Computers and Chemical Engineering 166 (2022) 107898

Contents lists available at ScienceDirect

Computers and Chemical Engineering

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

Review

Perspectives on the integration between first-principles and data-driven modeling

William Bradley [a,1], Jinhyeun Kim [a,1], Zachary Kilwein [a,1], Logan Blakely [b], Michael Eydenberg [b], Jordan Jalvin [b], Carl Laird [b], Fani Boukouvala [a,*]

[a] School of Chemical & Biomolecular Eng., Georgia Institute of Technology, Atlanta, GA 30332, United States
[b] Sandia National Laboratories, Albuquerque, NM 87185, United States

ARTICLE INFO

Keywords:
Hybrid modeling
Model calibration
Physics-informed machine learning
Dynamical systems
Process systems design/operations

ABSTRACT

Efficiently embedding and/or integrating mechanistic information with data-driven models is essential if it is desired to simultaneously take advantage of both engineering principles and data-science. The opportunity for hybridization occurs in many scenarios, such as the development of a faster model of an accurate high-fidelity computer model; the correction of a mechanistic model that does not fully-capture the physical phenomena of the system; or the integration of a data-driven component approximating an unknown correlation within a mechanistic model. At the same time, different techniques have been proposed and applied in different literatures to achieve this hybridization, such as hybrid modeling, physics-informed Machine Learning (ML) and model calibration. In this paper we review the methods, challenges, applications and algorithms of these three research areas and discuss them in the context of the different hybridization scenarios. Moreover, we provide a comprehensive comparison of the hybridization techniques with respect to their differences and similarities, as well as advantages and limitations and future perspectives. Finally, we apply and illustrate hybrid modeling, physics-informed ML and model calibration via a chemical reactor case study.

1. Introduction

Recent developments in the broad field of data-science have led to a series of breakthroughs in Machine Learning (ML) (Qin and Chiang, 2019; Lee et al., 2018) techniques. The Process Systems Engineering (PSE) community is having an important debate on the roles data science should have over first-principles, physics-based science (e.g., thermodynamics, transport phenomena, kinetics and mass balances) (Venkatasubramanian, 2019). The key reasons for doubting the value of ML in chemical engineering is their black-box nature, a term generally used to acknowledge their poor extrapolating capabilities, lack of interpretability. and unbounded uncertainty in predictions that may not satisfy physical constraints. “Hybridization” holds the promise that the data-dependent models are more reliable because they learn from both data and physics (von Stosch et al. 2014b). The concept of Hybrid-Modeling is not new to PSE. Indeed, hybrid modeling techniques and applications have been growing in number since the early 90′s (Psichogios and Ungar, 1992; Rico-Martínez et al., 1992; Thompson and Kramer, 1994), especially for modeling dynamic systems common to PSE. Following an explosion of advances in purely black-box ML techniques, merging first-principles knowledge and ML is becoming the next big trend, since it can increase accuracy and interpretability with less data.

When reviewing all potential ways to merge data-driven with physics-based models, a diverse set of approaches have been proposed in different literatures, ranging from feature engineering (i.e., careful selection of the inputs/outputs to a model), to simple additive models comprised of separate physical and data-driven equations, to advanced methods that embed physics within data-driven models using customized training and numerical techniques. In fact, concepts targeting the same fundamental idea are often disguised under different terminology, in different literatures and time periods. A helpful resource for distinguishing the nuances between like-minded ideas is a recent review (Sansana et al., 2021), which offers an encyclopedic comparison of HM approaches.

Though techniques for linking mechanistic and data-driven tools are

* Corresponding author.
E-mail address: fani.boukouvala@chbe.gatech.edu (F. Boukouvala).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.compchemeng.2022.107898
Received 10 November 2021; Received in revised form 27 May 2022; Accepted 20 June 2022
Available online 22 June 2022
0098-1354/

as numerous as the systems they model, some methods shown promise across multiple disciplines. In particular, three areas are reviewed in this paper are: (a) "Hybrid Submodeling" (HSM), (b) "Physics-Informed ML" (PI-ML), and (c) "Model calibration" (MC). Specifically, HSM contributions focus on identifying unknown or partially-known process mechanisms constrained by known first-principles equations (typically, a system of differential equations representing conservation balances) (von Stosch et al. 2014b). PI-ML is in many ways a synonym to HSM, but typically refers to techniques for constrained training of deep ML models (e.g., adding physics-based loss terms, or changing weight parameters or the ML model structure) based on prior knowledge (Chen et al., 2018; Raissi et al., 2019). PI-ML and HSM techniques aim to maintain the advantages inherent to data driven models (low computational cost), while making models more generalizable and physically consistent. MC is the process of updating the parameters of physics-based models with statistical techniques (typically Bayesian methods) to compensate for the model-data discrepancy (Kennedy and O'Hagan, 2001b). Although these terms are widely used in the PSE, ML, and Statistics and Operations Research literatures, respectively, these methods offer many similarities and potential synergies that we hope to elucidate.

In this perspectives paper we first review the basic principles, methods, algorithms and applications of HSM, PI-ML and MC. Most importantly, we have compiled the similarities and differences in the above areas and provide a comprehensive discussion on the challenges and limitations of each approach, potential synergies and future perspectives. Finally, we present the capabilities of each approach on a reactor modeling case study.

The sections of this paper are structured as follows. In Data-Driven Models (Section 2), we discuss purely data-driven approaches for modeling input-output data and focus on two popular techniques, namely Neural Networks and Gaussian Process Models that will be used in this work. Next, in Merging Data-Driven with First-Principles Models (Section 3) we define First-Principles models and discuss potential scenarios for hybridization, as well as methods in HM, PI-ML and MC. In the Applications section (Section 4) we review areas where HM, PI-ML and MC have been applied, and in Software Implementations (Section 5) we provide a list of algorithms and software available for each of the techniques so far. Following this, this paper applies HM, PI-ML and MC concepts on a case study for a chemical reactor (Section 6), ending with a Perspectives discussion and Conclusions (Sections 7 and 8, respectively).

2. Data-driven models

Purely data-driven models differ from first principles models in that their parameters are fitted based on available data and often their parameters do not have a physically interpretable meaning. Thus, data-driven models are often referred to as "black-box" or empirical models and are primarily used for extracting correlations from data. The data may come from designed physical experiments, historical databases, or designed samples from mechanistic models or simulations, or any combination of the above sources. One popular framework involving a data-driven model is to replace a mechanistic model with a data-driven surrogate (or emulator or metamodel). This surrogate model is trained using data simulated from the mechanistic model. Surrogate modeling is primarily motivated by the reduced computation time of the surrogate model, which can be used to accelerate time-sensitive tasks such as optimization, monitoring or control. Multiple reviews have been written on this topic within chemical engineering alone (McBride and Sundmacher, 2019; Bhosekar and Ierapetritou, 2018), and the reader is encouraged to consult these for specific examples. Note, however, that surrogate modeling is not considered a hybrid or physics-informed modeling technique as mechanistic information is not incorporated into the model training or simulation. However, as done in Schäfer et al. (2019), the surrogate model is free to be merged with mechanistic models at the time of application.

Numerous data-driven models exist and can range in complexity from generalized linear regression (e.g., linear and nonlinear terms that maintain linearity in parameters) to universal nonlinear approximators (e.g., Gaussian process, Neural Networks, Random Forests) and many more. A general distinction between "parametric" and "nonparametric" models can be found in the literature and will be adopted in this work. A data-driven model is characterized as parametric if its parameters associated with their accompanying terms, have some type of physical interpretation. For example, a generalized linear regression model with linear and quadratic terms can be considered as a parametric model since the parameters associated with each term signify the importance of linear and quadratic effects on the output prediction. On the other hand, parameters in nonparametric models are not associated with terms that have any physical meaning, such as the weights and biases of the nodes of a Neural Network.

This review focuses on only a subset of nonparametric data-driven models, namely Neural Networks (NN) and Gaussian Process (GP) models and explores how these can be merged with mechanistic knowledge. The selection of these two techniques for this review is based on the fact that they are universal approximators, which implies that they are flexible and generalizable to be merged with physics in different settings and applications. This is supported by the fact that NNs and GPs are by far the most popular data-driven models in the areas reviewed in this paper. Moreover, it can be argued that parametric models (i.e., generalized linear regression models) are in certain cases physics-informed, if *a-priori* knowledge regarding the input-output relationships is used to specify which features to include. However, this type of hybridization is outside the scope of this review, and will only be mentioned briefly. On the other hand, the non-parametric modeling NNs and GPs are less restricted and can be modeled as black-boxes, or purely data-driven. For completeness, a brief overview of the NNs and GPs is provided below.

2.1. Gaussian process models

Gaussian Process (GP) modeling is a powerful tool to model probability over functions under the Bayesian framework (Rasmussen and Williams, 2005). Due to its flexibility, a GP model can approximate an arbitrary continuous function (Karniadakis 2020b) and capture nonlinear dependencies between inputs (Gorbach et al., 2017). In addition, its probabilistic nature enables the incorporation of different sources of uncertainty (e.g., parameter uncertainty and experimental uncertainty) as part of the model (Higdon et al., 2004). Gaussian Process Regression (GPR) models the relationship (f) between the inputs $X = \{x^{(i)}\}_{i=1}^{N}$ and the outputs $Y = (y^{(1)}, y^{(2)}, \ldots, y^{(N)})^T$. Here, X is a collection of N sampled inputs $x^{(i)}$ and Y is a collection of the corresponding N observed responses $y^{(i)}$ where $i = 1, 2, \ldots, N$. GPR models the mapping $f : X \rightarrow Y$. Each $x^{(i)}$ is a p-dimensional vector that contains the input variables of the system, $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)})^T$, and we assume that each observation $y^{(i)}$ is scalar for simplicity. GPR is a generalization of a multivariate Gaussian distribution over functions (Rasmussen and Williams, 2005), where for any finite selection of points $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$, the observation vector $Y = (Y(x^{(1)}), Y(x^{(2)}), \ldots, Y(x^{(N)}))^T = (y^{(1)}, y^{(2)}, \ldots, y^{(N)})^T$ follows the multivariate Gaussian distribution (MacKay, 2003). For noisy observations Y, independent and identically distributed (i.i.d) Gaussian noise $\varepsilon \sim N(0, \sigma_n^2)$ can be introduced to capture the measurement noise: $Y = f(X) + \varepsilon$. GPR starts by assuming a prior distribution over function f that we wish to learn as follows:

$$p(f|X) \sim N(m(X), \Sigma(X, X)) \tag{1}$$

In Eq. (1), each element of the kernel matrix $\Sigma(X, X)$ describes the covariance between two outputs sampled in X. If the popular squared exponential (SE) kernel is used, $(i, j)^{th}$ element of $N \times N$ covariance

matrix $\Sigma(X,X)$ is calculated as (Eq. (2)):

$$\Sigma_{i,j}(X,X) = cov\left(f\left(x^{(i)}\right),f\left(x^{(j)}\right)\right) = k\left(x^{(i)},x^{(j)}\right) = \tau^2\exp\left(-\frac{1}{2}\sum_{q=1}^{p} w_q\left(x_q^{(i)} - x_q^{(j)}\right)^2\right) + \sigma_n^2\delta_{ij} \tag{2}$$

where δ_{ij} is a Kronecker delta ($\delta_{ij} = 1$ iff $i = j$, otherwise 0). We call $\phi = [\tau^2, w_q, \sigma_n^2]$ as the hyperparameters of the GP. Different types of kernel functions (i.e., constant, Matern, periodic and many more) can be used to encode specific characteristics (e.g., smoothness, periodicity) of the underlying function f. Once the optimal parameters have been identified, given training measurements Y at sampling locations X, function values f_* at new test points X_* can be predicted by (Eq. (3)):

$$\begin{bmatrix} Y \\ f_* \end{bmatrix} \sim N\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} \Sigma(X,X) + \sigma_n^2 I & \Sigma(X,X_*) \\ \Sigma(X_*,X) & \Sigma(X_*,X_*) \end{bmatrix}\right) \tag{3}$$

