined for Eq. (11) is used.

The objective function of our method, i.e. Eq. (14), includes only the loss function and a mechanistic penalty; we do not include the weight-decay term, as in Eq. (10).[5] Furthermore, we adopt the collocation points approach, i.e. we evaluate the mechanistic constraint on a dataset $C = \{\mathbf{u}_n, \mathbf{x}_n, \mathbf{z}_n\}_{n\in\{1,\dots,N_c\}}$ and on the corresponding data-driven model predictions $\{\hat{\mathbf{y}}_n = f(\mathbf{u}_n; \mathbf{p}_{\mathrm{ml}})\}_{n\in\{1,\dots,N_c\}}$. The collocation points can be designed to cover the region of interest in the input space, and may overlap with the datapoints in the training dataset D. The reasons for our choices are two-fold. On the one hand, we want to exploit the available mechanistic knowledge to reduce the overfitting in the regions of the input domain covered by the training dataset D. On the other hand, we exploit the same mechanistic knowledge (rather than empirical weight-decay terms) to improve the generalization performance of the model in regions of the input space with scarce density/missing observations. Both these objectives can be achieved with a careful design of the collocation points.

The formulation of the mechanistic penalty in Eq. (15) features two crucial differences with respect to that adopted by PGNNs, shown in Eq. (10). First, we introduce the weight vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, which are used in place of the single weight α_{p}. While this choice increases the number of parameters to be set prior to the training process, it allows to control individual weights of every equation in the mechanistic model ψ and in the inequality constraint function ψ. For example, equations that are known with a high degree of confidence (e.g. material balances) can be weighted more heavily than equations that have lower confidence (e.g. expressions for the kinetic rate laws of biological reactions). We also remark that the possibly different scales of the terms in the mechanistic penalties can be tackled by employing a normalization procedure appropriate for the chose data-driven model (e.g. normalization for

[5] The weight-decay penalty provides an empirical regularization aimed at reducing the complexity of the model in an "empirical" way. Such objective could conflict with the regularization operated by the mechanistic penalty, aimed at driving the model towards physically meaningful solutions.

ANNs), which simplifies the tuning of the individual weights in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$.

The second difference regards the use of the absolute value of the mechanistic model equations (i.e. L_1 regularization) in place of the squared Euclidean norm (i.e. L_2 regularization). An obvious reason for this choice is to uniformize the regularization class of the equality and inequality constraints, as compared to Eq. (10). However, a deeper reason for choosing the L_1 regularization for both constraints is that this penalty function is satisfied exactly for sufficiently large weights $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, which is typically not the case for the L_2 penalty (Zou and Hastie, 2005; Hastie et al., 2009). This property is exploited in LASSO regression to induce sparsity in the regression coefficients, to automatically select relevant variables (Tibshirani, 1996). In the context of hybrid modeling, the L_1 allows the strict enforcement of some mechanistic constraints, i.e. the constraints of high confidence, by assigning them *a priori* high weights in $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$. On the other hand, uncertain constraints can be satisfied only approximately, i.e. balancing physical consistency and data fitting. For such constraint, optimal values of the relevant components of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ can be set, e.g. by cross-validation (Hastie et al., 2009).

While our approach is clearly related to version of PGNNs proposed by Daw et al. (2021), it addresses all the limitations discussed in Section 1. Below we comment on the differences between the proposed approach and the methods discussed by Lagerquist et al. (2021) and Koksal and Aydin (2023).

A penalty term representing a mechanistic constraint can be indirectly added to the objective function of the data-driven model by incorporating auxiliary outputs into the model (Lagerquist et al., 2021). This strategy can be readily implemented, but offers scarce control over the penalty term: the form of the regularization is set by the loss function (e.g. L_2 regularization for an MSE loss); the penalty term cannot be weighted in the objective function by adjustable parameters; the mechanistic constraint is enforced only at the locations of the data in the training dataset, effectively preventing physics-based regularization outside of the training domain (i.e. improvement of the generalization performance of the model); no inequality constraints can be stated in a simple way; and the inclusion of auxiliary outputs increase the complexity of the data-driven model. In light of these limitations, the direct modification of the objective function appears a better way to include the mechanistic constraints.

Regarding the use of soft or hard constraints (Koksal and Aydin, 2023), it appears reasonable to state the mechanistic model as a hard constraint to the optimization problem solved to train the data-driven model. This approach would ensure that the mechanistic constraints are satisfied exactly. However, such an approach would also require the use of algorithms able to deal with constrained optimization problems, which are typically much more complex (i.e. computationally expensive) than algorithms for unconstrained problems (Nocadel and Wright, 2006). Furthermore, Koksal and Aydin (2023) investigated the performance of the approaches using soft or hard constraints, reporting no significant difference in the quality of the models. Finally, we remark that an exact correspondence between problems subject to hard constraints and problems including equivalent soft constraints can be established by the Lagrangian optimization theory (Zou and Hastie, 2005; Bishop and Bishop, 2024). The use of soft constraint thus appears as an appropriate solution.

Finally, we remark that the proposed approach does not rely on a specific data-driven model. In this article, we chose ANNs as an example data-driven model as they are very flexible and have been object of similar studies in the past. Furthermore, several powerful frameworks are already available for ANNs (e.g. Pytorch and TensorFlow for Python), which allow use to seamlessly integrate mechanistic penalties by defining custom objective functions. The proposed methodology can be easily extended to other data-driven models that are trained by gradient-based optimization. On the other hand, specific strategies are needed for data-driven model that leverage tailored training algorithm (e.g. regression trees). These topics are object of future research.

3.2. Examples of qualitative knowledge

The mechanistic penalty term in Eq. (14) relies on functions ψ and ϕ, which define the equality and inequality constraints, respectively. Although the use of quantitative mechanistic knowledge (i.e. mechanistic models) has been widely explored in the hybrid modeling literature, simple qualitative knowledge has been equally widely overlooked, with a relevant exception being in the process monitoring (Vedam and Venkatasubramanian, 1999; Chiang and Braatz, 2003; Reis et al., 2019; Paredes et al., 2023) and control (Yang et al., 2020a) literatures. Qualitative knowledge can provide valuable information on the behavior of the system being modeled, thus acting as a powerful regularization term in training the data-driven model of choice.

Below, we provide examples of qualitative knowledge to aid in the data-driven modeling of chemical processes. Some of the examples are explored in detail in Section 4. Additional details on the examples can be found in Appendix A.

The simplest example of qualitative knowledge is given by the bounds of variables, e.g. concentrations must be greater than 0 and the pH is upper bounded by 14. These represent *bound constraints* that can be stated by means of the function ϕ by constraining the values of the outputs of the data-driven models. Another example is given by *monotonicity constraints*: some variables are known to exhibit a monotonic relationship, e.g. the titration curve in pH neutralization processes (Wright and Kravaris, 1991), the flux–pressure relationship in membrane filtration (Arnese-Feffin et al., 2024b), or the pressure drop–velocity relationship in fluid flows (Bird et al., 2006). The functions ϕ can be used to state constraints on the derivatives of the outputs of the data-driven model.

Additional examples include *point constraints* and *boundary constraints*. In point constraints, the value of a function $f(u)$ is known for some value of u, e.g. a reaction rate is 0 if the concentration of the reactant is null. In boundary constraints, the value of a function $f(u, v)$ is known for some value of u regardless of the value of v, e.g. the convective component of heat transfer is 0 when a fluid is still (i.e. the Reynolds number is null) regardless of its thermal properties (i.e. the Prandtl number). Both these constraints can be stated by the function ϕ and enforced in the relevant domain regions by means of the collocation points C.

Table 1 clarifies how to implement the constraints discussed above by means of the function ψ and ϕ in (15). Inequality constraints require only equations for the data-driven model f and its derivatives, plus values of the relevant bounds. Monotonicity constraints can be stated also on transformation of the inputs by exploiting the rules of derivatives (see Section 4.2 for an example). Equality constraints require special attention. We recommend using the explicit version, in which a reference value for the output variables is pre-computed, which simplifies the formulation of the function ψ and avoids potential instabilities in training due propagation of the sensitivities of f through the equations of the mechanistic model. However, this requires solving the mechanistic model beforehand at all the collocation points rather than simply evaluating it during training. This operation might be potentially burdensome, especially with high-dimensional, complex mechanistic models. The tradeoff between explicit and implicit constraints, including the general design of collocation points and their effect of the training stability and model performance, are object of future research.

4. Case studies

This Section demonstrates the proposed framework for two case studies (simulations allow us to have full knowledge of the systems

Table 1
Examples of implementation of mechanistic constraints by means of the function ϕ and ψ. The data-driven model function f is understood to be evaluation at the collocation points, while the values of the relevant constraints are given or computed from the available mechanistic model.

Constraint	Category	Mapping
Lower bound	Inequality	$\phi = -\left(f(\mathbf{u}) - y_{\text{lb}}\right)$
Upper bound	Inequality	$\phi = -f(\mathbf{u}) - y_{\text{ub}}$
Decreasing monotonicity wrt u_i	Inequality	$\phi = \frac{\mathrm{d}}{\mathrm{d}u_i} f(\mathbf{u})$
Increasing monotonicity wrt u_i	Inequality	$\phi = -\frac{\mathrm{d}}{\mathrm{d}u_i} f(\mathbf{u})$
Point/boundary (explicit)	Equality	$\psi = f(\mathbf{u}) - y_{\text{eq}}$
Point/boundary (implicit)	Equality	Eq. (12)

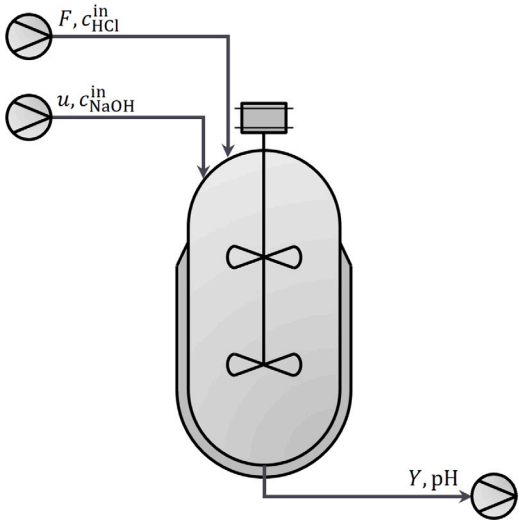

Fig. 3. Schematics of the process for the pH case study.

being modeled, which is needed to properly assess the PGNNs framework). A pH neutralization process is presented in Section 4.1, where a PGNN is used to parameterize the pH-strong acid equivalent relationship. In Section 4.2, a catalytic reaction system is considered: PGNNs are used to track the deactivation of the catalyst. For each case study, detailed mechanistic models are developed in MATLAB R2024a and used for data generation (see Appendices B and C for details). PGNN modeling is carried out in Python 3.10.14 using in-house developed code, which is available for download (see Data and Code Availability). For each case study, several PGNN models are developed with various degrees of process knowledge, both qualitative and quantitative, to illustrate their effects.

4.1. pH neutralization process

We consider a pH neutralization process in which a stream F of HCl (the process stream) with concentration $c^{\text{in}}_{\text{HCl}} = 0.01\,\text{mol/L}$ is neutralized by using a flowrate u of NaOH (the titrating stream) with concentration $c^{\text{in}}_{\text{NaOH}} = 0.1\,\text{mol/L}$. The liquid-phase reaction takes place in a tank reactor operating continuously, with constant temperature, constant density, and imperfect mixing conditions. The inlet streams are assumed to be saturated with aqueous CO_2, which takes part in hydration and equilibrium reaction. The water dissociation reaction is considered as well. All reactions are assumed to be at equilibrium at all times (i.e. quasi-equilibrium assumption). A scheme of the process is reported in Fig. 3.

In neutralization processes, the flowrate of the titrating stream is typically manipulated by a control system to achieve a desired pH in the outlet stream. While the pH is easily measured online, its relationship with u is strongly nonlinear, which makes the control problem nontrivial. The strong acid equivalent, Y, is typically used as the controlled variable instead of the pH (Wright and Kravaris, 1991). Model-based controller synthesis requires modeling the pH as a function of Y, i.e. identifying the so-called titration curve (Kim et al., 2012). In this case study, we use PGNNs to model the titration curve, i.e. $\text{pH} = f(Y; \mathbf{p}_{\text{ml}})$.

Data are generated based on a detailed mechanistic model of the process (see Appendix B for details on data generation). Time profiles of F, u, pH, and Y are available in the dataset. The detailed mechanistic model is used for data generation only (i.e. as the process) and is assumed to be unknown for the data-driven modeling task. Instead, a simplified mechanistic model based on some approximations is assumed to be available: *(i)* the effect of the CO_2 on the pH is neglected (i.e. the pH is affected only by HCl and NaOH); *(ii)* perfect mixing is assumed. Under these assumptions, a material balance of the process can be written in terms of strong acid equivalent (Wright and Kravaris, 1991),

$$V\frac{\mathrm{d}}{\mathrm{d}t}Y = Y^{\text{in}}_{\text{HCl}}F + Y^{\text{in}}_{\text{NaOH}}u - Y(F+u), \tag{16}$$

where $Y = c_{\text{HCl}} - c_{\text{NaOH}}$, thus $Y^{\text{in}}_{\text{HCl}} = c^{\text{in}}_{\text{HCl}}$ and $Y^{\text{in}}_{\text{NaOH}} = -c^{\text{in}}_{\text{NaOH}}$. An approximate titration curve (neglecting the buffering effect of the carbonate equilibria) can be derived analytically based in the definition of Y, which yields

$$\text{pH} = -\log_{10}\left(\frac{Y + \sqrt{Y^2 + 4K_w}}{2}\right) \tag{17}$$

where $K_w = 1 \cdot 10^{-14}$ is the water dissociation constant.

Eqs. (16) and (17) constitute quantitative knowledge on the systems, thus can be exploited in the proposed framework as physics-based regularization terms.

Qualitative knowledge is available as well. The general definition of the strong acid equivalent (Wright and Kravaris, 1991) implies that the titration curve is monotonically decreasing, thus

$$\frac{\mathrm{d}}{\mathrm{d}Y} f(Y; \mathbf{p}_{\text{ml}}) \le 0 \tag{18}$$

Furthermore, we see from Eq. (16) that the pH is bounded between $\text{pH}^{\text{in}}_{\text{HCl}} = 2$ and $\text{pH}^{\text{in}}_{\text{NaOH}} = 13$, which correspond to cases in which $u = 0\,\text{L/min}$ and $F = 0\,\text{L/min}$, respectively. Therefore, two additional bound constraints can be stated:

$$f(Y; \mathbf{p}_{\text{ml}}) \ge \text{pH}^{\text{in}}_{\text{HCl}} \tag{19}$$

$$f(Y; \mathbf{p}_{\text{ml}}) \le \text{pH}^{\text{in}}_{\text{NaOH}} \tag{20}$$

In the following, we demonstrate the effect of inclusion of mechanistic knowledge with incremental complexity in the PGNN model. The baseline model is a simple ANN with one input (Y) and one output (pH). One single hidden layer is considered, which includes nine neurons equipped with tanh activation function; the output layer has an identity activation function. The ANN is trained on the available data using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm.[6] The fit of the baseline model is reported in Fig. 4.

The strong nonlinearity of the titration curve is apparent from Fig. 4, which justifies the choice of ANN with tanh activation. However, we see that the simple ANN violates the slope and bound constraints, i.e. Eqs. (18), (19), and (20). Such constraints can be enforced using the proposed PGNN approach. To do so, we first defined the set of collocation points $\mathcal{C}$ as 2001 equispaced values of Y in the interval $[-0.1, 0.02]$, then exploit the available mechanistic knowledge to craft the functions ψ and ϕ to be used in the mechanistic penalty (15).

[6] We have chosen not to employ any of the traditional strategies to reduce overfitting in ANNs (e.g. stochastic optimization, mini-batching, dropout, etc.) to clearly illustrate the benefits of the physics-based regularization when it is used alone.

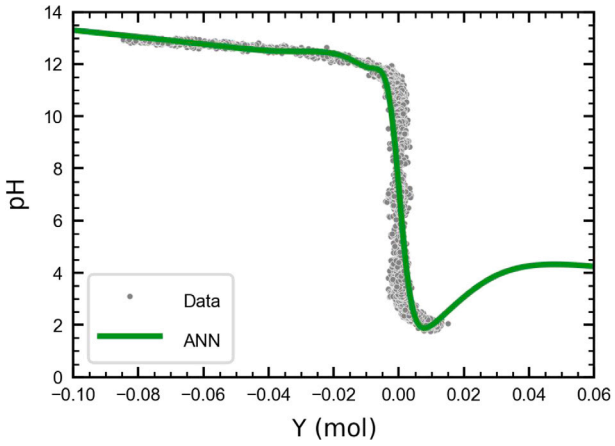

Fig. 4. Model 1.0: Baseline ANN model for the pH case study. Training data are depicted as gray dots, while the solid line is the ANN model fit.

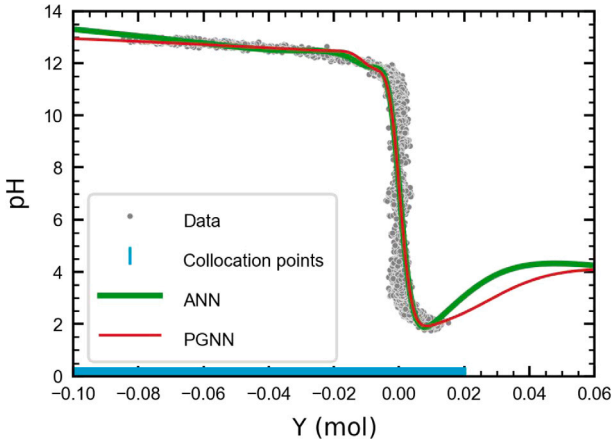

Fig. 5. Model 1.1: PGNN model for the pH case study with upper bound on pH. Training data are depicted as gray dots, while the cyan vertical lines mark collocation points. The thick green line is the baseline ANN model fit (same as Fig. 4), while the PGNN model fit is the thin red line.

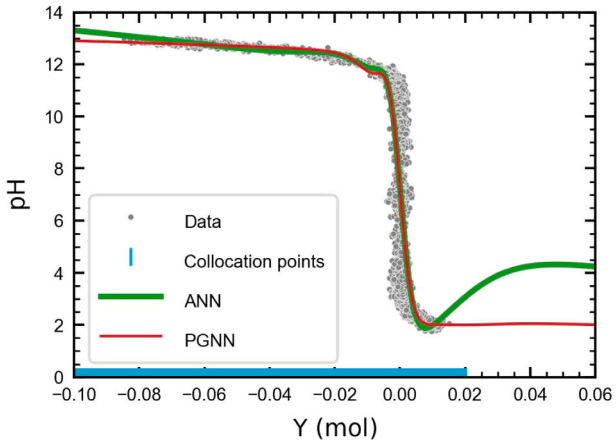

Fig. 6. Model 1.2: PGNN model for the pH case study with bounds on pH and monotonicity of the titration curve.

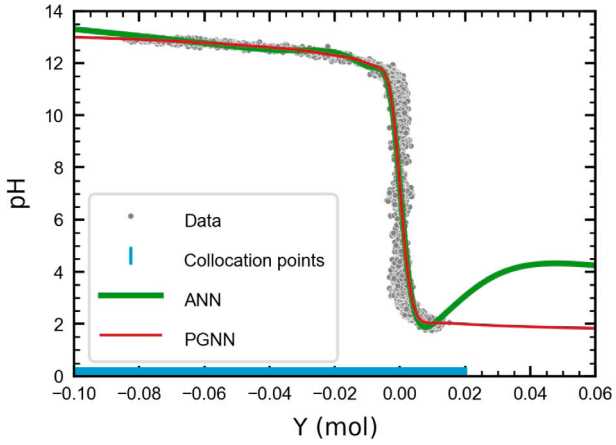

Fig. 7. Model 1.3: PGNN model for the pH case study with bounds on pH and monotonicity of the titration curve, plus equality constraint on the simplified mechanistic model (17).

We first consider a simple upper bound penalty based on Eq. (19), setting the constraint weight as $\gamma = 0.1$. The resulting PGNN is reported in Fig. 5. We clearly see that the constraint is satisfied by the PGNN.

Next, we add penalties for the lower bound and for the monotonicity, i.e. Eqs. (18) and (19). We represent the derivative in the monotonicity constraint by means of central finite difference discretization (Abramowitz and Stegun, 1965) on the collocation points. Weights for all constraints are set to $\gamma = 0.1$. The resulting PGNN is reported in Fig. 6.

The beneficial effect of the qualitative knowledge built into the PGNN is apparent from Fig. 6 as compared to the simple ANN. However, the model still shows some undesirable features, such as the "wiggle" around $Y = -0.01$. We can address this behavior by using Eq. (17), i.e. the approximate titration curve, as a mechanistic constraint. By doing so, we seek to balance the representation of the behavior of the real system (described by the data) and the smoothness encoded by the simplified mechanistic model at our disposal. A satisfactory balance is obtained by setting a $\beta = 0.2$ as weight for the mechanistic model constraint, while weights for the upper bound, lower bound, and monotonicity constraints are set as the components of the vector $\boldsymbol{\gamma} = (0.1, 2.5, 0.2^{\top})$. The resulting PGNN is reported in Fig. 7.

The PGNN model in Fig. 7 fits the data while having a physically consistent, smooth behavior. These characteristics not only reflect the physics of the process being modeled, but are also desirable for the solution of the aforementioned controller synthesis problem (Kim et al., 2012).

Table 2
Values of the training RMSE for each model of the pH case study. The "Model ID" refers to the figures reporting the model fit.

Model ID	Training
Model 1.0	2.0860
Model 1.1	2.0864
Model 1.2	2.0875
Model 1.3	2.0885

Finally, we assess the effects of the mechanistic regularization term in the PGNN objective function on the fitting quality of the model as measured by the Root-Mean-Squared Error (RMSE) between the model fit and the training data. Results for all models discussed in this Section are reported in Table 2. We see that the inclusion of mechanistic constraints can yield models with physically meaningful behavior while not significantly compromising the fitting quality. By enforcing a beneficial regularization on the final data-driven model, the hybrid models are also expected to have improved generalization capabilities. We will further investigate this point in the second case study.

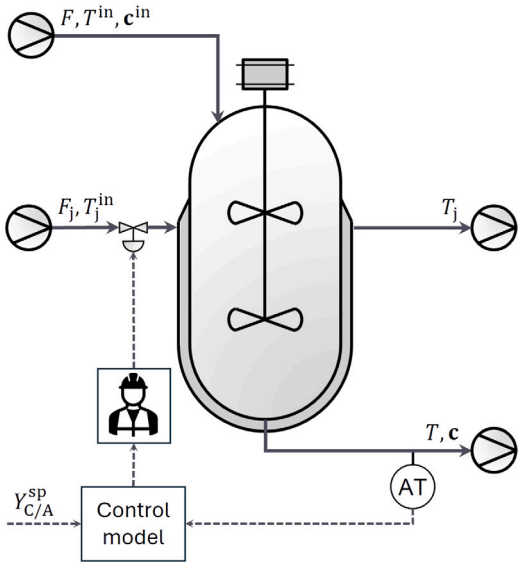

Fig. 8. Schematics of the process for the catalyst deactivation case study.