To optimize the hyperparameters of a GPR model, the Bayesian updating scheme is used. By conditioning the joint Gaussian prior on the observed data points (X, Y) (i.e., Bayesian update) (Rasmussen and Williams, 2005), we get the conditional distribution (i.e., posterior predictive distribution) of f_*. The core step of the methodology, is the update of the prior distribution (belief) over function f (i.e., $p(f|X)$) to the posterior distribution $p(f|X,Y)$ based on the given observation data (X,Y) (Eq. (4)).

$$p(f|X,Y) = \frac{p(Y|f,X)p(f|X)}{p(Y|X)} = \frac{p(Y|f,X)p(f|X)}{\int p(Y|f,X)p(f|X)df} \tag{4}$$

In Eq. (4), $p(Y|f,X)$ is the likelihood function, $p(Y|X)$ is the marginal likelihood function, and $p(f|X,Y)$ is the posterior predictive distribution. The posterior predictive distribution f_* at test points X_* follows a Gaussian distribution as shown in Eq. (5):

$$f_*|X_*,X,\ Y \sim N\Big(m(X_*) + \Sigma(X_*,X)\left[\Sigma(X,X) + \sigma_n^2 I\right]^{-1} (Y - m(X)),\ \Sigma(X_*,X_*) - \Sigma(X_*,X)\left(\Sigma(X,X) + \sigma_n^2 I\right)^{-1}\Sigma(X,X_*)\Big) \tag{5}$$

The hyperparameters ϕ of GPR is often estimated by maximizing marginal log-likelihood function (Eq. (6)):

$$\phi^* = \underset{\phi}{argmax}[logp(Y|X)] = \underset{\phi}{argmax}\left[-\frac{1}{2}log\left|\Sigma(X,X) + \sigma_n^2 I\right| - \frac{1}{2}(Y - m(X))^T\left[\Sigma(X,X) + \sigma_n^2 I\right]^{-1}(Y - m(X)) - \frac{N}{2}log2\pi\right] \tag{6}$$

For the noise-free observations, σ_n^2 can be fixed to with zero.

2.2. Neural networks

Artificial neural networks (NN) (Krogh, 2008) are non-parametric regressors that have received significant attention as computational resources have grown. These models have excelled in Big-Data applications such as natural language processing, imaging, and automation (Goldberg, 2017; McCann et al., 2017). Due to their feature of being universal approximators (Cybenko 1989), NNs are capable of modeling complex, nonlinear relationships in high dimensional spaces, provided that there is a deterministic relationship between inputs and outputs and sufficient/representative training data. The mathematical foundation of a NN is based on the multi-layer perceptron model (Grossberg, 1988). Fig. 1 provides a visual representation.

Here a simple two-layer model is shown that can be used to connect a set of inputs to outputs. Input nodes feed the hidden nodes as a sum of bias parameters z_o and $i = 1, \ldots, D$ inputs x_D multiplied by $j = 1, \ldots, M$, corresponding weights w_{ij} (represented as lines). The resulting value at each hidden node passes through an activation function ($h(x)$) before

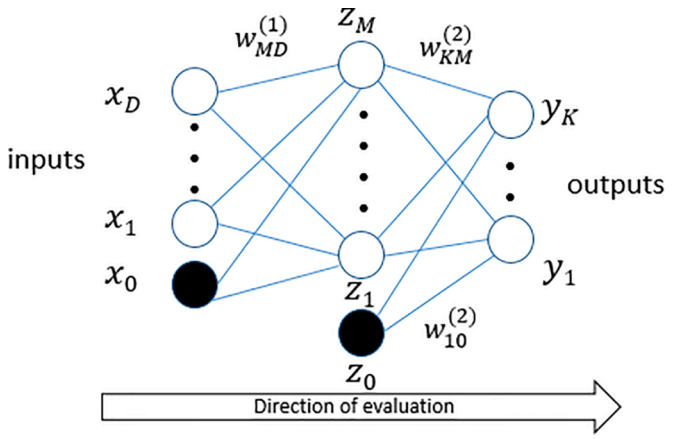

Fig. 1. Neural Network structure showing input, hidden, and output variables represented by nodes. Weights are represented by connecting lines and biases by dark nodes (x_0,z_0) (Adapted from Bishop-Pattern Recognition and Machine Learning (Bishop 2006)).

feeding the output layer, which again has its own activation function ($\sigma(x)$), and weight and bias parameters corresponding to the number of outputs, $k = 1, \ldots, K$. Thus, the NN output can be mathematically derived as a function of input and network parameters, as shown in Eq. (7), for a 2-layer NN.

$$y_k(x,w) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + z_{j0}^{(1)}\right) + z_{k0}^{(2)}\right) \tag{7}$$

The choice of the activation function can significantly affect the NN's predictive performance. Common activation functions include sigmoids, such as the logistic function, softmax function or hyperbolic tangent. These are continuous functions that mimic the behavior of biological neurons that turn on and off. In large-scale deep NNs, the piecewise linear ReLU activation function has been used widely to speed up training of network parameters, since sigmoid functions can present vanishing gradient problems (Hochreiter, 1998). Eq. (8), which shows the explicit form of a NN predictor can be trained by minimizing an objective function that captures the error between predictions and data (loss function). A single output loss function is shown below, where y_n is the model prediction, $\tilde{y}_n$ is the true value with N total data points.

$$pMSE = f(w) = \frac{1}{N}\sum_{n=1}^{N}(y_n - \tilde{y}_n)^2 \tag{8}$$

The mean-squared error (MSE) is commonly used for regression applications, where the goal is to minimize error between model prediction and the true value in the dataset. Cross entropy loss functions are often used for classification models with discrete outputs. Using the chain rule, the gradients of the objective function with respect to model weights are evaluated ($\nabla f(w)$). Weights are most commonly optimized using the gradient descent method Eq. (9), while high-performing alternatives also include genetic algorithms and quasi-Newton methods.

$$w^{\tau+1} = w^{\tau} - \eta\nabla f(w^{\tau}) \tag{9}$$

Hyperparameter η denotes the learning rate, which controls the step size in the optimization routine. NN evaluation, gradient calculation, and parameter updating continues iteratively until a finite set of training iterations has been reached or an objective function tolerance is satisfied.

3. Merging data-driven with first-principles models

There are numerous ways to merge first-principles and data-driven tools for modeling, which span the entire gamut of applications in

science and engineering where data is available and expert knowledge can be communicated in mathematical form. First, let us define the types of first-principles models discussed in this contribution. First-principles models in this paper refer to models derived from fundamental laws of physics, chemistry, thermodynamics, kinetics and transport phenomena, such as mass balance and energy balances. The terms "mechanistic", "engineering", and "physics-based" models are used in the literature to describe such models and are used interchangeably in this paper. These models come in different forms, such as fundamental algebraic equations, or a system of Ordinary or Partial Differential Equations (ODE or PDE), or a combination of both, to form a general nonlinear algebraic partial differential system of equations (NAPDE). Depending on the level of the modeling detail, or fidelity, a first-principles model can be in the form of a large complex computer simulation, with embedded hundreds of different equations and numerical techniques coupled to produce outputs, such as a Finite Element Method (FEM) model, a Computational Fluid Dynamic (CFD) simulation, or a Discrete Element Method (DEM) simulation. While first-principles models are available in different levels of fidelity and accuracy, in this work we will refer to any model derived based on fundamental knowledge as a first-principles model. Although this contribution will focus on methods for systems with continuous dynamics, there is no prohibition that such methods could not be extended to stochastic systems or systems with discrete or non-continuous relationships.

Broad surveys of hybrid approaches to integrate various types of theory and data can be found in Karpatne et al. (2017), Willard et al. (2020) and Rüden et al. (2019). Those surveys show that there are numerous proposed approaches which are used to inform training of data-driven models with physical knowledge, several of which are strongly application-dependent, of which only a few are covered in detail in subsequent sections. We limit our review to hybridization techniques that involve the presence of a spatio-temporal and/or algebraic mechanistic model that is used in some way during the training process (i.e., in the form of a constraint, or in combination with a data-driven component).

The main reasons for merging mechanistic knowledge with data-driven knowledge are ultimately better predictive ability of the final hybrid model (especially with reduced data requirements or in the presence of noise), and/or improved interpretability of the hybrid model. Depending on the structure of the known and unknown parts of the system, or the reason for building a hybrid model, there are multiple structures of hybrid models found in literature. In the next section we attempt to delineate these different structures and discuss them with respect to different methodologies.

3.1. Different structures of combined data-driven and first-principles models

Building models to find correlation or causation between system inputs and outputs is known by many names including model fitting, training, parameter estimation, statistical inference, supervised learning, or regression. Even within the subclass of regression-based problems, various methods exist to merge data and physical insights. The most prominent of these are categorized in Fig. 2. Each of these schemes is reviewed in depth in this article. However, we first briefly discuss here different motives for applying these methods.

Fig. 2 introduces general notation that will be used throughout this article for describing hybrid modeling formulas. In Fig. 2, the output of each hybrid model Y is considered a function of system inputs x. Depending on the framework, the final predictive model may be a data-driven model $DD(\cdot)$ (if first-principles is only considered during training) with parameters ϕ or a data-driven model combined with a first-principles model $f(\cdot)$ with mechanistic parameters θ. The terms δ and ε are unique to the calibration framework and refer to the model discrepancy and error function, respectively.

While no graphic could possibly capture all hybrid approaches and some approaches have no consistent literature definition, we list common hybrid approaches using Fig. 2 as a guide.

- Scheme 2A: A data-driven surrogate model is used to create offline a replacement model for a mechanistic model, which can then be simulated faster online than the original mechanistic model. The surrogate model is trained only on data and no further considerations or constraints are imposed to embed physics.

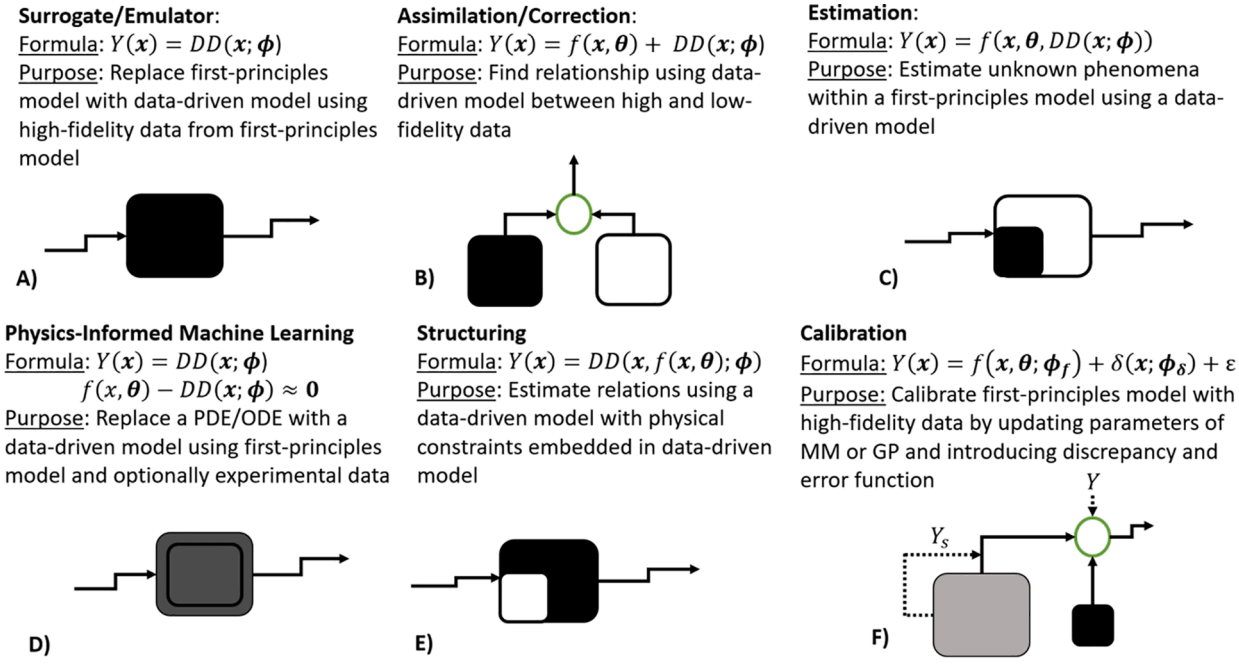

Fig. 2. Structure, notation and motivation for six hybrid modeling scenarios addressed in this paper. θ and ϕ represent the parameters of the first principles and the data-driven model, respectively. DD refers to Data-Driven Model and MM refers to the Mechanistic Model. Note that multi-fidelity data (Y_s : low-fidelity data from computer simulation, Y: high-fidelity data from experimental observations) are considered for MC.