4.2. Catalyst deactivation tracking

The pH neutralization case study offers a simple testbed to understand and evaluate the effects of the mechanistic constraints in PGNNs. In this Section, we test the proposed PGNN model in a more challenging scenario: tracking the activity of a catalyst. A simplified process scheme is reported in Fig. 8. In the following, we describe only the fundamental aspects of the model of such process, with additional details reported in Appendix C.

We consider a catalytic reactor (perfectly mixed, non-isothermal) fed with a flowrate F of a mixture of two reactants, A and B, with concentrations c_A^{in} and c_B^{in}, respectively, and temperature T^{in}. Species A and B react to form the desired product, R, in the presence of a catalyst with relative activity $a \in [0, 1]$. A secondary reaction turns the desired product R into an undesired impurity S by reaction with A, and is unaffected by the presence of the catalyst. Both reactions are exothermic.

The catalyst activity decreases as function of the time-on-flow, until reaching an equilibrium value (i.e. the activity at which the rates or deactivation and self-regeneration are balanced) $a_{eq} = 0.15$. The deactivation rate is modeled by the generalized power law expression (Bartholomew, 1993; Forzatti and Lietti, 1999), thus it is sensitive to the reactor temperature.

The reactor temperature is controlled to maintain a given yield of R from A, which changes as the catalyst deactivates. The manipulated variables is the flowrate of cooling water, F_j, entering the jacket of the reactor (assumed to be perfectly mixed as well) with temperature T_j^{in}. The control logic simulates the action of an operator and is described in detail in Appendix C.

Monitoring catalyst activity is vital for the operation of the process and for maintenance planning. Modeling of catalyst activity is a subject of extensive research (Forzatti and Lietti, 1999; Shakor and Al-Shafei, 2023). Empirical models are most used for this task, with limited attempts to use mechanistic models due to the complexity of the deactivation mechanisms. Recently, hybrid models have shown promising performance in modeling catalyst activity (Bui et al., 2022). Therefore, we use PGNN to establish a relationship between the process variables, collectively denoted as $\mathbf{x}$, and the catalyst activity, i.e. $a = f(\mathbf{x}; \mathbf{p}_{ml})$.

We use the detailed mechanistic model of the process described above for data generation (additional details on the data generation can be found in Appendix C). The results presented in this Section are based on three "batches" or "catalyst charges": one training batch including only the in-control phase used for model training; one batch including both in-control and out-of-control phases used as collocation points to enforce mechanistic constraints in PGNNs; and one testing batch including both in-control and out-of-control phases used to test the predictive performances of the model.

In the modeling exercise, we assume to have knowledge of the main reaction only, thus neglecting the secondary reaction and the impurity S. Therefore, the available data include time profiles of F, T^{in}, c_i^{in}, F_j, T_j^{in}, c_i, T, and T_j, with $i \in \{A, B, R\}$. The time profile of a is available only for the training batch during modeling (time profiles of a are included also in the collocation points and testing batches, but will be used only to evaluate the performance of the models presented herein, not to train them).[7] Furthermore, we assume that catalyst deactivation develops much more slowly than the reaction and energy transfer dynamics, thus that all process variables besides a are at the steady state. Based on the aforementioned assumption, quantitative knowledge about the process can be represented in the form of material balances for species A, B, and R,

$$0 = F(c_A^{in} - c_A) - Var \tag{21}$$

$$0 = F(c_B^{in} - c_B) - Var \tag{22}$$

$$0 = F(c_R^{in} - c_R) + Var \tag{23}$$

where V is the volume of the reactor and r is the rate of the main reaction, defined by the kinetic model

$$r = k_0 e^{-\frac{\tilde{E}_a}{RT}} c_A^{o_A} c_B^{o_B} \tag{24}$$

where k_0 and $\tilde{E}_a$ are the pre-exponential factor and activation energy of the Arrhenius law, respectively, while o_i is the partial order of reaction of reactant i in the reaction. The energy balance of the reactor also constitutes quantitative knowledge on the process,

$$0 = \rho_m \hat{C}_{P,m} F(T^{in} - T) + \rho_w \hat{C}_{P,w} F(T_j^{in} - T_j) + V(-\Delta H_r)ar \tag{25}$$

where ρ_m and $\hat{C}_{P,m}$ are the density and specific heat of the reaction mixture, respectively, ρ_w and $\hat{C}_{P,w}$ are the corresponding properties of the cooling water, and ΔH_r is the enthalpy of reaction. The physical properties of the materials involved (ρ_m, $\hat{C}_{P,m}$, ρ_w, and $\hat{C}_{P,w}$) and the reactor volume (V) can be easily measured and are assumed to be known. On the other hand, the kinetic and thermochemical parameters of the reaction (k_0, $\tilde{E}_a$, o_A, o_B, and ΔH_r) cannot be easily measured; therefore they are fitted to the data in the training batch prior to the data-driven modeling exercise.

No mechanistic model is assumed to be available for the catalyst activity, which is estimated by PGNN modeling as function of the process conditions. However, qualitative knowledge on the catalyst activity is available. a is bounded between 1 and $a_{eq} = 0.15$, which translates into the bound constraints

$$f(\mathbf{x}; \mathbf{p}_{ml}) \geq a_{eq} \tag{26}$$

$$f(\mathbf{x}; \mathbf{p}_{ml}) \leq 1 \tag{27}$$

Finally, we assume that no regeneration of the catalyst is done along the batches, which implies that a is monotonically decreasing with the time-on-flow or, equivalently, the processed volume, defined as $V_f = \int_0^{t_{fin}} F \, dt$, where t_{fin} is the end-time of the batch. This provides an additional slope constraint

$$\frac{d}{dV_f} f(\mathbf{x}; \mathbf{p}_{ml}) \leq 0 \tag{28}$$

[7] In this example, we assume the catalyst activity, a, to be explicitly available as data. In a real application, a would need to be estimate by some model, based approach, e.g. as done by Bui et al. (2022).

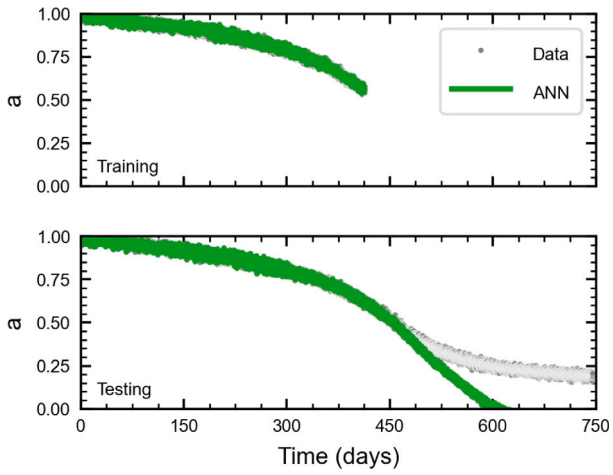

Fig. 9. Model 2.0: Baseline ANN model for the catalyst deactivation case study. Training data are depicted as gray dots, while the solid line is the ANN model fit. The top panel reports the fitting on the training batch, while the bottom panel shows the performance of the trained ANN on the testing batch.

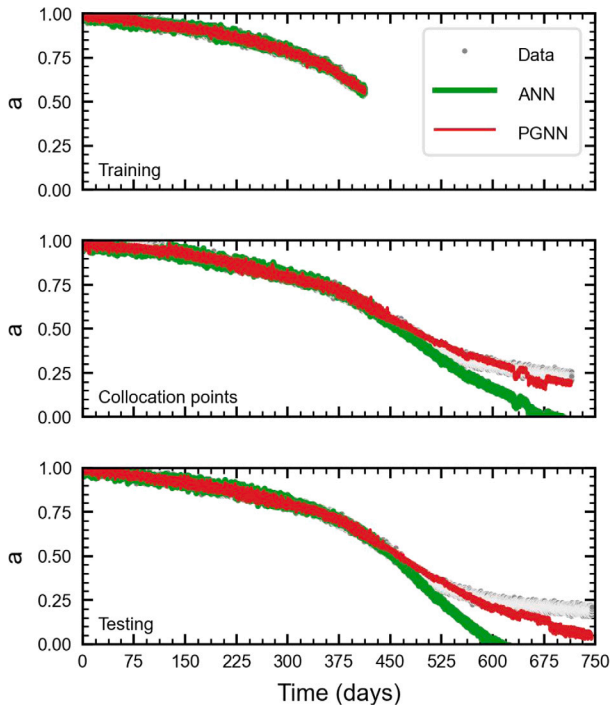

Fig. 10. Model 2.1: PGNN model for the catalyst deactivation case study with bounds on a and monotonicity of the activity vs. time-on-flow curve. Training data are depicted as gray dots. The thick green line is the baseline ANN model fit (same as Fig. 9), while the PGNN model fit is the thin red line. The top panel reports the fitting on the training batch, the middle panel shows a comparison of the model prediction with the activity in the collocation points batch, while the bottom panel shows the performance of the trained PGNN on the testing batch.

We first develop a baseline ANN model without using any of the available process knowledge. We select a subset of the available measurements as inputs to the ANN, namely the measurements featuring structured variability over the training data (i.e. not randomly oscillating around a constant value), which are F, T^{in}, F_{j}, T_{j}^{in}, c_i (with $i \in \{\text{A}, \text{B}, \text{R}\}$), T, and T_{j}. The activity is selected as the output of the ANN. Note that, coherently with the quasi-steady-state assumption made above, we seek a static relationship between the process conditions and the activity.[8] Therefore, we use a simple feed-forward ANN with one hidden layer counting twelve hidden neuron with Rectified Linear Unit (ReLU) activations,[9] and train the ANN on the training batch using the BFGS algorithm. The performance of the baseline ANN model is shown in Fig. 9, where both the training and testing batches are reported.

The model in Fig. 9 very closely fits the training data. The good performance of the model is also found in the initial part of the testing batch. However, performance greatly degenerates after $t \simeq 450\,\text{d}$. This is due to the switch from the in-control phase of the process to the out-of-control phase: data show a quite different behavior in the two phases (see Appendix C for details on the process phases). A simple ANN model has trouble generalizing to the conditions in the testing batch, even yielding results with no physical meaning (i.e. $a < 0$ after $t \simeq 600\,\text{d}$ in testing).

We therefore switch to the PGNN model, exploiting qualitative knowledge in the form of the bound constraints (26) and (27), plus the slope constraint (28), which are respectively weighted by the components of the vector $\boldsymbol{\gamma} = (0.1, 0.1, 0.017)^{\top}$ in the PGNN objective function. The derivative in the slope constraint is approximated by 4-point central finite difference discretization (Abramowitz and Stegun, 1965) on the collocation points. The resulting model is shown in Fig. 10. Recall that the mechanistic constraints are enforced on the collocation points batch (middle panel), and that the activity in such a batch is not used for model training or as a mechanistic constraint: it is reported in Fig. 10 to the sole purpose of easily visualizing the performance of the model.

While the performance of the PGNN model on the training data is basically unchanged with respect to the baseline ANN model, the inclusion of qualitative knowledge shows significant benefits in the generalization performance of the model. The bound constraints are perfectly satisfied; however, the slope constraint is only partially fulfilled. The data used in the catalyst deactivation case study feature several different noise sources, including process and measurement noise, plus periodic and dynamic variations in the feed variables, all of which is further fed back to the control inputs and states through the control system (see Appendix C for additional details). Numerical approximation of derivatives might be inaccurate in this scenario, even when using high-accuracy methods, thus the constraint could not be effective enough without destabilizing the training algorithm. If data are dense enough, smoothing splines (Hastie et al., 2009) can be used to approximate the activity vs. time-on-flow curve and to obtain a more accurate derivative approximation.

Another possible solution relies in smoothing out the PGNN model predictions using additional process knowledge. Material and energy balances of the process implicitly account for the effects of process noise and disturbance variations on the activity by combining the available measurements in a physically meaningful way. The PGNN model reported in Fig. 11 is obtained by stating bounds on the activity by Eqs. (26) and (27), plus equality constraints on the simplified mechanistic model in Eqs. (21) through (25) (recall that the kinetic and thermochemical parameters are fitted to the training data prior to data-driven modeling). The weights for the bound constraints are set to $\boldsymbol{\gamma} = (0.1, 0.1)^{\top}$ while the weights for the equality constraints (three

[8] The choice of a static model is also motivated by our intention to focus on the value of mechanistic constraints in PGNN modeling. Dynamic models of the activity would be more helpful in practice: we regard physics-based regularization of dynamic models as a future research direction.

[9] For this case study, we also investigated the performance of ANNs and PGNNs with different activation functions, numbers of hidden layers, and numbers of neurons in each hidden layer. No significant difference was found in the effect of the mechanistic regularization (as long as the model is flexible enough to fit the data), therefore we omit discussing the additional tests for brevity.

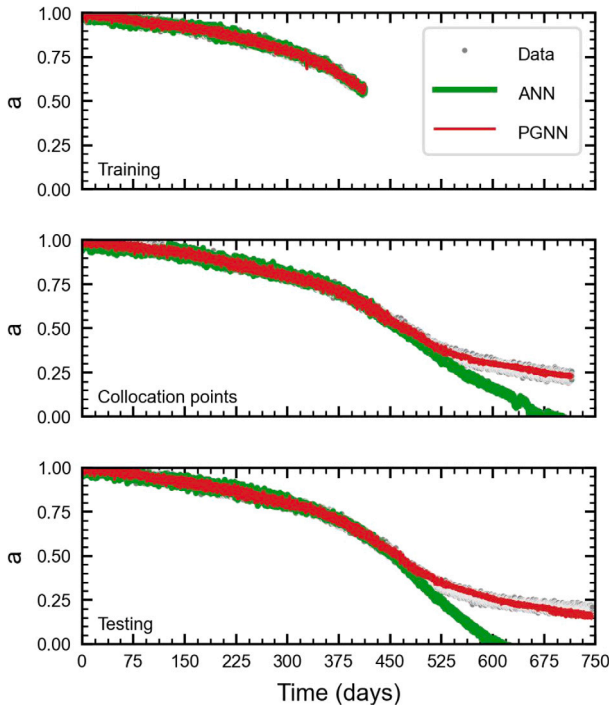

Fig. 11. Model 2.2: PGNN model for the catalyst deactivation case study with bounds on a and equality constraint on the simplified mechanistic model in Eqs. (21) through (25).

Table 3
Values of the training, testing, and extrapolation RMSE for each model of the catalyst deactivation case study. "Model ID" refers to the figures above.

Model ID	Training	Testing	Extrapolation
Model 2.0	0.01870	0.15599	0.24513
Model 2.1	0.01673	0.04998	0.07577
Model 2.2	0.01621	0.01810	0.01977

material balances and one energy balance) are set to constraints are set to $\boldsymbol{\beta} = (0.025, 0.025, 0.025, 0.025)^{\top}$.

The PGNN model shows excellent generalization performance when applied to the testing batch. Although simplified, the mechanistic model captures well the physics of the process, which effectively regularizes the training process and yields a physically consistent data-driven model.

The presence of testing data in the catalyst deactivation case study allows us to explicitly evaluate the improvement in the generalization capabilities of the models induced by the mechanistic regularization. Table 3 reports the training and testing RMSEs between of each one of the models discussed in this Section. The table also reports an estimate of the extrapolation error, which is computed on the portion of the testing batch outside of the range of activities found in the training data. As in the previous case study, the mechanistic constraints do not cause a degradation of the fitting performance of the models. However, we also explicitly see the improvement in the generalization performance of the models, as measured by the RMSE in testing. The extrapolation RMSE highlights that such improvement happens is most significant in the region not covered by the training data, providing strong evidence for the value of the mechanistic regularization.

In conclusion, we shall remark that formulating qualitative process knowledge is typically quite easy. The results presented in the case studies above show that embedding such knowledge in PGNN models can significantly enhance their generalization performance. Quantitative process knowledge can further help on this side, even if only simplified models are available. However, we shall remark that such models may be simple, but need to be correct, i.e. to capture at least the major trends in the data. For the same reason, constraints must not be conflicting (i.e. a feasible region must exist). While these requirements may seem trivial, neglecting them can lead to poor models. A simple example is the use of mechanistic models that do not match the physics of the process.

As a matter of example, assume the use of a reaction rate law with elementary kinetics in the simplified mechanistic model, i.e. set $o_A = 1$ and $o_B = 1$ in Eq. (24). This assumption leads to a severe process-model mismatch, which in turn destabilizes the PGNN training algorithm. This confirms the findings of previous studies on the matter (Rogers et al., 2023): using only simplified mechanistic knowledge with low uncertainty is preferable over using detailed knowledge with high uncertainty.

5. Conclusions

This article describes an approach for using qualitative (and quantitative) knowledge to aid in the data-driven modeling of process data. After surveying contributions from the literature, we propose a hybrid modeling method based on the modification of the objective function of the data-driven model. Namely, we include a regularization term derived from the available process knowledge, and provide clear theoretical justifications for the chosen formulation of the said term. As our approach works by directly modifying the objective function of the data-driven model, it can be applied by using existing code for data-driven modeling, thus being readily available to practitioners (the only overhead being the coding of the additional regularization term), as opposed to the tailored training procedures typically required by the classical block-structure hybrid modeling strategy.

Focusing on Artificial Neural Networks (ANNs) as data-driven model of choice, we demonstrated the proposed Physics-Guided Neural Networks (PGNNs) approach for two case studies. A simple pH neutralization process, in which PGNNs are used to parameterize the titration curve, allowed us to demonstrate the beneficial effects of different constraints based on quantitative knowledge. We then tested the proposed approach on a more realistic and challenging case study: tracking of the catalyst activity. When compared to pure data-driven modeling methods, the proposed approach significantly improved the generalization performance of the model, paving a promising way for the application of PGNNs in real, industrial settings.

While our method represents a first, important step towards a truly general hybrid modeling paradigm, we deem several other directions worth exploring in future research. We remark that we demonstrated the proposed approach based on ANNs, but it is not tied to any specific data-driven models. Other models can be assessed. Our approach is based on a quite general formulation of the mechanistic constraints. In the case of quantitative models used as constraints, several classes of models can be used, e.g. steady-state (algebraic) models or dynamic (differential) models. In this study, we have used only algebraic constraints, while the use of dynamic models as constraints remain to be evaluated. Furthermore, the distribution of collocation points can have a strong impact on the generalization performance of the model, as it determines where the mechanistic constraints are enforced. This topic deserves further research. Finally, we have assumed that any parameters in the mechanistic models are known or easy to determine. This might not be the case in some applications. The simultaneous identification of parameters of both mechanistic and data-driven elements of a hybrid model remains an open research direction in hybrid modeling.

CRediT authorship contribution statement

Elia Arnese-Feffin: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Nidhish Sagar:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Luis A. Briceno-Mena:** Writing – review & editing, Project administration, Methodology. **Birgit Braun:** Writing – review & editing, Conceptualization. **Ivan Castillo:** Writing – review & editing, Conceptualization. **Caterina Rizzo:** Writing – review & editing. **Linh Bui:** Writing – review & editing, Methodology. **Jinsuo Xu:** Writing – review & editing. **Leo H. Chiang:** Writing – review & editing. **Richard D. Braatz:** 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.

Acknowledgment

Financial support from The Dow Chemical Company is acknowledged.

Appendix A. Examples of qualitative process knowledge

Examples of qualitative process knowledge were introduced in Section 3.2. In this Section, we provide additional details, including mathematical formulation for each case. Each type of qualitative knowledge is motivated by specific chemical process examples.

A.1. Bounds and slope constraints for pH titration curves

Consider a pH neutralization process in which an acid stream must be neutralized (i.e. its pH brought to 7) with a base stream before wastewater collection. For complex mixtures of bases and acids, the process can be characterized by the strong acid equivalent Y, which is related to the pH by a strongly nonlinear function, the so-called titration curve. The identification of such function allows for the synthesis of controller to solve the neutralization problem (Wright and Kravaris, 1991).

If the composition of the system if known exactly, the titration curve can be modeled mechanistically (Wright and Kravaris, 1991). Data-driven models have been used for partially known systems in the form $\mathrm{pH} = f(Y)$ (Kim et al., 2012). Qualitative knowledge is available for the algebraic relationship between the strong acid equivalent and the pH. First, the relationship between Y and pH is negative monotonic,

$$\frac{\mathrm{d}}{\mathrm{d}Y}\mathrm{pH} < 0, \quad \forall\, Y. \tag{A.1}$$

Second, the pH of a solution obtained by mixing acid and base streams must lie between the pH of the acid and base streams,

$$\mathrm{pH}_{\mathrm{acid}} \leq \mathrm{pH} \leq \mathrm{pH}_{\mathrm{base}}, \quad \forall\, Y. \tag{A.2}$$

These two examples represent, respectively, a slope constraint and a bound constraint that can be use to regularize the training of the data-driven model by the function ϕ in Eq. (14).

A.2. Point constraints in crystal growth kinetics

In crystallization processes, the growth rate of crystal nuclei in a liquid solution, G can be modeled mechanistically based on the difference between the chemical potential of the molecule in the liquid and solid phases (Ohara and Reid, 1973). However, a fully mechanistic model is rarely used in practice because the determination of the activity coefficients needed to compute the chemical potential typically requires extensive experimental time and effort. Instead, the empirical model $G = f(S)$ is commonly used, where f is an unknown algebraic function and S is a measure of the supersaturation of the solution, e.g. $S = c - c^*$, with x being the solute concentration and c^* being the saturated solute concentration (i.e. the solubility). This approach absorbs the concentration dependence of the activity coefficients into the algebraic function f.

The function $f(S)$ must be equal to zero when $S = 0$. While it is simple to design a linear data-driven model implicitly incorporating such constraint, e.g. a linear regression model with no intercept, guaranteeing such a simple physical constraint may be challenging when flexible nonlinear models (i.e. the forms typically used to capture the complex relationship between G and S) are employed. One possible solution consists of re-parameterizing the unknown algebraic function as $f(S) = Sh(S)$, where $h(S)$ is a data-driven model that does not allow singularities (e.g. is always positive). An alternative, more straightforward way is to state a point constraint $f(S) = 0$ for $S = 0$[10] by means of the function ϕ, which can be used to drive the data-driven model training towards a physically consistent solution.

A.3. Boundary constraints in convective mass transfer

Consider the mass transfer between a fluid flow forced around or through an object. Simple transport analysis (Bird et al., 2006) shows that the Sherwood number $\mathrm{Sh} := k_c L/D$ is equal to 2 as the velocity v approaches zero for any object of any shape, where k_c is the convective mass transfer coefficient, L is the length scale of the object, and D is the molecular diffusivity of the solute in the fluid. This mechanistic knowledge is described by the parameterization

$$\mathrm{Sh} = 2 + h(\mathbf{p}) \tag{A.3}$$

where $\mathbf{p}$ is a vector of physical parameters including v, while h is an algebraic function such that and $f = 0$ for $v = 0$. The specific relevant physical parameters for any object having one relevant length scale is typically $\mathbf{p} = (L, v, \mu, \rho, D)^\top$, where μ and ρ are the fluid viscosity density, respectively.