- Scheme 2B: When a mechanistic model is available, but fails to accurately capture system behavior, a correction scheme may be applied by modeling the residual between low-fidelity mechanistic model and data via a data-driven model. Both the mechanistic model and data-driven correction are simulated jointly to produce the corrected output. If only the residual information is required, the HM scheme follows the scheme in Scheme 2B
- Scheme 2C: Like the hybrid correction scheme, this approach creates a model with data-driven and mechanistic components. However, unlike the correction scheme, the output of the data-driven component aims to model a specific phenomenological relationship, yielding a more interpretable output.
- Scheme 2D: Similar to the scheme 2A in that a data-driven surrogate model replaces a more complex mechanistic model. However, PI-ML is more involved in that it accounts for mechanistic constraints during training. In recent years, of special interest is when $f(\boldsymbol{x}(t), \theta)$ represents a system of differential equations and methods in derivative estimation (in particular, automatic differentiation) must be used to ensure the data-driven model is a solution of the differential form of the mechanistic model.
- Scheme 2F: The Structuring approach builds known mechanistic relationships (i.e., constraints) into an otherwise data-driven model. Although applications for this approach are more narrow than other hybrid approaches, when appropriate this scheme can improve interpretability over a purely data-driven approach while identifying both causal and correlative relationships. Depending on the embedded structure, scheme 2F may call upon similar methods as the PI-ML approach.
- Scheme 2E: Model Calibration calibrates a low-fidelity mechanistic model by utilizing both low-fidelity data (from the mechanistic equation) and the high-fidelity data (from the experiment) to generate high-fidelity output. It differs from the Correction scheme in its use to fit or 'calibrate' unknown parameters of the mechanistic model. Moreover, it attempts to more explicitly distinguish sources of error by separately modeling discrepancies due to data uncertainty and model uncertainty.

3.2. Hybrid (sub) modeling

Introduction

The framework most frequently associated with the keywords *hybrid modeling* in academic literature consists of constructing models that have distinct mechanistic and data-driven submodel(s). This framework is frequently referred to as hybrid semi-parametric modeling, or simply hybrid modeling (HM). To distinguish this method from other methods discussed in this review, we refer to this framework as hybrid submodeling (HSM) and the individual models merged to construct the HM as submodels (SMs). Hybrid models should not be confused with hybrid systems from control theory, which refer to control problems with both discrete and continuous decisions (Goebel et al., 2009). HMs may consist of multiple data-driven models, mechanistic models, or combinations thereof. Methods for combining multiple data-driven models have been reviewed extensively elsewhere (Hajirahimi and Khashei, 2020; Tascikaraoglu and Uzunoglu, 2014; Deb et al., 2017; Zendehboudi et al., 2018). Instead, this review focuses on HMs that merge both mechanistic and data-driven sub-models.

Methods

Applications in model building for process systems engineering (PSE) that call for HSM can generally be divided into two categories: mechanism estimation and mechanism correction. In the case of mechanism estimation, the modeler has available a mechanistic model (e.g. a series of conservation balances) within which one or more physical relationships is unknown or partially-known (e.g., a reaction rate or friction term) and estimates the unknown phenomena using a data-driven model. Alternatively, if the mechanistic model is available, but there exists a substantial discrepancy between observed data and mechanistic model predictions, the data-driven model is used to model the discrepancy (Su et al., 1992; Lee et al., 2005; Duarte et al., 2004; Chen et al., 2004). This latter formulation, sometimes referred to as discrepancy or residual modeling, can be viewed as a more general version of mechanism estimation and is herein referred to as mechanism (or model) correction. Naturally, formulations that combine mechanism estimation and correction have also been proposed (Thompson and Kramer, 1994; Wang et al., 2010; Chen and Ierapetritou, 2020). Eqs. (10)–(12) show potential formulations of each for the case where the HSMs can be modeled by differential equations.

Mechanism Estimation

$$\frac{dx}{dt} = f(x(t), c(t), \theta, DD(x(t), \phi)) \tag{10}$$

Mechanism Correction

$$\frac{dx}{dt} = f(x(t), c(t), \theta) \tag{11}$$

$$x_{\Delta t+t} = DD(x_t, \phi) \tag{12}$$

In the above equations (Eqs. (10)–(12)), the state variables x are modeled by mechanistic model $f()$, which is a function of the external forcing (i.e., control or operating) variables $c(t)$, mechanistic parameters θ, and data-driven relationships parameterized by ϕ. It is worth noting that mechanistic parameters μ can be constant values or represent mechanistic relationships with parameters that must also be estimated. When using formulas of the form of Eq. (10), the mechanistic and data-driven relationships must be evaluated simultaneously as the differential equations are integrated. Conversely, in the mechanism correction framework (Eqs. (11) and (12)) the mechanistic SM can be simulated independently of the data-driven SM; the outputs of the mechanistic model at time t are used as inputs to the data-driven SM to predict process conditions at time $t + \Delta t$. The data-driven submodel can then predict the system state directly or predict the state residual, which are added to the mechanistic submodel predictions for the final state prediction. Schematic representations of the above equations are presented in Fig. 3.

While not representative of all HSM arrangement possibilities, HSM arrangements depicted in Fig. 3 attempt to portray a general class of methods for arranging HSMs for modeling dynamic data. Notably, many authors choose to distinguish frameworks in Fig. 3 based on whether information is exchanged between SMs "in series" or "in parallel" (von Stosch et al., 2014b; van Can et al., 1997). However, as can be seen from Fig. 3, this can be an oversimplification since HSM for model estimation and model correction often exchange information in ways that are both serial and in parallel. This is especially true for differential equation models with mechanistic and data-driven terms. For these hybrid DEs, relationships between SMs are generally recursive and coupled and thus the relationship between data-driven and mechanistic terms are likewise recursive rather than serial or parallel. This has been acknowledged by other authors, preferring to use 'integrated hybrid models' (Quaghebeur et al., 2021; Quaghebeur et al., 2022) or 'universal differential equations' (Rackauckas et al., 2020; Bangi et al., 2022) to describe hybrid differential equation models. To avoid possible confusion, we use mechanism estimation and correction to distinguish the end use of the HSM framework. A study by Agarwal et al. exhaustively compared possible ways to arrange hybrid submodels for applications in modeling and control (Agarwal, 1997).

Whether used for mechanism estimation or correction, building the HSMs follow the general steps of data preprocessing, model formulation, model-fitting, validation, and testing/implementation. Preprocessing can be further divided into steps for outlier removal, interpolation of missing data, and feature selection. Bollas et al. demonstrated an approach for identifying significant features when constructing HSMs (Bollas et al., 2003). Preprocessing can sometimes include steps to correlate measurable observables to unmeasurable quantities of interest,

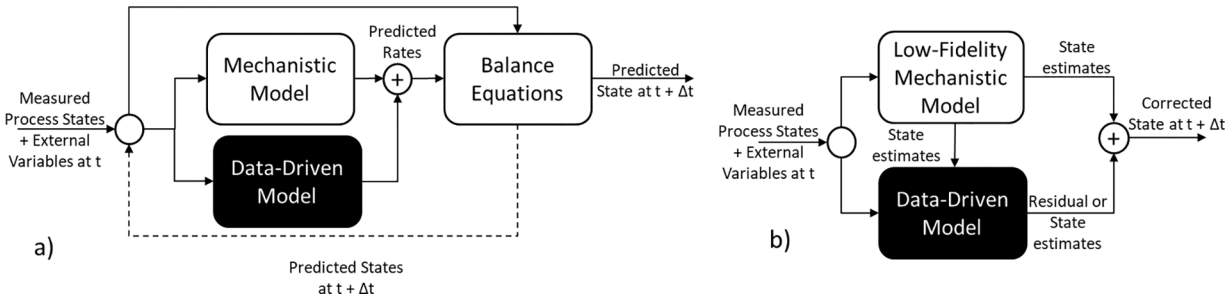

Fig. 3. Information flow between submodels for HMs for (a) mechanism estimation and (b) mechanism correction.

using a combination of data-driven and mechanistic modeling, creating the equivalent of a soft sensor. When these correlation models are combined with an overall system model, they may be referred to as submodels of the HM (Meng et al., 2019; Thompson and Kramer, 1994; Gibert et al., 2016; Lopez et al., 2020).

Challenges

An important question to ask is what role the mechanistic SM plays during training of the data-driven SM. Merging SMs during parameter estimation invariably increases the modeling effort and compute time. The answer generally depends on whether the features of the data-driven model can be measured directly. In the case of mechanism correction, the residual often can be computed with a pretrained mechanistic SM and measurement data, and the data-driven SM is trained separate from the mechanistic SM. However, when large gaps exist in measured data, fitting the data-driven correction model together with the mechanistic model can lead to improved performance (Wu and Movellan, 2012).

Likewise, parameter estimation of the data-driven SM for mechanism estimation can be performed in the presence (Psichogios and Ungar, 1992; Schubert et al., 1994) or separately from (Fiedler and Schuppert, 2008) the mechanistic model relationships. However, for systems formulated by differential algebraic equations (DAEs) often the data-driven SM is used to estimate a rate term which is not directly measured. To avoid using the mechanistic SM during training of the data-driven SM, early works proposed estimating the rates from state data, using for example finite differences (van Can et al., 1997). Yet for highly nonlinear, sparse or noisy data the accuracy of these rate estimates can be inadequate. For such situations it has been shown that integration of the differential equations during training is essential for accurate modeling (van Can et al., 1999) and for enforcing physical constraints (Oliveira, 2004).