The quantities in vector $\mathbf{p}$ are typically combined in dimensionless numbers to obtain a general description of the Sherwood number, leading to

$$\mathrm{Sh} = 2 + f(\mathrm{Re}, \mathrm{Sc}) \tag{A.4}$$

with Reynolds number $\mathrm{Re} := \rho v L/\mu$ and Schmidt number $\mathrm{Sc} := \mu/\rho D$, where f is a positive monotonic function of Re and Sc that is zero when the Reynolds number is zero. These conditions can be represented by the constraints

$$f(0, \mathrm{Sc}) = 0 \quad \forall\, \mathrm{Sc} > 0, \tag{A.5}$$

$$\frac{\partial}{\partial \mathrm{Re}} f > 0 \quad \forall\, \mathrm{Re} > 0, \tag{A.6}$$

$$\frac{\partial}{\partial \mathrm{Sc}} f > 0 \quad \forall\, \mathrm{Sc} > 0. \tag{A.7}$$

The first equality is a constraint that must be satisfied on one of the boundaries of the set of allowable values of $f(\mathrm{Re}, \mathrm{Sc})$, being therefore termed a boundary constraint. The second and third inequalities are

[10] The same approach can be used anytime the value of an unknown algebraic function is known for some value of its input.

slope constraints that must be satisfied for all allowable values of Re and Sc. While such constraints are typically built into empirical models, i.e. compositions of power laws of Re and Sc (Bird et al., 2006), it is not trivial to design a data-driven model f incorporating them by design. However, they can be stated by means of the function ϕ in an easy way.

Appendix B. Mechanistic model of the pH neutralization process

The detailed mechanistic model of the pH neutralization process described in Section 4.1 and schematized in Fig. 3 is based on material balances of the series of reactors in the tank-in-series model (Levenspiel, 1998) used to simulate imperfect mixing conditions. We set the number of tanks in the series to $K = 3$ and define the volume of each tank as $V_k = V/K$, with V being the overall volume of the reactor. The $N_C = 8$ species considered in the model, whose molar concentrations in the kth reactor are gathered in the vector $\mathbf{c}_k \in \mathbb{R}^{N_C}$, are CO_2, H_2CO_3, HCO_3^-, CO_3^{2-}, H^+, OH^-, Na^+, and Cl^-; the water concentration is not needed in the model equations as water is the solvent. These species take part in $N_R = 4$ reversible reactions: the hydration of CO_2, the carbonate acid equilibria, and the ionic equilibrium of water

$$CO_2 + H_2O \overset{K_h}{\rightleftharpoons} H_2CO_3 \tag{B.1}$$

$$H_2CO_3 \overset{K\_1}{\rightleftharpoons} HCO_3^- + H^+ \tag{B.2}$$

$$HCO_3^- \overset{K\_2}{\rightleftharpoons} CO_3^{2-} + H^+ \tag{B.3}$$

$$H_2O \overset{K_w}{\rightleftharpoons} H^+ + OH^- \tag{B.4}$$

where K_h is the hydration equilibrium constant of CO_2, and K_1 and K_2 are the first and second acid dissociation constants of H_2CO_3, respectively, and K_w is the water dissociation constant.

The material balance of the first reactor is

$$V_1 \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c}_1 = F\mathbf{c}^{\mathrm{in},F} + u\mathbf{c}^{\mathrm{in},u} - (F+u)\mathbf{c}_1 + V_1\mathbf{S}\mathbf{r}_1 \tag{B.5}$$

where F and u are the process and titrating streams, respectively, entering the reactor, $\mathbf{c}^{\mathrm{in},F} \in \mathbb{R}^{N_C}$ and $\mathbf{c}^{\mathrm{in},u} \in \mathbb{R}^{N_C}$ are the vectors of inlet concentrations of the respective streams, $\mathbf{r}_1 \in \mathbb{R}^{N_R}$ is the vector of reaction rates in the first reactor, and $\mathbf{S} \in \mathbb{R}^{N_C} \times \mathbb{R}^{N_R}$ is the stoichiometric matrix of the reaction system. The material balance of all subsequent reactors is

$$V_k \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c}_k = (F+u)(\mathbf{c}_{k-1} - \mathbf{c}_k) + V_k\mathbf{S}\mathbf{r}_k \quad \forall\, k \in \{2, \dots, K\}. \tag{B.6}$$

The outlet of the overall reactor is the outlet to the last tank in the series, $\mathbf{c}_K$, which is used to compute the pH as

$$\mathrm{pH} = -\log_{10} c_{K,H^+} \tag{B.7}$$

and the strong acid equivalent for this system is (Wright and Kravaris, 1991)

$$Y = (c_{K,H_2CO_3} + c_{K,HCO_3^-} + c_{K,CO_3^{2-}})\frac{2 + 10^{\mathrm{p}K_2 - \mathrm{pH}^{\mathrm{sp}}}}{1 + 10^{\mathrm{p}K_2 - \mathrm{pH}^{\mathrm{sp}}} + 10^{\mathrm{p}K_1 + \mathrm{p}K_2 - 2\mathrm{pH}^{\mathrm{sp}}}} - c_{K,Na^+} + c_{K,Cl^-} \tag{B.8}$$

where $\mathrm{pH}^{\mathrm{sp}}$ is the desired set point of the pH controller to be designed and $\mathrm{p}K_i = -\log_{10} K_i$, with K_i the ith dissociation constant of the carbonic acid.

The quasi-equilibrium assumption is invoked for the four reactions, so the material balances are subject to the algebraic constraints

$$K_h = \frac{c_{k,H_2CO_3}}{c_{k,CO_2}} \tag{B.9}$$

$$K_1 = \frac{c_{k,H^+}c_{k,HCO_3^-}}{c_{k,H_2CO_3}} \tag{B.10}$$

$$K_2 = \frac{c_{k,H^+}c_{k,CO_3^{2-}}}{c_{k,HCO_3^-}} \tag{B.11}$$

Table B.1
Parameters of the detailed mechanistic model of the pH case study. Equilibrium constants from Perrin (1982).

Parameter	Value	Units
K	3	(–)
V	5	m^3
K_h	$1.70 \cdot 10^{-3}$	(–)
K_1	$2.512 \cdot 10^{-4}$	(–)
K_2	$4.677 \cdot 10^{-11}$	(–)
K_w	$1 \cdot 10^{-14}$	(–)

$$K_w = c_{k,H^+}c_{k,OH^-} \tag{B.12}$$

for $k \in \{1, \dots, K\}$.

The detailed mechanistic model is a differential–algebraic system. The model is solved applying the null-space method (Kakhu and Pantelides, 2003). Consistent initial conditions on the algebraic states are computed by solving the system comprising the algebraic constraints and the auxiliary variables defined by the null vectors of the stoichiometric matrix. The same method is used to compute consistent values of the concentrations in the inlet streams. F is assumed to contain HCl with concentration 0.01 mol/L, while u contains NaOH with concentration 0.1 mol/L. Both streams are assumed to be saturated with CO_2 in atmospheric conditions: assuming a molar fraction $y_{CO_2} = 4.21 \cdot 10^{-4}$, Henry's law yields an equilibrium concentration $c^*_{CO_2} = H_{CO_2}Py_{CO_2} = 1.45 \cdot 10^{-5}$ mol/L for $H_{CO_2} = 0.034\,\mathrm{mol/(L \cdot bar)}$ and $P = 1.013\,25$ bar.

The simulation is initialized by setting $\mathbf{c}_k = \mathbf{c}^{\mathrm{in},F}\ \forall k \in \{1, \dots, K\}$. The flowrates F and u are scheduled in order to excite the system, inducing wide variations of outlet pH and Y. The model parameters used in the simulation are summarized in Table B.1.

Time profiles of F, u, pH, and Y are recorded with sampling time of 1 s, then corrupted as follows. Time profiles are downsampled to 6 observations per minute (i.e. the sampling time is 12 s). Noise variables with a dynamic structure are then summed to the variables. For the pH, a Gaussian variable is sampled from a distribution with zero mean and standard deviation equal to 0.1 pH units, then passed through an autoregressive filter with weight equal to 0.4, and the resulting dynamic variable is summed to the time profile of the pH. A similar procedure is used for the other variables, with the standard deviations for the Gaussian variables set such that the signal-to-noise ratio of the measurements (defined as ratio between the signal variance and noise variance) is 1 : 0.0075. The time profiles of F, u, pH, and Y that are obtained are available in the dataset used for data-driven model training, and are shown in Fig. B.1.

Appendix C. Mechanistic model of the catalyst deactivation process

The detailed mechanistic model of the catalytic reactor described in Section 4.2 and schematized in Fig. 8 is based on a perfectly mixed, non-isothermal reactor model. The model includes $N_C = 4$ species (A, B, R, and S) taking part to $N_R = 2$ reactions

$$A + B \xrightarrow{a} R \tag{C.1}$$

$$A + R \longrightarrow S \tag{C.2}$$

where $a \in [0, 1]$ is the relative catalyst activity, which affects only the primary reaction.

The material balances of the species in the reactor (in vector form) are

$$V\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{c} = F(\mathbf{c}^{\mathrm{in}} - \mathbf{c}) - V(a, 1)^{\top}(\mathbf{S}\mathbf{r}) \tag{C.3}$$

where V is the volume of the reactor, $\mathbf{c} \in \mathbb{R}^{N_C}$ is the vector of concentrations in the reactor, F is the feed flowrate, $\mathbf{c}^{\mathrm{in}} \in \mathbb{R}^{N_C}$ is the

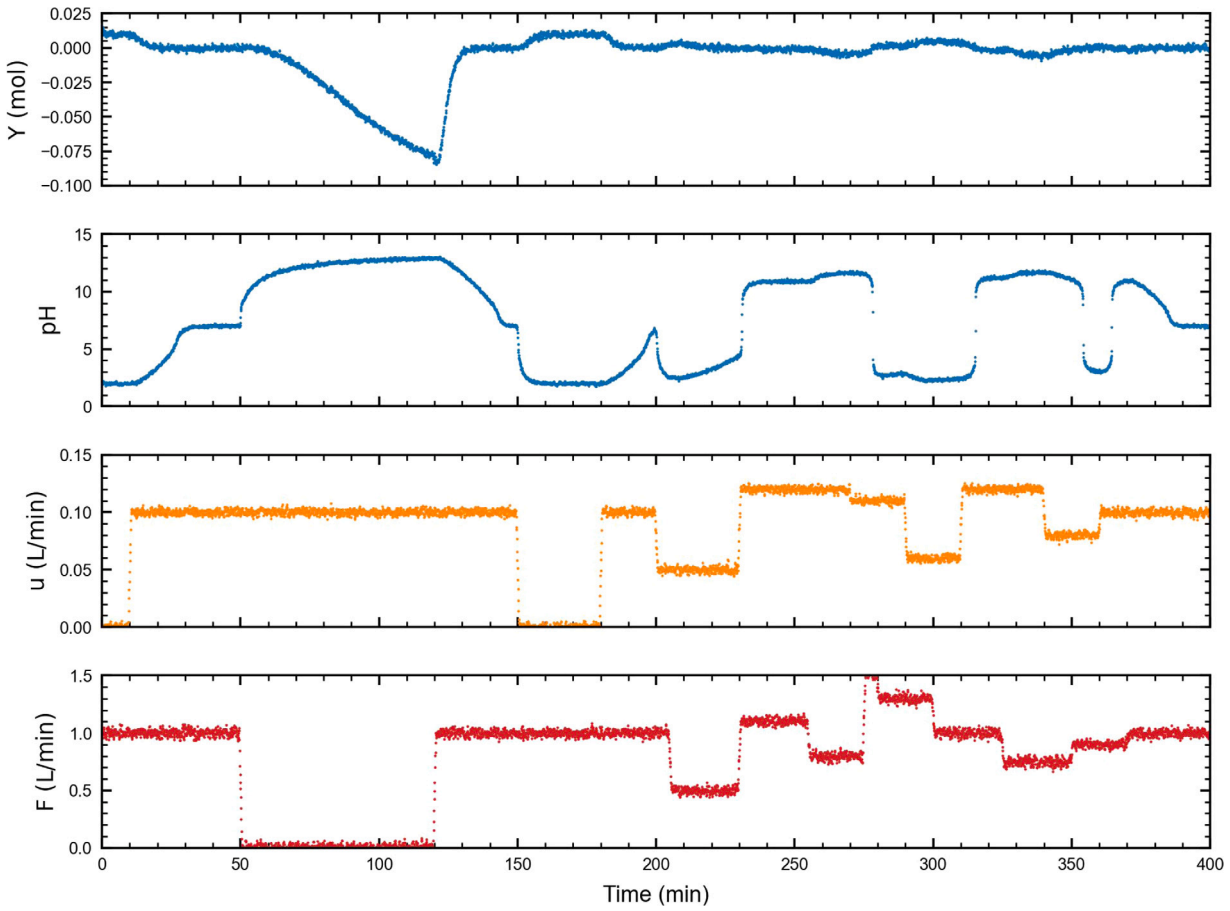

Fig. B.1. Available dataset for the pH case study.

vector of concentrations in the feed, a is the relative catalyst activity, $\mathbf{S} \in \mathbb{R}^{N_C} \times \mathbb{R}^{N_R}$ is the stoichiometric matrix of the reaction system, and $\mathbf{r} \in \mathbb{R}^{N_R}$ is the vector of reaction rates. The matrix $\mathbf{S}$ is defined as

$$\mathbf{S} = \begin{pmatrix} -1 & -1 \\ -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} \tag{C.4}$$

and the reaction rates are modeled by a mass action model with non-elementary kinetics, i.e. the partial orders of reaction do not match the stoichiometric coefficients of the species involved (Levenspiel, 1998)

$$\mathbf{r} = \begin{pmatrix} k_1 c_A^{o_{A,1}} c_B^{o_{B,1}} \\ k_2 c_A^{o_{A,2}} c_R^{o_{R,2}} \end{pmatrix} \tag{C.5}$$

where $o_{i,j}$ is the partial order of reaction of species i in reaction j. The effect of temperature on the reaction rates is modeled by the Arrhenius law (Levenspiel, 1998)

$$k_i = k_{0,j} e^{-\frac{\tilde{E}_{a,j}}{RT}}, \quad j \in \{1, 2\} \tag{C.6}$$

where $k_{0,j}$ and $\tilde{E}_{a,j}$ are the pre-exponential factor and activation energy, respectively, of reaction j, and $R = 8.314\,\mathrm{kJ/(kmol \cdot K)}$ is the universal gas constant. Both reactions are exothermic.

The energy balance of the reactor is

$$\rho_m \hat{C}_{P,m} V \frac{\mathrm{d}}{\mathrm{d}t} T = \rho_m \hat{C}_{P,m} F (T^{in} - T) - U A_j (T - T_j) + V a (-\boldsymbol{\Delta H}_r)^\top \mathbf{r} \tag{C.7}$$

and the energy balance of the reactor jacket is

$$\rho_w \hat{C}_{P,w} V_j \frac{\mathrm{d}}{\mathrm{d}t} T_j = \rho_w \hat{C}_{P,w} F_j (T_j^{in} - T_j) + U A_j (T - T_j) \tag{C.8}$$

where F_j is the flowrate for cooling water flowing through the jacket; T and T_j are the temperatures of the reactor mixture and cooling water, respectively; T^{in} and T_j^{in} are the temperatures of the feed to the reactor and to the jacket, respectively; ρ_m and $\hat{C}_{P,m}$ are the density and specific heat of the reaction mixture, respectively; ρ_w and $\hat{C}_{P,w}$ are the corresponding properties of the cooling water; V_j and A_j are the jacket volume and exchange surface, respectively; U is the global heat exchange coefficient of the system; and $\boldsymbol{\Delta H}_r \in \mathbb{R}^{N_R}$ is the vector of reaction enthalpies.

Catalyst deactivation is modeled by the generalized power law expression (Bartholomew, 1993; Forzatti and Lietti, 1999)

$$\frac{\mathrm{d}}{\mathrm{d}t} a = -F k_a e^{-\frac{\tilde{E}_a}{RT}} (a - a_{eq})^m \tag{C.9}$$

where k_a and $\tilde{E}_a$ are the parameters of the temperature model (analogous to the pre-exponential factor and activation energy, respectively, in the Arrhenius law), m is the deactivation order, and a_{eq} is the equilibrium catalyst activity (i.e. the activity at which the rates or deactivation and self-regeneration are balanced). The equilibrium value is set as a_{eq}. We also include the feed flowrate, F, in Eq. (C.9) to relate the catalyst activity to the time-on-flow (i.e. the amount of material processed) rather than to the pure time. The parameters of the model equations are summarized in Table C.1.

The model comprises 7 differential equations. The state variables are $\mathbf{c}$, T, T_j. The feed variables, i.e. F, $\mathbf{c}^{in}$, T^{in}, and T_j^{in}, are considered disturbances, meaning variables affecting the process but that cannot be manipulated. One output variable is considered in the model, i.e. the yield of R from A, defined as the amount of R produced divided by the amount of A consumed,

$$Y_{R/A} = \frac{c_R - c_R^{in}}{c_A^{in} - c_A} \tag{C.10}$$

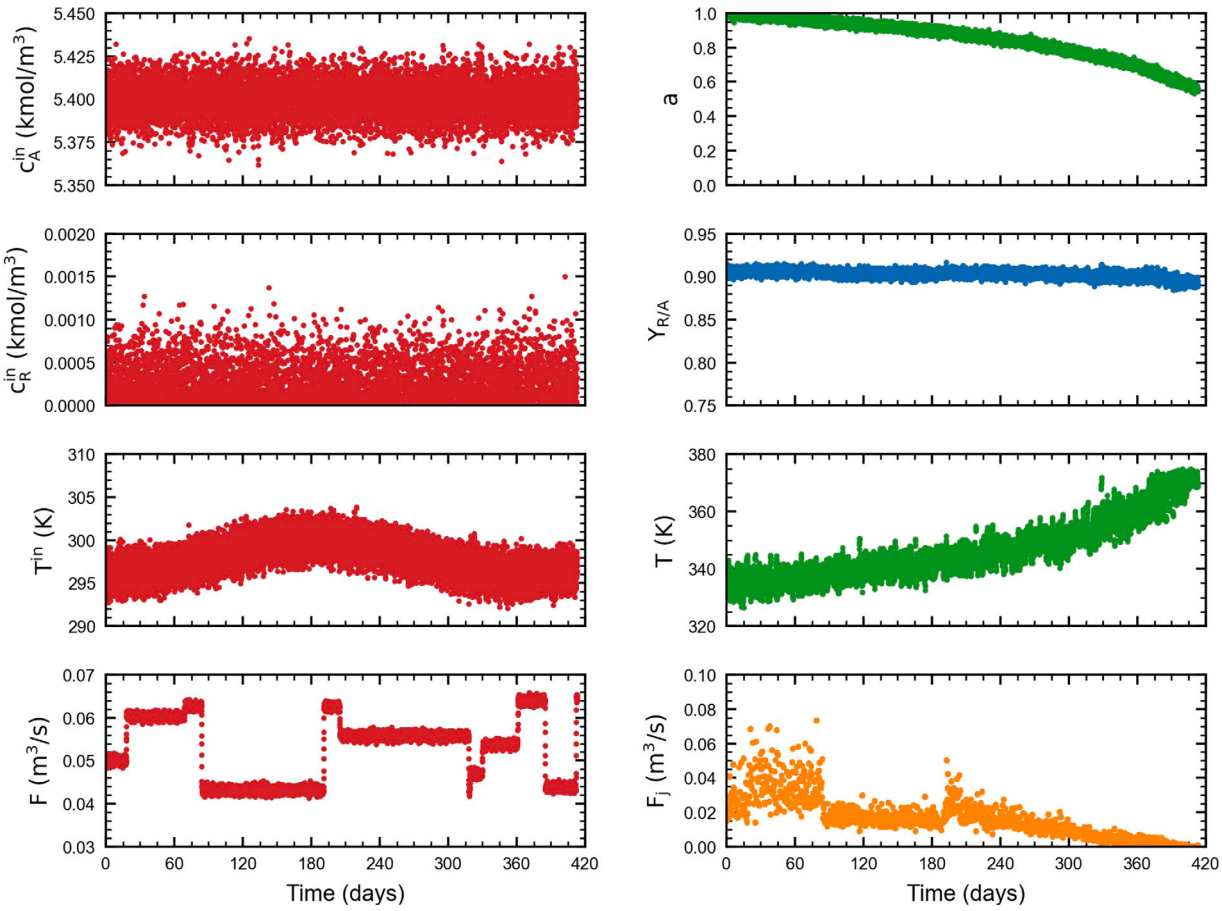

Fig. C.1. Selected variables from the training batch for the catalyst deactivation case study.

Table C.1
Parameters of the detailed mechanistic model of the catalyst deactivation case study.

Parameter	Value	Units
V	5	m^3
ρ_m	900	kg/m^3
$\hat{C}_{P,m}$	1.8	kJ/(kg · K)
V_j	0.1775	m^3
A_j	4.675	m^2
U	30	kW/(m^2·K)
ρ_w	1000	kg/m^3
$\hat{C}_{P,m}$	4.181	kJ/(kg · K)
$k_{0,1}$	$3.535 \cdot 10^9$	m^3/(kmol · s)
$\tilde{E}_{a,1}$	55 000	kJ/kmol
$o_{A,1}$	1.3	–
$o_{B,1}$	0.7	–
ΔH_{r_1}	−23000	kJ/kmol
$k_{0,2}$	$1.568 \cdot 10^5$	m^3/(kmol · s)
$\tilde{E}_{a,2}$	400 000	kJ/kmol
$o_{A,2}$	1.2	–
$o_{R,2}$	0.8	–
ΔH_{r_2}	−16000	kJ/kmol
k_a	$2.893 \cdot 10^4$	1/m^3
$\tilde{E}_a$	71 500	kJ/kmol
a_{eq}	0.15	–
m	1	–

The set point for the output is $Y^{sp}_{R/A} = 0.90$.

As only the main reaction is catalyzed, the yield of the desired product (R) decreases as the catalyst deactivates. The set-point can be maintained by controlling the temperature of the reactor, which is achieved by manipulating the control input F_j. The control system operates as follows. The composition of the mixture leaving the reactor is measured by means of an analyzer, then used as input to an imperfect model of the reactor (labeled as "control model" in Fig. 8). This model computes a control action (value of F_j), which is acquired by an operator, who in turn makes a subjective evaluation and implements the final control action on the plant. This chain of events is meant to simulate the operation of a real plant, where slow phenomena (such as catalyst deactivation) are typically managed by operators rather than by a fully automated control system.

Given that the main reaction is exothermic and that its rate decreases as the catalyst deactivates, the thermal generation of such reaction decreases with a as well. This implies that F_j is reduced along the process. The simulator is designed in such a way that F_j hits $0\,m^3/s$ when $a \simeq 0.6$. After that point, the reactor operates at its adiabatic temperature, which cannot be controlled, thus the yield cannot be maintained at the desired set-point and drops with the catalyst activity. While an industrial process would be interrupted shortly after the termination of the "in-control phase" for catalyst maintenance, the "out-of-control phase" still provides important information about the physics of the process (in a real scenario, the out-of-control phase could be still carried out on a small-scale process, e.g. pilot or laboratory scale, if there is interest in exploring the full range of the catalyst activity). Therefore, only data from the in-control phase are used for model training, while data from both phases are used to enforce the mechanistic constraints (i.e. as collocation points) and to evaluate the extrapolation capabilities of the model.