When combined training of the data-driven SM with the mechanistic DAEs cannot be avoided, several strategies are available for managing computational costs. One strategy, applicable when multiple formulations of the data-driven model are being considered, is the incremental approach. In this approach, the modeler chooses to first fit multiple data-driven models separate from other mechanistic relationships and then selects the best-performing data-driven SM via cross-validation before training data-driven and mechanistic SMs together (Kars and Marquadt 2008). A similar incremental approach has been proposed when estimating the parameters of both the mechanistic and data-driven SMs (Yang et al., 2011) as well as selecting candidate mechanisms from a basis set via sparse regression (Willis and von Stosch, 2017). Another strategy is to reduce the cost of integrating the DAEs. The integrated framework proposed by (Psichogios and Ungar, 1992) relies on forward sensitivity analysis for parameter estimation, which is computationally expensive since the number of differential equations that must be integrated increases with the number of parameters (Narayanan et al., 2019). Thus, one way to reduce compute time is to leverage less precise numerical methods when finer accuracy is not needed. In (Oliveira, 2004), researchers compared the computational efficiency of the Euler method with a more sophisticated Runge-Kutta-based discretization method, showing the latter required an increase in computation time by two orders of magnitude. Other works have investigated methods for reducing computation with adaptive step sizes (de Azevedo et al., 2015). Most recently, strategies for reducing compute costs by avoiding integration of the sensitivity equations altogether have been investigated. This strategy has been made possible by the recent development of software integration routines with pervasive automatic differentiation (AD) (Rackauckas et al., 2018). In brief, by avoiding the integration of the sensitivity equations, there is potential to prevent the explosion in computational cost for training DAE models with a large number of parameters (Rackauckas et al., 2020; Chen et al., 2018).

Overall it should be mentioned that HSM structures should be pursued once the modeler is as sure as can be that any errors are not caused by local minima of the parameter estimation of the mechanistic model. In other words, it is worth applying multi-start local or global optimization methods, to find the optimal parameters of the mechanistic model before deciding to add data-driven corrections. In most cases, HSM methods are pursued because it is known *a-priori* that some simplifying assumption has been made, and/or it is very expensive or impractical to identify the fully mechanistic model for a certain system.

As a final consideration, it is often the case that both the mechanistic and data-driven SMs have parameters that require identifying (for examples see Yang et al., 2011). However, works that simultaneously train the mechanistic and data-driven submodels predominantly assume that the mechanistic model has no parameters or that their parameters are fixed at known values. Thus, additional research is needed to weigh the merits of schemes that estimate parameters, whether simultaneously or sequentially, of multiple SM types. While identifying mechanistic parameters offers the potential for increased interpretability, a foreseeable challenge is the presence of multiple local minima of the data-driven SM, which may make finding meaningful values of mechanistic parameters difficult (Francis-Xavier et al., 2021). If greater interpretability is needed than what can be offered by the HSM paradigm, it may be advantageous to use HSM as a data-driven means to a more mechanistic end (Bradley and Boukouvala, 2021) rather than as an end in itself.

If the data available for training the DD components of the HSM contain noise, HSM structures are advantageous over pure black-box surrogates, especially when the DD component is trained together with DE constraints. However, it is important to ensure that in all cases a regression-type DD component is employed, and appropriate tuning of its parameters is performed using cross-validation procedures, to ensure overfitting is avoided.

3.3. Physics-informed machine learning

Introduction

An area that has gained significant attention in recent years is Physics-Informed Machine Learning (PIML). It has emerged as a way to take advantage of major advances in ML for the purposes of surrogate modeling and system identification, while still enforcing physical knowledge that is known about the system at hand. This can assuage concerns that ML approaches abandon all of the useful information

given from first principles models or that data-driven models may give erratic results. While some of the proposed mathematical foundations for enforcing differential-equation physical constraints or knowledge (e.g. boundary and initial conditions) into ML models have been around for decades (Lagaris et al., 1998), increased computational power, major break-throughs in ML, and the power of automatic differentiation (Baydin et al., 2017) has allowed for PIML models to show promise across many disciplines and applications. These models incorporate deep-learning architectures such as neural networks or Gaussian processes, but also leverage known physical constraints specific to the application. This may be particularly useful in applications where the underlying physical models are too expensive to solve using traditional methods or where first-principles knowledge is helpful yet insufficient in predicting relationships between inputs and outputs. Major work has been done in the areas of differential equations, physics, power systems, and robotics. Many process systems engineering models rely on analogous conservation balances that can also be formulated into a PIML model.

Methods

Standard deep-learning models take labeled data and map the relationships between inputs and outputs. Labeled data refers to data points that have a measured or ground-truth output, which is very useful for training, validating and testing against an objective measure of model accuracy. Even when input-output relationships are highly-nonlinear and not completely understood, ML models have shown great success as function approximators with sufficient labeled training data. A great insight in the physics-informed ML literature is the ability to use unlabeled data points in order to penalize constraint violations over the entire input space of the ML model (Raissi et al., 2019). Unlabeled data refers to points in the input space without available measured or ground truth values to compare model predictions to. Though most ML training routines would ignore parts of the input space without data, physics-informed training structures can instead enforce general physical knowledge of the system we know to be true. While many variations and applications exist, the most common method is to add these constraint violations directly to the loss function during ML model training (see Fig. 5). The loss function evaluates how well the model is performing and is minimized by changing model parameters over the training routine. By including knowledge directly in the loss function, the resulting model parameters will be biased towards the embedded knowledge. This can be thought of as a soft constraint, as there is no guarantee that it will be satisfied. Instead, the learning task balances the two learning goals simultaneously: improving model agreement with data and adjusting model parameters to follow known constraints. This is useful if the full engineering model is too computationally expensive to simulate repeatedly. For example, say we want to fit a surrogate ML model to labeled data with inputs x and t and output y (Eq. (13))

$$y(t,x) = DD(t,x,\phi) \tag{13}$$

Furthermore, in the physical system we are modeling, we know the functional form of constraints that depends on input or output variables, such as the generic equation and inequality shown in Eqs. (14) and (15).

$$f(t,x,y) = 0 \tag{14}$$

$$g(t,x,y) \leq 0 \tag{15}$$

If these constraints apply to the full input domain, this input space can be discretized and the constraint can be numerically evaluated at the discretized points, what some in the literature refer to as collocation points. A loss function with a soft constraint can then be formulated as shown in Eq. (16).

$$\begin{aligned} Loss \;=\; & \lambda_y \frac{1}{N}\sum_{i=1}^{N}\left|DD\left(t_y^i,x_y^i\right)-y^i\right|^2 + \lambda_f \frac{1}{N_f}\sum_{j=1}^{N_f}\left|f\left(t_f^j,x_f^j,y\left(t_f^j,x_f^j\right)\right)\right|^2 \\ & +\lambda_g \frac{1}{N_g}\sum_{k=1}^{N_g} max\left(0,g\left(t_g^k,x_g^k,y\left(t_g^k,x_g^k\right)\right)\right) \end{aligned} \tag{16}$$

In Eq. (16), the first term evaluates the mean squared error between a set of N labeled data points and the ML model's prediction and would be common to the loss function in any regression task. The second and third terms evaluate the known equality and inequality constraints, respectively, at any unlabeled data point of sets N_f and N_g, respectively. Hyperparameters ($\lambda_f,\lambda_g,\lambda_y$) can be adjusted to control the relative weight of these terms. In all deep learning models, finding optimal hyperparameters can be expensive and most methods simply consider grid search techniques to balance over and underfitting of validation data. In this way, the mechanistic terms in Eq. (16) can be treated the same as regularization terms used in fully DD models.

A more rigorous variation of the soft constraint method outlined above uses Lagrangian optimization theory and the hyperparameters become Lagrangian multipliers (Fioretto et al., 2020). Depending on the final values of λ's, this method can provide some level of guarantee for the constraints at the training values. It also may avoid some ill-conditioning issues in the simpler penalty approach. A full description of this Lagrangian method applied to deep learning models can be found in (Fioretto et al., 2020) and the underlying theory in Freund (2004).

Another common way to embed physical knowledge into ML models is to pre-train, or initialize, model parameters. This could be done using a subset of data collected from a single larger system or using a separate data set from a system known to share physical characteristics with the target system. Since many of the ML algorithms depend on stochastic gradient descent, a pre-trained model can help to avoid local minima that don't obey physical knowledge. This is especially helpful when fitting deep ML models to sparse data and is often called transfer learning. A full review of transfer learning can be found in (Pan and Yang, 2010). This idea has further been used in chemical engineering literature under the term model migration (Lu et al., 2009), where their goal is to utilize underlying physical knowledge of an old process model to inform their new process model, then use a small data set to calibrate new process parameters and conditions using NNs as surrogates. In (Luo and Gao, 2015), model migration is explored for Gaussian process models used to predict chemical reactor performance. A GP model is trained under certain concentrations and temperatures, then model migration is used to make predictions at extrapolated points. Furthermore, the authors explain an approach to perform model migration and process optimization simultaneously.

Another area of research under the umbrella of PIML has looked into designing the architecture of ML models to confer known physical knowledge (Xia et al., 2008). This is the Physics-constrained Structuring approach depicted in the HM comparison figure presented earlier (Fig. 2, scheme E). By leveraging part of the ML model to impose physics, the ML model becomes more interpretable while guaranteeing domain knowledge is satisfied. This approach is especially relevant to NN models that are highly customizable and modular in structure. A simple example of this could include the use of ReLU or softmax activation functions, to enforce non-negativity in output or intermediate variables. Jia and co-workers (Jia et al., 2020) uses this idea when modeling the density of water as a function of lake depth to ensure their prediction model monotonically increases in density as a function of depth, as this is a well-known fluid property. Similar ideas are used to enforce structure in chemical flowsheet models (Wu et al., 2020). Monotonicity constraints have been systematically applied to other data driven model structures, such as decision tree models (Potharst and Feelders, 2002). This algorithm has been implemented into Python packages such as LightGBM (Hart et al., 2017).

More complex physical knowledge can be built into ML

architectures, such as local mass balances in differential form. In many applications, encoder-decoder structures are used to filter out measurement noise and transform data into a lower dimensional, information rich space (Vincent et al., 2008; Chen et al., 2018; Kim et al., 2020). Work in turbulence modeling has used non-trainable layers that compute the continuity equation to provide constraints that are built into the model itself (Mohan et al., 2020). A schematic of their approach is shown in Fig. 4.

In the above schematic, V represents input velocity data, which has some noise associated with it. $\widehat{V}$ represents the "coarse-grained" representation of the velocity after passing through the autoencoder to filter out noise. $\widehat{A}$ represents the vector potential which is explicitly defined in the physical fluid dynamics balance equations. Autoencoders are common when working with black-box models, since it is important to fit underlying signal instead of data-set noise. They comprise of two parts: an encoder which takes the input data and recasts into a lower dimensional representation and decoder that transforms the low dimensional data back into the same dimensional space as the encoder input. Local coherence in physical data motivates the use of Convolutional Neural Network (CNN) architectures, as high-dimensional data that has spatial dependencies on nearby features (e.g. pixels in image processing) use CNNs to abstract spatial information from data. The final Convolutional Neural Network (CNN) has a kernel defined in a way that acts as the ∇ operator. For a full discussion and explanation of CNNs see (Dhillon and Verma, 2020). The overall model structure allows a lower dimensional representation to be found, while still enforcing the continuity equation. In the same field, researchers have used customized CNNs to incorporate various physical knowledge, such as uniform motion, rotation, and scaling (Wang et al., 2020). Their results show significant improvements in generalizability.

The area with the most extensive literature in Physics-Informed ML deals with the solution of complex differential equation models. ML models, including NNs, have long been used in the numerical solution of these systems due to their characteristic as universal function approximators (Dissanayake and Phan-Thien, 1994; Cybenko, 1989). Many approaches to this are very relevant to the earlier work of Lagaris et al. (1998), which formulates a trial solution to a differential equation as shown below.

$$\psi_t(x) = A(x) + F(x, N(x, w)) \tag{17}$$

In Eq. (17), $A(x)$ represents initial and boundary condition contributions to the solution, while F represents the functional form of the solution that is decoupled from these conditions and N represents the output of the neural network model. The trial solution in Eq. (17) can be analytically differentiated with respect to the independent variable (x), to match the original differential form and from here an unconstrained optimization problem is formulated to minimize violation of the differential equation with network parameters (w) as the decision variable. The authors demonstrate several functional forms of the trial solution for first order ODEs, second order ODEs, systems of ODEs and nonlinear PDEs. It is important that samples from an analytical solution are available and that the candidate solution can be constructed from the known differential equation. Subsequent work in this area has increased the dimensional size of problems solved through this DL method (>200 dimensions) (Sirignano and Spiliopoulos, 2018) and various generalized algorithms have been developed to automatically generate trial solutions. In (Sirignano and Spiliopoulos, 2018), authors propose a "Deep Galerkin Method" algorithm which generates random sample points to avoid the need for traditional mesh methods (intractable in high dimensions) and utilizes stochastic gradient descent to find parameters that minimize squared error between the model prediction and the analytical solution.

Building off the work of Lagaris et al. (1998), Raissi et al. utilized the power of automatic differentiation to preclude the need for analytical derivatives and applied PDE knowledge to feed-forward neural networks, which they call "physics-informed neural networks" (PINNs) (Raissi et al., 2019). The PINN formulation is analogous to the Eqs. (14)–(16), where f becomes a differential constraint computed via automatic differentiation. They show that this is a very powerful technique for modeling PDE's when model outputs are differentiable with respect to model inputs. A schematic of the PINN is shown in Fig. 5.

The authors use sampled points across the problem domain or "collocation points" to calculate coherence to PDE and BC knowledge, while using analytical solutions for data. This work uses primarily Dirichlet boundary conditions. Despite the heavy focus of PI-ML methods on NN models, Gaussian Process models have also been employed. In (Raissi and Karniadakis (2017b) Raissi, Perdikaris, and Karniadakis 2017b), the authors present how numerical Gaussian Process regression models can be used to solve PDEs from noisy data. They use a backwards Euler approximation of the PDE to express each solution point as a function of previous Gaussian Process prior. Hyperparameters in the kernel model are optimized at each step with the resulting Gaussian Process model able to predict data for the next step. By linking each time step in this way, PDE knowledge is incorporated into the DL model. Their results show that for classic physics-based PDEs accurate solutions can be found with sparse training data. This method did not perform as well with non-linear operators, since linear approximations must be used. Since then, an inference procedure usually nonlinear Gaussian Processes has been developed that can be considered a Bayesian version of PINNs (Yang et al., 2021). One major advantage of the physics-informed Gaussian process models (PIGPs) is the estimate of

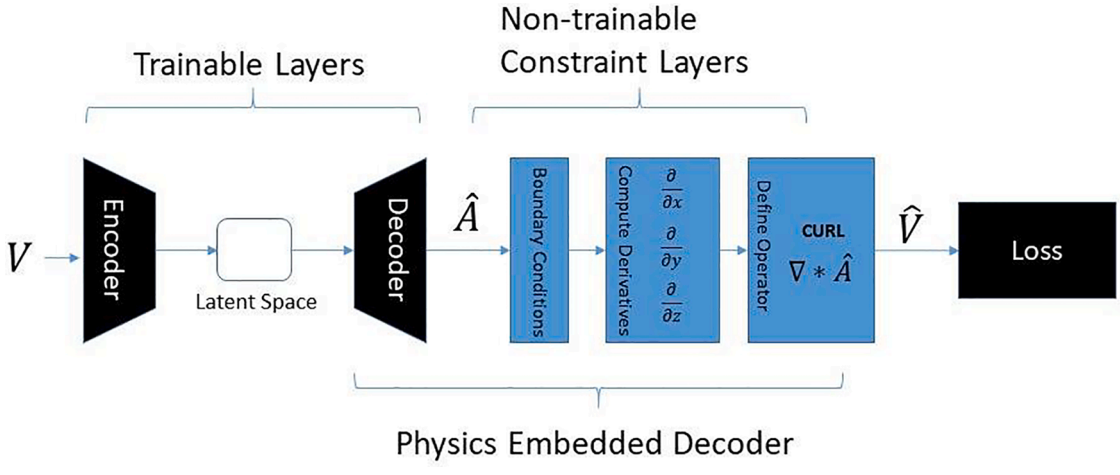

Fig. 4. Example of enforcing physical knowledge within NN architectures via constraint layers and encoder-decoder layers.

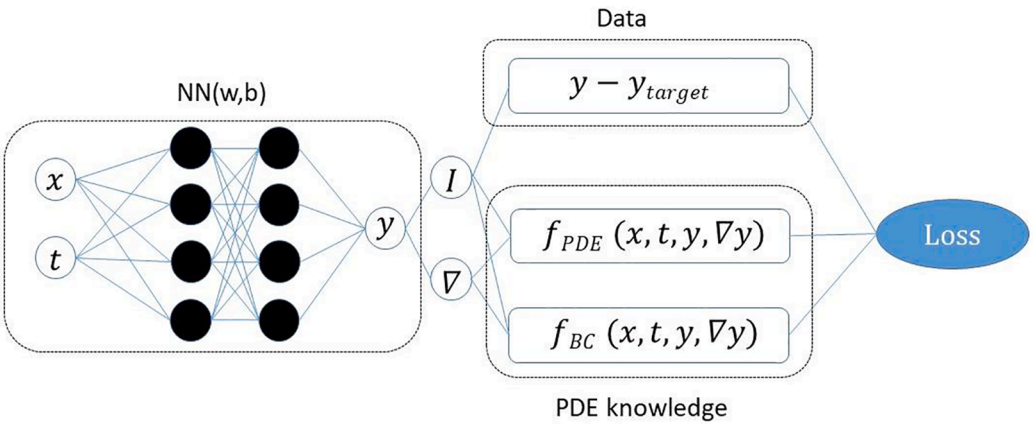

Fig. 5. Schematic representation of Physics-Informed NNs Used for PDE's.

uncertainty which is absent in analogous PINN models. A full review on the comparison between PINNs and PIGPs can be found in Kevrekidis et al. (2020). Another methodologically related topic is Sobolev training of NNs (Czarnecki et al., 2017), where the model is trained on not just state data but also derivative data if available. This may be useful in the area of network compression, where one wants a smaller network that best approximates the large model or when computing derivatives of high fidelity models isn't computationally prohibitive (Tsay, 2021).

Challenges

Overall, the methods and applications outlined show many current strengths, along with limitations and areas for future work. Physics-informed ML show great potential in the field of PSE and can be directly applied to many already existing research directions. A direct application includes improving the robustness of PSE surrogate models which are commonly used to approximate complex unit operations with the ultimate goal of optimization or control. To be suitable for control, a model must define the relationship between system states and controllable variables. However, PINNs designed for the solution and fitting of differential equations do not use states but rather independent variables such as time and spatial coordinates as inputs to enable derivative calculation. Recent work has sought to extend PINNs to state-space modeling for control (Arnold and King, 2021), though a comparison with standard data-driven approaches is lacking. Other PIML approaches formulate the PINNs as a purely state-space model. These approaches owe their success to taking advantage of the stationary nature of the problem (Fioretto et al., 2020) or system symmetry (Lutter et al., 2019). An issue with these approaches is their complex integration between the physics and data-driven model may not be extensible to other systems. Assuming these barriers can be overcome, PIML may enable PSE surrogates to find optimal values that are more consistent with the ground truth physics. Model Predictive Control (MPC) is an area where hybrid model structures can thrive due to their quick computation and on-line 'learning'. Analogous to how NN models in robotics control can be embedded with Newtonian physics balances to achieve physics-informed online learning, one may imagine incorporating an energy balance into an ML based MPC model for process temperature regulation.

On the other hand, PINNs still share many of the weaknesses of traditional NNs: lack of interpretability, heavy initial training cost, challenges with extrapolation, and the requirement of many representative data or collocation points. Though they can mitigate some of the previously listed concerns, it is doubtful that they can displace many of the heavier first-principles models that have long been accepted academically and in industry. While PINNs have a great number of studies for systems well-posited physical relationships, there is sparsity of studies in areas relevant to chemical process systems, such as optimization of systems with time-varying control actions, disturbances, and poorly understood physics. Initial studies comparing PINNs with HSMs employing numerical methods have shown PINNs to be less flexible and less accurate (Mitusch et al., 2021). Another study explored the forward and inverse solution of a reaction system using a modified PINNs framework and illustrated how unmeasured state data could limit the identifiability of the fitted reaction parameters (Gusmão et al., 2020). A more general treatment of challenges associated with PINNs can be found in (Karniadakis et al., 2021).

However, there has also been some work to overcome these weaknesses for specific applications. Addressing the issue of lengthy training times, conservative PINNS (cPINNs) (Jagtap et al., 2020) and extended PINNs (XPINNs) (Karniadakis 2020a) are frameworks for sub-dividing the spatio-temporal domain into intervals regressed by separate NNs. Although not demonstrated, this approach lends itself to high parallelizability. More difficult to quantify, however, is the added time required to train hyperparameters of multiple NNs and decide how to properly decompose the spatio-temporal domain into an appropriate number of sub-intervals. Previously discussed Langrangian Dual techniques can improve and automate hyperparameter training. Further speed-ups in training and prediction accuracy have been achieved by employing adaptive activation functions (Jagtap et al., 2020) and modifying gradient contributions (Wang et al., 2020).

In all these examples, the authors demonstrate that the PINN framework supersedes the accuracy of purely data-driven approaches and requires less data. They also simulate faster than and are competitive in accuracy to the analytical model, at least for most interpolation tasks and certain extrapolation tasks (discussed later in Section 7). Moreover, in certain circumstances the PINN can predict more accurately than the analytical model. For example, in a robotics control study the PINN was shown to outperform the control actions of an analytical model since the PINN was able to be updated online (Lutter et al., 2019). Work on power systems have shown PINNs to generate far more accurate solutions than other reduced order modeling methods (Fioretto et al., 2020). In many ways, physics-informed ML represents a bridge between two major areas of scientific computing which may enable tackling real-world problems requiring both theoretical and empirical resources.

3.4. Model calibration

Introduction

In Model Calibration (MC), the typical scenario includes three components (a) a complex computer simulation that contains many equations and unknown parameters (e.g., Finite Element Model, Computational Fluid Dynamics model, system of Partial Differential Equations, etc.), (b) observed or experimental data (considered high-fidelity data), and (c) surrogate model(s) (typically GPR model(s)) to calibrate the computer simulation and capture the discrepancy between the computer simulation and the observed data. Computer simulations

are essential to understand and predict complex systems (Santner et al., 2003), however, simplifying assumptions often used in computer simulations cause a discrepancy between the simulation output and the experimental data. Moreover, the parameters of the computer simulation are often unknown. In this scenario, a GPR model can be trained using the low-fidelity simulation output and high-fidelity experimental data, by adjusting the parameters (i.e., calibration parameters (θ), GPR hyperparameters (ϕ)), with the ultimate goal of calibrating the simulation model to predict the high-fidelity experimental data. MC can be viewed as an example of multifidelity modeling in the sense that it considers both high-fidelity and low-fidelity data.