Time profiles of disturbance variables are designed to yield a realistic process behavior. Such variables undergo "structured variation" (i.e. defined schedules) to simulated plausible average trends, and are further perturbed with "process noise" (i.e. random variables with built-in

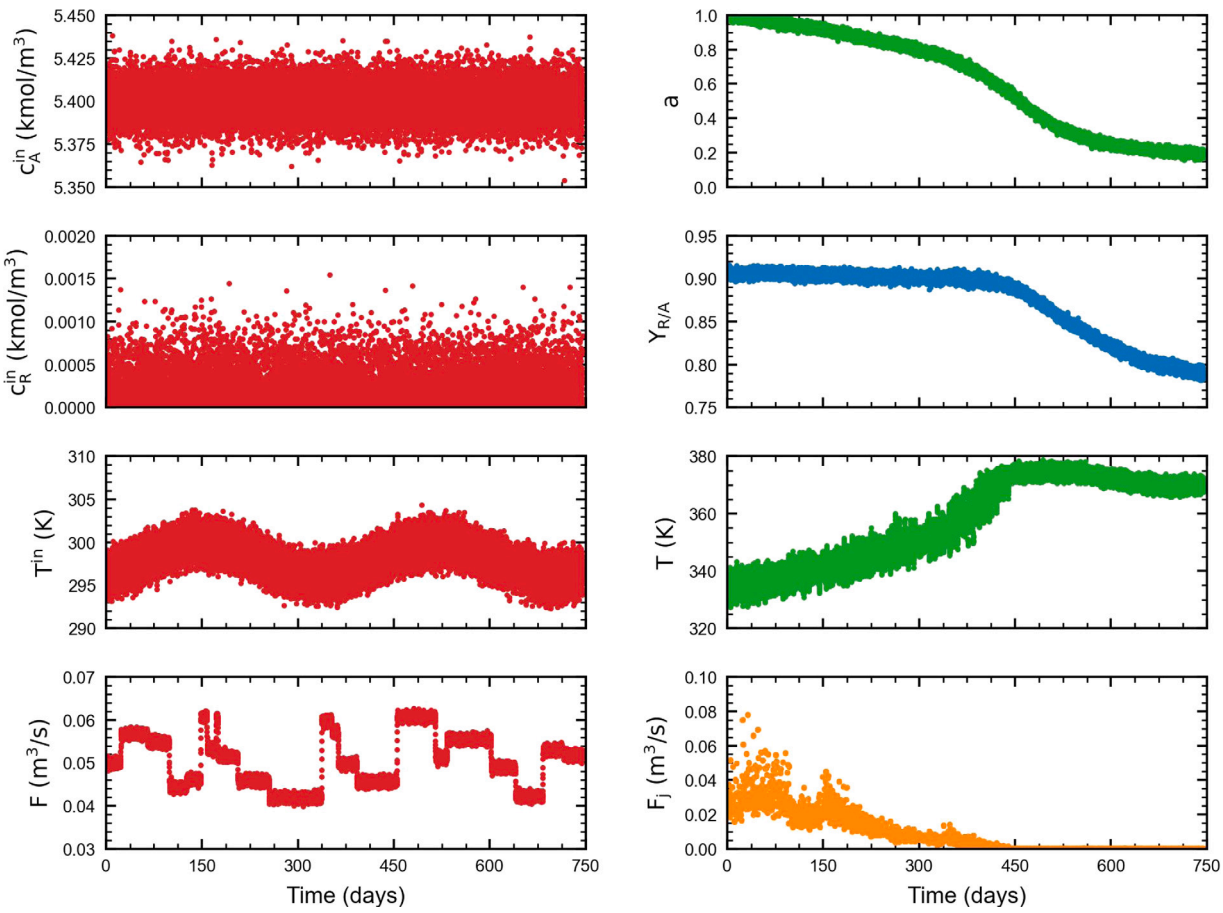

Fig. C.2. Selected variables from the testing batch for the catalyst deactivation case study.

dynamics) to simulate random disturbances from the environment of hypothetical process units preceding the reactor.

The feed flowrate is scheduled as a piecewise-constant profile to simulate varying production demands. The number of levels and the percentage perturbations are sampled from univariate uniform distributions. Feed temperatures are scheduled with a modulated sinusoidal signal: a "slow" sine with period of 365 d simulates seasonal disturbances due to the average daily temperature, while a "fast" sine with period of 1 d mimics day-to-night oscillations around the average temperature. Feed concentrations are set to predefined levels (i.e. $c_A^{in} = 5.4\,\mathrm{kmol/m^3}$, $c_B^{in} = 6\,\mathrm{kmol/m^3}$, $c_R^{in} = 0\,\mathrm{kmol/m^3}$, and $c_S^{in} = 0\,\mathrm{kmol/m^3}$). Process noise variables (summed to the schedules described above) are obtained by sampling Gaussian variables with zero means and given standard deviations, then passing them through autoregressive filters of varying orders: 4th order for F, 6th order for $\mathbf{c}^{in}$, 4th order for T^{in}, and 2nd order for T_j^{in}.

The model is simulated as follows. At the initial time point, the states are set to given initial values, and disturbances are acquired based on the generated profiles. The "true" values are saved for model integration, then corrupted with measurement noise variables (from Gaussian distributions) and recorded in the dataset as "measured" variables. The output variable is computed using the corrupted states. To compute the control action (value of F_j) measured variables are fed to a "control model", which is a steady state version of Eqs. (C.3) through (C.8) (i.e. all derivate terms are set to 0) with elementary kinetics assumed (i.e. partial orders of reaction in Eq. (C.5) are all set to 1). The control model is solved to obtain F_j. Such value is the corrupted with a strong (Gaussian) perturbation to simulated operator error, saved for model integration, and finally corrupted with further measurement noise prior to be saved in the dataset.

After the initial stage described above, the model equations are integrated for 1 h. The acquisition, corruption, and saving of states, disturbances, and outputs is repeated, but the control action is left unchanged (it is only corrupted with a new noise variable prior to saving it in the dataset). After 8 h of integration, the control action is re-computed and updated as described above. Once the catalyst activity has decreased enough and the process goes out of control, F_j is fixed to $0\,\mathrm{m^3/s}$ and the process runs in open loop.

Along the integration, all true states, disturbances, and control actions are clipped to 0 each time they are acquired to guarantee physically meaningful results. Coherently, the integration is performed with a numerical method that constrains the solution to positive values only (Shampine et al., 2005). Measured concentrations are also clipped to 0, but measured flowrates are not.

Seven "batches" are generated. Each batch features a different schedule of disturbance variables and initial conditions for the states, most notable of the catalyst activity. Three batches are intended for data-driven model training and include the in-control phase only. One batch is used as collocation points and includes both the in-control and out-of-control phases. Three more batches are used for testing and also include both process phases. All batches are included in the dataset provided with the code to reproduce the case studies (see Data and Code Availability). However, as mentioned in Section 4.2, the results discussed in this article use only one training batch and one testing batch (besides the collocation points batch). Such batches are identified as by batch numbers 1 and 5 in the full dataset (the collocation points batch is number 4). Time profiles of selected variables from the training and testing batches are reported, respectively, in Figs. C.1 and C.2.

As mentioned in Section 4.2, PGNN modeling is carried out under the steady state assumption for all model states. Therefore, states and

outputs need to be aligned with the control action by shifting them backwards by one sampling time (i.e. 1 h). This is due to the data generation mechanism outline above. States and disturbances are acquired at time t and used to compute the control action, which is in turn implemented in the plant. States then evolve under the implemented control action and current disturbances (assumed constant over the sampling interval), until eventually achieving a new steady state, that will be recorded at time $t+1$. Back-shifting the states (and outputs) thus guarantees that the observations in the data table satisfy steady state equations of the simplified mechanistic model used in Section 4.2. The alignment operation is performed on all batches in the dataset prior to modeling. Missing observations (due to the shifting) are removed.

Data availability

The code used to obtain the results discussed in this article and data for each one of the case studies discussed herein are available at the following GitHub repository: https://github.com/EliaAF/QualInfoHybridModelingPGNN.

References

Abramowitz, M., Stegun, I.A., 1965. Handbook of Mathematical Functions. Dover Publications.

Arnese-Feffin, E., Facco, P., Turati, D., Bezzo, F., Barolo, M., 2024a. Hybrid modeling of a biorefinery separation process to monitor short-term and long-term membrane fouling. Chem. Eng. Sci. 283, 119413. http://dx.doi.org/10.1016/j.ces.2023.119413.

Arnese-Feffin, E., Facco, P., Turati, D., Bezzo, F., Barolo, M., 2024b. Understanding fouling in an industrial biorefinery membrane separation process by feature-oriented data-driven modeling. Ind. Eng. Chem. Res. 63, 9136–9150. http://dx.doi.org/10.1021/acs.iecr.4c00590.

Aykol, M., Gopal, C.B., Anapolsky, A., Herring, P.K., van Vlijmen, B., Berliner, M.D., Bazant, M.Z., Braatz, R.D., Chueh, W.C., Storey, B.D., 2021. Perspective–combining physics and machine learning to predict battery lifetime. J. Electrochem. Soc. 168, 030525. http://dx.doi.org/10.1149/1945-7111/abec55.

Bartholomew, C.H., 1993. Sintering kinetics of supported metals: New perspectives from a unifying GPLE treatment. Appl. Catal. A: Gen. 107, 1–57. http://dx.doi.org/10.1016/0926-860X(93)85114-5.

Beucler, T., Pritchard, M., Rasp, S., Ott, J., Baldi, P., Gentine, P., 2021. Enforcing analytic constraints in neural-networks emulating physical systems. Phys. Rev. Lett. 126, 098302. http://dx.doi.org/10.1103/PhysRevLett.126.098302.

Bikmukhametov, T., Jäschke, J., 2020. Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models. Comput. Chem. Eng. 138, 106834. http://dx.doi.org/10.1016/j.compchemeng.2020.106834.

Bird, R.B., Stewart, W.E., Lightfoot, N., 2006. Transport Phenomena, second ed. Wiley.

Bishop, C.M., 2006. Pattern Recognition and Machine Learning. Springer.

Bishop, C.M., Bishop, H., 2024. Deep Learning: Foundation and Concepts. Springer.

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.

Bui, L., Joswiak, M., Castillo, I., Phillips, A., Yang, J., Hickman, D., 2022. A hybrid modeling approach for catalyst monitoring and lifetime prediction. ACS Eng. Au 2, 17–26. http://dx.doi.org/10.1021/acsengineeringau.1c00015.

Chiang, L.H., Braatz, R.D., 2003. Process monitoring using causal map and multivariate statistics: fault detection and identification. Chemometr. Intell. Lab. Syst. 65, 159–178. http://dx.doi.org/10.1016/S0169-7439(02)00140-5.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems 2, 303–314. http://dx.doi.org/10.1007/BF02551274.

Daw, A., Karpatne, A., Watkins, W., Read, J., Kumar, V., 2021. Physics-guided neural networks (PGNN): An application in lake temperature modeling. arXiv arXiv:1710.11431v3.

Destro, F., Facco, P., García-Muñoz, S., Bezzo, F., Barolo, M., 2020. A hybrid framework for process monitoring: Enhancing data-driven methodologies with state and parameter estimation. J. Process Control 92, 333–351. http://dx.doi.org/10.1016/j.jprocont.2020.06.002.

Eivazi, H., Tahani, M., Schlatter, P., Vinuesa, R., 2022. Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. Phys. Fluids 34, 075117. http://dx.doi.org/10.1063/5.0095270.

Forzatti, P., Lietti, L., 1999. Catalyst deactivation. Catal. Today 52, 165–181. http://dx.doi.org/10.1016/S0920-5861(99)00074-7.

Hastie, T., Tibshirani, R., Friedman, J.H., 2009. The Elements of Statistical Learning, second ed. Springer.

Hoerl, A.E., Kennard, R.W., 1970. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics 12, 55–67. http://dx.doi.org/10.1080/00401706.1970.10488634.

Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., Kumar, V., 2019. Physics guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles. arXiv arXiv:1810.13075v2.

Kakhu, A.I., Pantelides, C.C., 2003. Dynamic modelling of aqueous electrolyte systems. Comput. Chem. Eng. 27, 869–882. http://dx.doi.org/10.1016/S0098-1354(03)00002-4.

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

Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., Kumar, V., 2017a. Theory-guided data science: A new paradigm for scientific discovery from data. IEEE Trans. Knowl. Data Eng. 29, 2318–2331. http://dx.doi.org/10.1109/TKDE.2017.2720168.

Karpatne, A., Jia, X., Kumar, V., 2024. Knowledge-guided machine learning: Current trends and future prospects. arXiv arXiv:2403.15989v2.

Karpatne, A., Watkins, W., Read, J., Kumar, V., 2017b. Physics-guided neural networks (PGNN): An application in lake temperature modeling. arXiv arXiv:1710.11431v1.

Kim, K.-K.K., Ríos-Patrón, E., Braatz, R.D., 2012. Robust nonlinear internal model control of stable Wiener systems. J. Process Control 22, 1468–1477. http://dx.doi.org/10.1016/j.jprocont.2012.01.019.

Koksal, E.S., Aydin, E., 2023. Physics informed piecewise linear neural networks for process optimization. Comput. Chem. Eng. 174, 108244. http://dx.doi.org/10.1016/j.compchemeng.2023.108244.

Koric, S., Abueidda, D.W., 2023. Data-driven and physics-informed deep learning operators for solution of heat conduction equation with parametric heat source. Int. J. Heat Mass Transfer 203, 123809. http://dx.doi.org/10.1016/j.ijheatmasstransfer.2022.123809.

Lagerquist, R., Turner, D., Ebert-Uphoff, I., Stewart, J., Hagety, V., 2021. Using deep learning to emulate and accelerate a radiative transfer model. J. Atmos. Ocean. Technol. 38, 1637–1696. http://dx.doi.org/10.1175/JTECH-D-21-0007.1.

Levenspiel, O., 1998. Chemical Reaction Engineering, third ed. Wiley.

Marques, R., von Stosch, M., Portela, R.M.C., Torres, C.A.V., Antunes, S., Freitas, F., Reis, M.A.M., Oliveira, R., 2017. Hybrid modeling of microbial exopolysaccharide (EPS) production: The case of *Enterobacter* A47. J. Biotech. 246, 61–70. http://dx.doi.org/10.1016/j.jbiotec.2017.01.017.

Masi, F., Stefanou, I., Vannucci, P., Maffi-Berthier, V., 2021. Thermodynamics-based artificial neural networks for constitutive modeling. J. Mech. Phys. Solids 147, 104277. http://dx.doi.org/10.1016/j.jmps.2020.104277.

Mowbray, M., Vallerio, M., Perez-Galvan, C., Zhang, D., Del Rio Chanona, A., Navarro-Brull, F.J., 2022. Industrial data science – a review of machine learning applications for chemical and process industries. React. Chem. Eng. 7, 1471–1509. http://dx.doi.org/10.1039/d1re00541c.

Mukherjee, A., Bhattacharyya, D., 2025. Development of mass, energy, and thermodynamics constrained steady-state and dynamic neural networks for interconnected chemical systems. Chem. Eng. Sci. 309, 121506. http://dx.doi.org/10.1016/j.ces.2025.121506.

Narayanan, H., Sokolov, M., Morbidelli, M., Butté, A., 2019. A new generation of predictive models: The added value of hybrid models for manufacturing processes of therapeutic proteins. Biotechnol. Bioeng. 116, 2540–2549. http://dx.doi.org/10.1002/bit.27097.

Nocadel, J., Wright, S.J., 2006. Numerical Optimization. Springer.

Ohara, M., Reid, R.C., 1973. Modeling Crystal Growth Rates from Solution. Prentice-Hall.

Oliveira, R., 2004. Combining first principles modelling and artificial neural networks: A general framework. Comput. Chem. Eng. 28, 766–775. http://dx.doi.org/10.1016/j.compchemeng.2004.02.014.

Paredes, R., Rato, T.J., Reis, M.S., 2023. Causal network inference and functional decomposition for decentralized statistical process monitoring: Detection and diagnosis. Chem. Eng. Sci. 267, 118338. http://dx.doi.org/10.1016/j.ces.2022.118338.

Perrin, D.D., 1982. Dissociation Constants of Inorganic Acids and Bases in Aqueous Solution, second ed. De Gruyter.

Picabea, J., Maestri, M., Cassanello, M., Horowitz, G., 2021. Hybrid model for fault detection and diagnosis in an industrial distillation column. Chem. Prod. Process. Model. 16, 169–180. http://dx.doi.org/10.1515/cppm-2020-0004.

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

Pukrittayakamee, A., Malshe, M., Hagan, N., Raff, L.M., Narulkar, R., Bukkapanum, R., Komanduri, R., 2009. Simultaneous fitting of a potential-energy surface and its corresponding force fields using feedforward neural networks. J. Chem. Phys. 130, 134101. http://dx.doi.org/10.1063/1.3095491.

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.

Reis, M.S., Gins, G., Rato, T.J., 2019. Incorporation of process-specific structure in statistical process monitoring: A review. J. Qual. Technol. 51, 407–421. http://dx.doi.org/10.1080/00224065.2019.1569954.

Reis, M.S., Saraiva, P.M., 2021. Data-centric process systems engineering: A push towards PSE 4.0. Comput. Chem. Eng. 155, 107529. http://dx.doi.org/10.1016/j.compchemeng.2021.107529.

Rendall, R., Chiang, L.H., Reis, M.S., 2019. Data-driven methods for batch data analysis – A critical overview and mapping on the complexity scale. Comput. Chem. Eng. 124, 1–13. http://dx.doi.org/10.1016/j.compchemeng.2019.01.014.

Rogers, A.W., Song, Z., Vega Ramon, F., Jing, K., Zhang, D., 2023. Investigating 'greyness' of hybrid model for bioprocess predictive modelling. Biochem. Eng. J. 190, 108761. http://dx.doi.org/10.1016/j.bej.2022.108761.

Romagnoli, J.A., Briceno-Mena, L.A., Manee, V., 2025. AI in Chemical Engineering. CRC Press.

von Rueden, L., Mayer, S., Beckh, K., Georgiev, B., Giesselbach, S., Heese, R., Kirsch, B., Pfrommer, J., Pick, A., Ramamurthy, R., Walczak, M., Garcke, J., Bauckhage, C., Schuecker, J., 2023. Informed machine learning - A taxonomy and survey of integrating prior knowledge into learning systems. IEEE Trans. Knowl. Data Eng. 35, 614–633. http://dx.doi.org/10.1109/TKDE.2021.3079836.

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.

Sansana, J., Rendall, R., Castillo, I., de Brujine, L., Huggins, J., Phillips, A., Reis, M.S., 2024. Hybrid approach for advanced monitoring and forecasting of fouling with application to an ethylene oxide plant. Ind. Eng. Chem. Res. 63, 10666–10676. http://dx.doi.org/10.1021/acs.iecr.4c00298.

Schmidt, A., Wallace, K. (Eds.), 2024. The Digital Transformation of Product Formulation. CRC Press.

Schubert, J., Simutis, R., Dors, M., Ivo, H., Lübbert, A., 1994. Hybrid modelling of yeast production process. Chem. Eng. Technol. 17, 10–20, 1994.

Schweidtmann, A.M., Zhang, D., von Stosch, M., 2024. A review and perspective on hybrid modeling methodologies. Digit. Chem. Eng. 100, 100136. http://dx.doi.org/10.1016/j.dche.2023.100136.

Severson, K.A., Attia, P.M., Jin, N., Perkins, N., Jiang, B., Yang, Z., Chen, M.H., Aykol, M., Herring, P.K., Fraggedakis, D., Bazant, M.Z., Harris, S.J., Chueh, W.C., Braatz, R.D., 2019. Data-driven prediction of battery cycle life before capacity degradation. Nat. Energy 4, 383–391. http://dx.doi.org/10.1038/s41560-019-0356-8.

Shah, P., Pahari, S., Bhavsar, R., Kwon, J.S.-I., 2025. Hybrid modeling of first-principles and machine learning: A step-by-step tutorial review for practical implementation. Comput. Chem. Eng. 194, 108926. http://dx.doi.org/10.1016/j.compchemeng.2024.108926.

Shakor, Z.M., Al-Shafei, E.N., 2023. The mathematical catalyst deactivation models: a mini review. RSC Adv. 13, 22579–22592. http://dx.doi.org/10.1039/D3RA02912C.

Shampine, L.F., Thompson, S., Kierzenka, J.A., Byrne, G.D., 2005. Non-negative solutions of ODEs. Appl. Math. Comput. 170, 556–569. http://dx.doi.org/10.1016/j.amc.2004.12.011.

Sharma, N., Liu, Y.A., 2022. A hybrid science-guided machine learning approach for modeling chemical processes: A review. AIChE J. 68, e17609. http://dx.doi.org/10.1002/aic.17609.

van Sprang, E.N.M., Ramaker, H.J., Westerhuis, J.A., Smilde, A.K., Wienke, D., 2005. Statistical batch process monitoring using gray models. AIChE J. 51, 931–945. http://dx.doi.org/10.1002/aic.10348.

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

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

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. J. R. Stat. Soc. Ser. B Stat. Methodol. 58, 267–288. http://dx.doi.org/10.1111/j.2517-6161.1996.tb02080.x.

Vedam, H., Venkatasubramanian, V., 1999. PCA-SDG based process monitoring and fault diagnosis. Control Eng. Pract. 7, 903–917. http://dx.doi.org/10.1016/S0967-0661(99)00040-4.

Venkatasubramanian, V., 2018. The promise of artificial intelligence in chemical engineering: Is it here, finally? AIChE J. 65, 466–478. http://dx.doi.org/10.1002/aic.16489.

Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V., 2022. Integrating scientific knowledge with machine learning for engineering and environmental systems. arXiv arXiv:2003.04919v6.

Wold, S., Kettaneh-Wold, N., MacGregor, J.F., Dunn, K.G., 2009. Batch process modeling and MSPC, first ed. In: Comprehensive Chemometrics, vol. 2, Elsevier, pp. 136–197.

Wright, R.A., Kravaris, C., 1991. Nonlinear control of pH processes using the strong acid equivalent. Ind. Eng. Chem. Res. 30, 1561–1572. http://dx.doi.org/10.1021/ie00055a022.

Wu, G., Yion, W.T.G., Dang, K.L.N.Q., Wu, Z., 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, W.-T., Jakey, B., Roussy, A., Pinaton, J., Reis, M.S., 2020a. A physics-informed run-to-run control framework for semiconductor manufacturing. Expert Syst. Appl. 155, 113424. http://dx.doi.org/10.1016/j.eswa.2020.113424.

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

Yang, X.I.A., Zafar, S., Wang, J.-X., Xiao, H., 2019. Predictive large-eddy-simulation wall modeling via physics-informed neural networks. Phys. Rev. Fluids 4, 034602. http://dx.doi.org/10.1103/PhysRevFluids.4.034602.

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. J. R. Stat. Soc. Ser. B Stat. Methodol. 67, 301—-320. http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x.