Methods

Variable inputs (x) and the calibration parameters (θ) are two kinds of inputs that are required to run a computer model. Variable inputs (x) are the inputs that can be observed or often controlled when conducting physical experiments. Calibration parameters (θ) refer to any physical or tuning parameters that are unknown or not measurable from the experiments but are used/required to run the computer simulation. As an illustrative example, when designing a distillation column for the separation of two unknown chemicals, let us assume that the number of stages, the reflux ratio, and the enthalpy of chemicals are needed to run the simulation. The number of stages and the reflux ratio are examples of variable inputs, while the enthalpy of chemicals are example of calibration parameters since true values are unknown to the modeler but are required to run a simulation. Since the true values of the calibration parameters are unknown, there is no way to observe the computer simulation output without specifying them (Kennedy and O'Hagan, 2001a). To run the simulation, researchers may try a brute force approach wherein they choose multiple values of the calibration parameters and settle for the value that minimizes the error between the computer simulation output $f(x,\theta)$ and the real data $Y(x)$. However, this approach is challenging when the dimension of calibration parameter space (θ) becomes large, or each computer simulation run is expensive. To overcome this issue, Kennedy and O'Hagan (Kennedy and O'Hagan, 2001a) proposed a combined framework to calibrate the computer model using the observed data (Eq. (18)):

$$Y(x) = f(x,\theta) + \delta(x) + \varepsilon \tag{18}$$

where $Y(x)$ is the true output of the system, $f(x,\theta)$ is the computer model output, $\delta(x)$ is a discrepancy function to capture errors between the model and the true output, and ε is the measurement error which captures the effect of noise or failed-to-include variables in the system (Joseph and Yan, 2015). In the case where $f(x,\theta)$ is an analytical equation, this can be directly embedded within Eq. (18). However, the most common scenario, which we also assume here, is the situation where $f(x,\theta)$ is expensive or complex computer model, which is represented by a GPR surrogate. The discrepancy function $\delta(x)$ is also typically represented by another GP surrogate.

We assume computer simulations Y_s are observed at set of points $S_1 = [(x_1^*, \widehat{\theta}_1), (x_2^*, \widehat{\theta}_2)\ldots, (x_N^*, \widehat{\theta}_N)]$and experimental observations Y are observed at $S_2(\theta) = [(x_1,\theta), (x_2,\theta)\ldots, (x_M,\theta)]$ with some unknown calibration parameters θ. Note that θ are generally assumed to be constant over the experiment. For each trial of computer experiments, computer simulation output y_i in $Y = [y_1, y_2, \ldots, y_N]$ is observed for $(x_i^*, \widehat{\theta}_i)$ when $i = 1,2,\ldots,N$. When we construct joint data vector d with simulation data Y_s and the observed data Y, $d = [Y_s,\ Y]^T$, the likelihood for the vector d follows the distribution in Eq. (19).

$$P(d|\theta,\ \phi) \propto |\Sigma_d|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(d-\mu_d)^T \Sigma_d^{-1} (d-\mu_d) \right) \tag{19}$$

Note that we have two sets of parameters (θ: calibration parameters, ϕ: GP hyperparameters) in the model. The mean and variance of the likelihood for the vector d are shown in (Eqs. (20) and (21)).

$$E(d|\theta,\ \phi) = \mu_d = \begin{bmatrix} m_f(S_1) \\ m_f(S_2(\theta)) \end{bmatrix} + \begin{bmatrix} 0 \\ m_\delta(S_2) \end{bmatrix} \tag{20}$$

$$var(d|\theta,\ \phi) = \begin{bmatrix} \Sigma_f(S_1,S_1) & \Sigma_f(S_2(\theta),S_1) \\ \Sigma_f(S_1,S_2(\theta)) & \Sigma_f(S_2(\theta),S_2(\theta)) + \Sigma_\delta(S_2,S_2) + \sigma_m^2 I_M \end{bmatrix} \tag{21}$$

where $m_f(\cdot)$ and $m_\delta(\cdot)$ denotes the mean function of computer simulation and discrepancy function, respectively; $\Sigma_f(S_1,S_1)$ is the covariance matrix between set of points in S_1; $\Sigma_f(S_1,S_2(\theta))$ is the covariance matrix between set of simulation points in S_1 and experimental observations in $S_2(\theta)$; $\Sigma_\delta(S_2,S_2)$ is the covariance matrix between set of points in S_2. Note that θ is not included in covariance matrix $\Sigma_\delta(S_2,S_2)$ because discrepancy function δ is not a function of θ. σ_m^2 is the noise variance where $\varepsilon \sim N(0,\sigma_m^2)$, and I_M is the $M \times M$ identity matrix.

If we use a squared-exponential kernel (See Eq. (2)) for the two GPR models (f and δ) and assume noise-free computer simulation data, $k_f((\cdot,\ \cdot),(\cdot,\cdot))$ and $k_\delta(\cdot,\cdot)$ become:

$$k_f\left((x_i^*, \widehat{\theta}_i), (x_j,\theta)\right) = \tau_f^2 \exp\left(-\frac{1}{2}\left[w_{f,1}(x_i^* - x_j)^2 + w_{f,\ 2}(\widehat{\theta}_i - \theta)^2 \right] \right) \tag{22}$$

$$k_\delta(x_i,x_j) = \tau_\delta^2 \exp\left(-\frac{1}{2} w_\delta (x_i - x_j)^2 \right) \tag{23}$$

where the hyperparameters of each kernel are $\phi_f = [\tau_f^2, w_{f,1}, w_{f,\ 2}]$ and $\phi_\delta = [\tau_\delta^2, w_\delta]$. From the joint distribution of the computer simulation Y_s and experimental observations Y, the joint posterior distribution of the GPR hyperparameters $\phi = [\phi_f, \phi_\delta,\ \sigma_m^2]$ and calibration parameters θ is obtained (Eq. (24)).

$$P(\theta,\ \phi|d) \propto P(d|\theta,\ \phi)P(\theta,\ \phi) = \ P(d|\theta,\ \phi)P(\theta)P(\phi) \tag{24}$$

In Eq. (24), an independent assumption between priors (e.g., $P(\theta,\ \phi) = P(\theta)P(\phi)$) is often used to facilitate the calculation for the posterior distribution (Higdon et al., 2004; Kennedy and O'Hagan, 2001a)

While an analytical expression for $P(\theta,\ \phi|d)$ can be obtained from full joint posterior distribution of ϕ and θ, it is an intractable function of ϕ (Kennedy and O'Hagan, 2001a). Therefore, full Bayesian analysis is difficult to obtain posterior calibration parameters $P(\theta|d)$ since it requires multidimensional integration (Bayarri et al., 2007; Higdon et al., 2004). As a substitite, the Markov chain Monte Carlo (MCMC) sampling method is often used to estimate a posterior (Diaconis, 2009); however, it requires expert knowledge and careful tuning of MCMC parameters to obtain proper posterior distributions (Bayarri et al., 2007; Liu et al., 2009). One of the popular methods that tackles this issue is called the modular Bayesian approach (Arendt et al., 2012), which separately estimates the GPR hyperparameters ϕ_f and $[\phi_\delta, \sigma_m^2]$, and uses the obtained posterior hyperparameters $\phi^* = [\phi_f^*, \phi_\delta^*, \sigma_m^{2*}]$ to calculate the posterior distribution of calibration parameters θ. The modular Bayesian approach is shown in Fig. 6.

If the computer model $f(\cdot)$ is a known mechanistic function with an explicit functional form, we can directly use the equation given instead of replacing it with a GP model. A computationally efficient way of estimating GPR hyperparameters ϕ_δ of the discrepancy function δ, calibration parameters θ, and experimental noise σ_m^2 is to maximize the joint posterior distribution $P(\theta,\ \phi|Y^T)$ (Joseph and Yan, 2015) (Eq. (25)).

$$\begin{aligned} (\theta,\ \phi|Y^T) &\propto P(Y^T|\theta,\ \phi)P(\theta,\ \phi) \\ &\propto |\Sigma_\delta + \Sigma_\varepsilon|^{-\frac{1}{2}} exp\left(-\frac{1}{2}(Y^T - f(\theta))^T (\Sigma_\delta + \Sigma_\varepsilon)^{-1} (Y^T - f(\theta)) \right) P(\theta,\ \phi) \end{aligned} \tag{25}$$

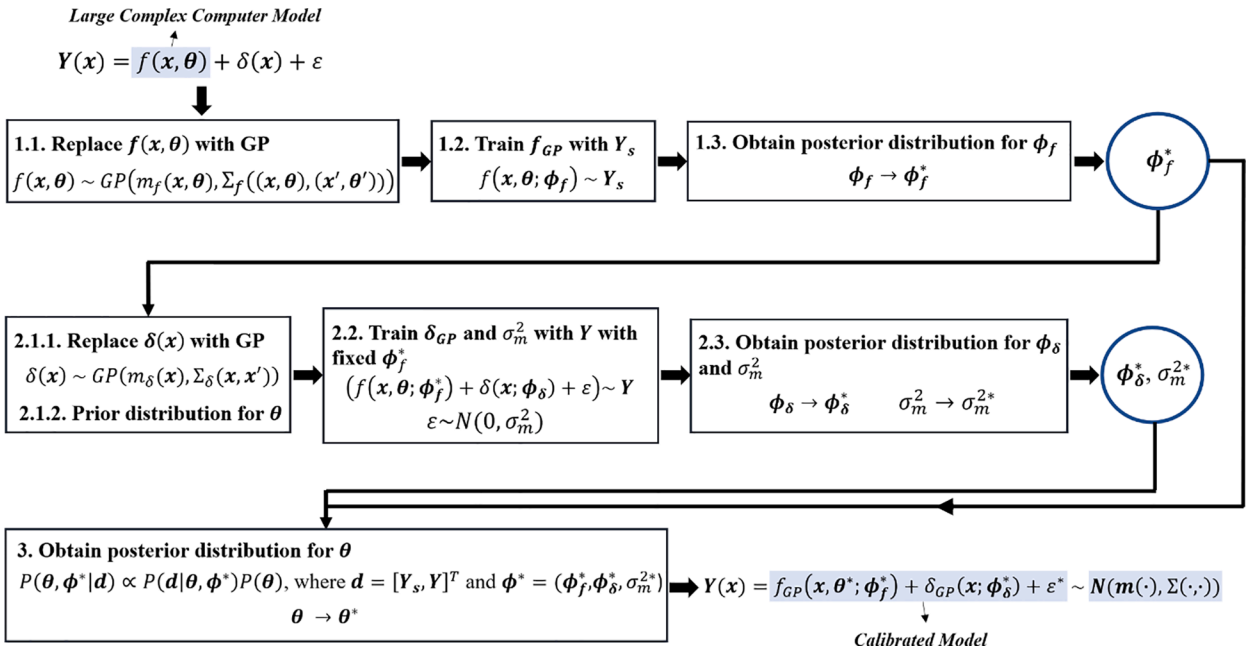

Fig. 6. Modular Bayesian Approach (Arendt et al., 2012) for model calibration when computer model is replaced with GP.

In Eq. (25), $\phi = [\phi_\delta, \sigma_m^2]$. Here, $P(Y^T|\theta,\ \phi)$ is obtained by integrating out $\delta(x)$ from the joint posterior.

Fig. 7 summarizes model calibration procedure discussed above.

To balance the objective function, a scale parameter ρ is often introduced (i.e., $Y(x) = \rho f(x,\theta) + \delta(x) + \varepsilon$) and it needs to be tuned together with the remaining hyperparameters. The posterior variance σ_m^2 is an important hyperparameter which controls the interpolating nature vs. regressive nature of the fitted model. Thus, if *a-priori* knowledge exists about the level of noise in the data, an informed prior assumption regarding this hyperparameter can be imposed, such that the final GPR model does not overfit the data and smooths out some of the noise. On the other extreme, if this is set or even fixed to zero, the GPR model will interpolate all of the data exactly. This will also affect the discrepancy component in the overall MC model, so it is important to use any prior information to initialize the MC approach, as well as study the final solution to identify whether an expected signal-to-noise ratio is captured in f versus δ. For more mathematical details on MC, please refer to

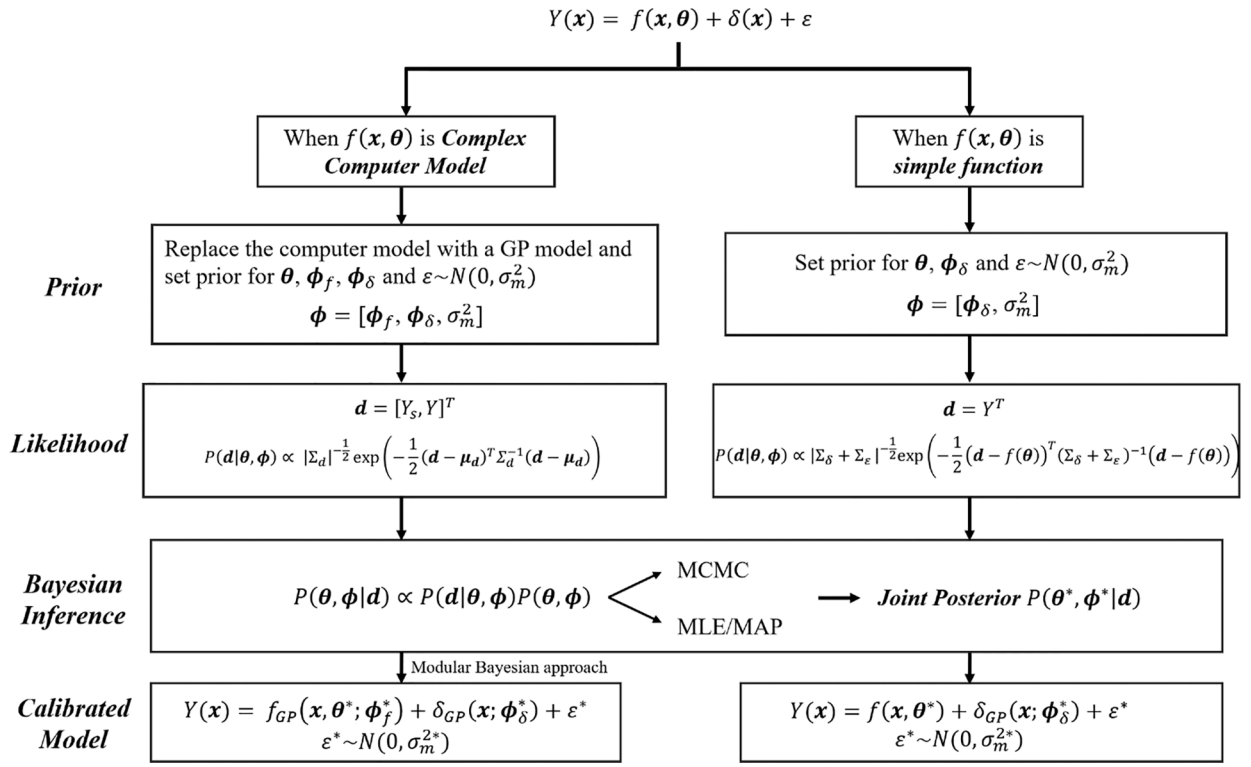

Fig. 7. Model Calibration Procedure. ϕ_f and ϕ_δ refers to the GP hyperparameters of computer model f and discrepancy function δ, respectively. σ_m^2 is the noise variance that captures experimental noise.

Kennedy and O'Hagan (2006).

One of the major advantages of model calibration is its ability to account for different sources of uncertainty (Arendt et al., 2012), which exists in almost all problems in science and engineering. Model calibration enables uncertainty quantification from different sources and helps make a proper decision by checking the degree of confidence and testing the reliability. The basic model calibration formulation (Eq. (18)) incorporates different kinds of uncertainty, such as the interpolation uncertainty through the $f(x,\theta)$ and $\delta(x)$ terms, the parameter uncertainty through the estimation of the parameter set θ, and the experimental uncertainty via ε. In a similar lens as Physics-Informed ML methods, incorporation of domain knowledge in MC is done via decisions for the form of prior settings for different parameters and the discrepancy function. These informative prior settings can help generate an accurate posterior, and this is similar to the process of pre-training or transfer learning in NN modeling. A MC structure with GPR models representing both the expensive computer simulation and the discrepancy have also been used to identify optimal locations for sampling to update the model parameters (i.e., Informative Experimental Design) (Chen et al., 2007), and for faster model validation (Chen et al., 2007; Lee et al., 2019).

Challenges

One of the main challenges in model calibration is that the inclusion of discrepancy term $\delta(x)$ makes the posterior calibration parameter θ^* hard to interpret. It is difficult to differentiate the effect of calibration parameters θ^* and the discrepancy function $\delta(x)$ as there can be many different solutions that produce similar performance of calibrated model. This is referred to as the "identifiability problem". Since we do not know which combination of calibration parameters and the discrepancy function is true, the system becomes non-identifiable (Arendt et al., 2012; Kennedy and O'Hagan, 2001a).

Fig. 8 shows different combinations of calibration parameters and the discrepancy function that explain the same system. The calibrated model with discrepancy function $f(x,\theta^*;\phi_f^*)+\delta(x;\phi_\delta^*)+\varepsilon^*$ predicts the same output $Y(x)$ well but the dynamics of $f(x,\theta^*;\phi_f^*)$ and $\delta(x;\phi_\delta^*)$ are different. It is shown (Arendt et al., 2012) that adding more simulation and experimental data for training the model does not solve the identifiability issue. To improve identifiability, an informative prior (accurate mean and low standard deviation) for the calibration parameter can be used (Bayarri et al., 2007; Liu et al., 2009). A specific functional form for the discrepancy function can also be employed for the discrepancy function if we have sufficient prior knowledge (Joseph and Melkote, 2009; Xiong et al., 2009). It is shown that identifiability is difficult but possible for some cases when proper priors for the calibration parameters and discrepancy function are set (Arendt et al., 2012). However, when the system is complex or not well understood, we may not have enough knowledge to set the informative prior for the calibration parameters and the discrepancy function. The identifiability problem often leads to the lack of interpretability in the model calibration technique. The physical interpretation of calibration parameters is not recommended since the calibrated value can be far from the true value (Kennedy and O'Hagan, 2001a). As an illustration example, if we set the molar weight of an unknown chemical as a calibration parameter (θ), the posterior calibration parameter (θ^*) can be negative, which is not physically valid.

A desired goal might be to employ MC to identify the cause of a discrepancy, which would then lead to mechanistic corrections of the computer simulation, including the addition of terms and parameters in the computer simulation instead of a discrepancy. This would be the ideal scenario, and different approaches have been proposed towards that goal (Lee et al., 2019; Wipf and Nagarajan, 2007; Yi et al., 2011; Piironen and Vehtari, 2016; Linkletter et al., 2006; Savitsky et al., 2011). However, this should be done with caution, since it has also been shown that the interpretation of GP hyperparameters can produce a false interpretation of the system (Lin and Joseph, 2019).

Overall, prior settings on model parameters and the discrepancy function are critical for model performance and interpretability. While informative prior settings can help improve model performance, inappropriate settings of a prior may lead to bad predictions and poor interpretability. However, specifying an appropriate prior is often challenging when we do not have enough prior knowledge of the system. The effect of different priors of hyperparameters on the GPR prediction performance is investigated in (Chen and Wang, 2017) and the authors concluded that the initial prior setting affects the convergence of the posterior hyperparameters to the true value. Selecting appropriate model discrepancy priors that capture missing physics in the system is also critical in model calibration. A different form of model discrepancy prior is compared in Ling et al. (2014) and they concluded that the calibrated parameters can be physically biased if we set inappropriate discrepancy priors, and as expected, this also affects the extrapolation

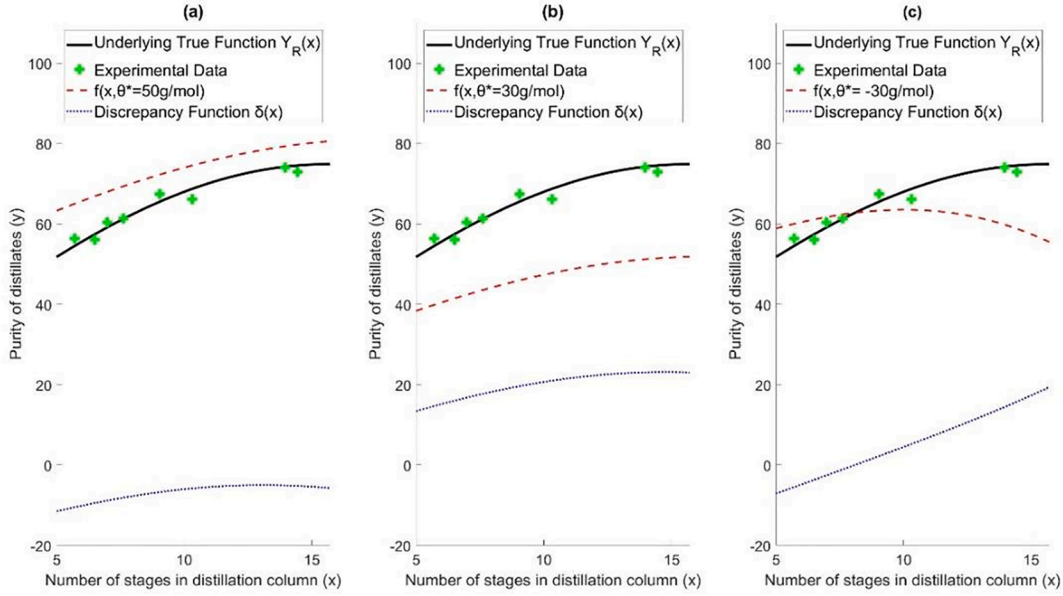

Fig. 8. Identifiability issue in model calibration. (a), (b), and (c) predict the same experimental data with different combinations of posterior calibration parameters θ^* and the discrepancy function $\delta(x;\phi_\delta^*)$.

capability. In addition, the choice of covariance function is a critical decision since it affects the model performance (Rasmussen and Williams 2005; Schulz et al., 2018).

High computational cost is another important issue. In standard GPR, estimation of the posterior includes inverting the matrix $[\Sigma + \sigma_n^2 \mathrm{I}]$, and the corresponding computational complexity is $O(N^3)$ where N is the number of observations. Since the computational cost is cubically proportional to the number of observations, the model becomes computationally expensive for a large dataset. In addition, the amount of data required for reliable analysis grows exponentially as the dimensionality of data increases (Jin et al., 2002; Higdon et al., 2004).

Although GPR is often considered a good smoothing interpolator, it has a limited capability in terms of extrapolation. Different factors including kernel structures can affect the extrapolation ability of GPR models. If the kernel is not comprehensive to capture the true underlying correlations, extrapolation performance would be poor (Wilson et al., 2013). For example, it has been shown that the squared exponential kernel fails to capture the non-local structure of data (Bengio et al., 2005). The extrapolation capability also depends on the extendibility of the discrepancy function, the confidence in the computer simulation and potential incorporation of mechanistic knowledge into the discrepancy function (Higdon et al., 2004). How well the model inadequacy (between experiment and computer model) is captured by the discrepancy function on the extrapolated region is an important issue (Bayarri et al., 2007; Ling et al., 2014).

Different research is conducted to mitigate the drawbacks of model calibration. To increase interpretability, statistical adjustment methods (Joseph and Yan, 2015), or empirical Bayes methods (Joseph and Melkote, 2009) have been proposed. Also, different frameworks for incorporating the discrepancy function have been introduced. For example, Plumlee (2017) proposes a discrepancy function prior that is orthogonal to the gradient of the computer model, which shows improved behavior of the posterior distribution. Sargsyan et al. (2015) utilize model parameterizations to capture the discrepancy instead of the additive discrepancy term.

The development of kernels that better reflect reality is an ongoing research question. Duvenaud et al. (2011) introduce a Gaussian Process with additive kernels, which improves interpretability and the extrapolation performance, by considering a possible set of input interactions. Closed-form kernels that involve spectral density with a Gaussian mixture are derived (Wilson and Adams, 2013) for extrapolation and pattern discovery. The physics-informed Kriging method (Yang et al., 2018) is proposed, where different realizations of available data are used for constructing kernels without assuming a specific form for the kernel function. Recently, active research is conducted to increase the scalability of GPR by finding the pseudo-inputs for the dataset (Gramacy and Apley, 2015; L'Heureux et al., 2017; Yan and Qi, 2010; McIntire et al., 2016; Snelson and Ghahramani, 2005; Liu et al., 2018). Further, the MapReduce framework (Dean and Ghemawat, 2008) has been proposed with the aim of accelerating the model calibration framework (Cai and Mahadevan, 2017) when processing big datasets.

MC as a parameter estimation technique has some similarities and differences from competing techniques, such as pure black-box or derivative-free optimization (DFO) (Lunderman et al., 2021; Abbas et al., 2016), or GP-based Bayesian optimization (BO) (Huang et al., 2021). One key difference in DFO approaches, is that the computer simulation is treated like a black-box, and one only wishes to learn what parameter values minimize the mean squared error between the simulation prediction and the available data. Another difference is that uncertainty is rarely explicitly handled using DFO methods, and no discrepancy is considered (i.e., assuming the computer simulation is accurate). Similarly, even though a similarity between MC and BO is the use of GP models, BO is a less hybrid approach because it directly optimizes the parameter values with respect to an output response, while in MC the mechanistic model states can be incorporated.

3.5. Other techniques for merging mechanistic knowledge with data-driven models

In addition to the techniques discussed in this paper, there are other approaches that incorporate mechanistic information within data-driven models. A very common approach is feature engineering, which involves selecting or identifying (using Machine Learning), the appropriate features (or inputs) to be used to fit a Machine Learning model (Dong and Liu, 2018). This approach is particularly influential in fields such as materials science, where large data sets are available without a-priori knowledge of the true physical descriptors that should be used to predict material properties (Butler et al., 2018). In PSE, it is typically likely that the important, controllable inputs to the models are already known. Associated with feature engineering, are techniques that perform subset selection for generalized linear regression, that aim to build generalizable parametric models with additive features that best describe the data (Cozad et al., 2014; Wilson and Sahinidis, 2017). These techniques lead to more interpretable models, when compared to nonparametric techniques such as NNs and GPs, and can be quite powerful and scalable when some *a-priori* knowledge is available to inform the initial "superset" of potential basis functions.

Another relevant research area to hybrid modeling is multifidelity modeling. Multifidelity techniques are used when multiple types of data and/or models are available for the same system (ranging from highly accurate to low-fidelity) and these and the correlation between them are jointly used to generate overall more accurate models (Peherstorfer et al., 2018). This approach is very relevant to hybrid submodeling and model calibration. The concept of developing models that are trained with various forms of fidelity has been often used for surrogate-based optimization, showing that learning from different sources of levels of fidelity of data can expedite the search for an optimal location. Multifidelity techniques have also been directly incorporated within Gaussian Process regression, or Kriging, via the co-Kriging algorithm that trains a GP model using various fidelity sources (Stein and Corsten, 1991; Perdikaris et al., 2016; Perdikaris et al., 2015; Meng and Karniadakis, 2020; Lee et al., 2019).

Finally, there is number of noteworthy contributions that attempt to fuse the physics-based knowledge into data-driven modeling but could not be categorized within the methods outlined in this paper. Specifically, active research is conducted to incorporate physics-based knowledge within Gaussian Process Regression models. For the case where physics-based knowledge is in the form of linear differential equations, it has been shown that the GP model can be constructed in a way to adhere such physics laws. The first approach includes the use of a specific covariance function that meets physical constraints. (Wahlström et al., 2013) embeds divergence and curl-free properties of the magnetic field by introducing a divergence-free kernel. The second approach utilizes the well-known properties of the Gaussian process, such as that the linear transformation of a Gaussian process is also Gaussian process, to embed the physics-based knowledge (Raissi et al., 2017b, 2017a). Särkkä (2011) embeds physics-based knowledge as a prior to GPR, while Jidling et al. (2017) incorporate the linear operator constraints into the covariance function by introducing another linear operator that fulfills known constraints. Lange-Hegermann (2020) incorporates the boundary conditions on a linearly constrained GP. The third approach incorporates domain knowledge by generating different realizations of the physics-based model to construct mean and covariance functions, purely based on the collection of realizations without assuming any covariance structure (Yang et al., 2018; Tipireddy and Tartakovsky, 2018).

4. Applications

4.1. Applications of hybrid sub-modeling

HSMs have been a been applied to a large and growing number of applications within chemical engineering, including modeling,

monitoring, optimization, scale-up and control (von Stosch et al. 2014b). Applications also span most industries that fall under the umbrella of (bio)chemical engineering, including petrochemicals, metallurgy, wastewater treatment, papermaking and pharmaceuticals. Applications in these areas have led to a number of general proposed frameworks (Oliveira, 2004; von Stosch et al., 2012; Abonyi et al., 2002; Sun et al., 2020) for HSMs, again many based on the original work of (Psichogios and Ungar, 1992). For a summary of HSM implementations across different industrial sectors and applications, the reader should consult pertinent reviews (Zendehboudi et al., 2018; von Stosch et al. 2014b; McBride et al., 2020; von Stosch et al. 2014a; Bangi and Kwon, 2020) as well as a recently published book (Glassey and Von Stosch, 2018).

4.2. Applications of physics-informed machine learning

PI-ML models have been formulated to solve the forward and inverse problems in physics. In the inverse problem, data from the solution is given and the model estimates parameter values or conditions. In the forward problem, the model is given all parameters, initial conditions, and boundary conditions and it outputs the solution to the differential equation. Raissi et al. demonstrate their method with case studies such as the Burgers equation, Schrödinger equation and Navier-Stokes equation for fluid flow. The authors extend the PINN approach, to solve fractional advection-diffusion equations (Pang et al., 2019). In their work, finite elements must be combined with the PINN approach relying on automatic differentiation, since the chain rule from integer calculus cannot be applied to fractional calculus. In (Pakravan et al., 2020), a feed forward NN finds PDE parameters which are fed to a custom layer that functions as a PDE solver using finite element methods. This layer is added to the overall NN, allowing for the model to be trained with the PDE solver embedded. In (Yang and Perdikaris, 2018), deep probabilistic models are used to quantify uncertainty in the output of a PINN. This method can be used for verification analysis.

In robotics, physics-informed ML models have been applied to system control. In (Lutter et al., 2019), neural networks are used to estimate inertial and force matrices of mechanical systems and the outputs of the neural network are combined into the physical equations of conservation (Lagrangian Mechanics). The output is automatically differentiated with a PID controller to give a control response. In power flow modeling, Fioretto et al. (2020) use the Lagrangian dual formulation to enforce Kirchoff's Current Law in NN models that act as surrogates for the AC optimal power flow problem. Misyris et al. (2019) use the PINN formulation to simulate simple power system dynamics. The same methodology has applied to problems in geophysics (He et al., 2020; Kadeethum et al., 2020) and cardiovascular modeling (Kissas et al., 2020). A preliminary study has shown PINNs to be competitive with adjoint methods for PDE based optimal control (Mowlavi and Nabi 2021).

4.3. Applications of model calibration

The model calibration framework and the Bayesian inference is a very flexible and powerful tool, and it is widely used in various fields including energy simulation (Fabrizio and Monetti, 2015; Kim and Park, 2016; Manfren et al., 2013), optical lithography-based manufacturing process (Matsunawa et al., 2015), composite fuselage simulation (Wang et al., 2019), methane air chemistry (Sargsyan et al., 2015), CO_2 capture (Bhat et al., 2017; Kalyanaraman et al., 2015; Kalyanaraman et al., 2016; Li et al., 2017), fluid dynamics (Tagade et al., 2013), and Li-ion cell operation (Tagade et al., 2016).

5. Computational algorithms and software implementations

Due to the large range of potential HSM structures that are dependent on the scenario and the characteristics of the mechanistic SM, automated off-the shelf algorithms for HSM are not available. However, there are many recent developments in computational packages that significantly expedite the training of HSMs. Specifically, to fit a surrogate model (NN or GP) within a differential equation, any DE solver that tracks parameter sensitivities (i.e., gradients) can be used. However, parameter estimation of DEs using the sensitivity equations tends to become inefficient when the number of parameters is large (>100) (Rackauckas et al., 2018). Thus, recent software that calculates parameter sensitivities using automatic differentiation and adjoint-tracking are promising. Software with these capabilities are limited with notable implementation in Python, TensorFlow and PyTorch, and Julia via SciML. For more complex differential equation systems (i.e., PDEs) finite element software with ML extensions are gaining momentum (Berg and Nystrom, 2017; Mitusch et al., 2021). As the computational gains of these AD-based schemes are demonstrated, implementations in other scientific programming languages are anticipated to become available.

Several of the most notable algorithmic implementations are shown in Table 1 in the area of PI-ML. PINNs and PIGPs are typically trained with software that has native ML models and supports automatic differentiation, such as Tensorflow and Pytorch (Paszke et al., 2019; Abadi et al., 2016). Recently, packages have also been developed in Python (Haghighat and Juanes, 2020a) and Julia (Rackauckas et al., 2020) to create PINN models. In training, gradient-based optimization algorithms are used such as Newton's Method, stochastic gradient decent, Adam, and L-BFGS. Notably, the same software packages enabling PI-ML implementations are envisioned to automate the building HSMs.

For Model Calibration, multiple software packages are available and those are listed in Table 2. These can be used to generate posterior distributions for model parameters based on experiment and simulation data and to make final predictions following a MC structure.

It is worth observing here that many of the software implementations listed in the previous two tables are rely on open-source libraries that are supplemented with full-implementations of reproducible code. The past decade has seen the chemical engineering profession increasingly embrace the use of open-source code as a means of making software implementations more widely available and reproducible. It is likely that continued collaboration between engineers and colleagues in the computer science and machine learning communities will help perpetuate this mutually beneficial trend, accelerating the pace at which flexible, robust code becomes available to those who could benefit from its use. Echoing remarks in Schäfer et al. (2020) a key goal of these collaborations should be to streamline the HM workflow such that not only the model training but also the data processing, model simulation and optimization can be performed in a single software environment. Such simplifications will naturally encourage more widespread industrial adoption of HMs.

Table 1
Algorithmic implementation of PI-ML algorithms.

Name	Algorithm	Language	Ref.
PINNs	Physics-Informed Neural Networks	Python	Raissi (2019)
SciML	Physics-Informed NN for PDEs	Julia	Rackauckas et al. (2020)
Lagrangian Dual	Training PI-NNs using Lagrangian Duality	Python	Freund (2004)
SciANN	Physics-Informed NNs for PDEs in Python	Python	Haghighat and Juanes (2020a)
torchdiffeq	Neural Ordinary Differential Equations	Python	Chen et al. (2018)

Table 2
Model calibration packages.

Name	Algorithm	Language	Reference
BACCO	Bayesian Calibration of Computer Codes	R	Hankin (2005)
SAVE	Statistical Analysis of Computer Models	R	Palomo et al. (2015)
PyMC3	Probabilistic Programming Library	Python	Salvatier et al. (2016)
GPM/SA	Gaussian Process Models for Simulation Analysis	Matlab	Gattiker et al. (2015)
CaliCo	Bayesian Calibration	R	Carmassi et al. (2018)
RobustCalibration	Full Bayesian Analysis for Model Calibration	R	GU (2018)

6. Motivating example

6.1. Case study description

In order to further elucidate how different formulations of hybrid models can be applied to systems familiar to the PSE community, we use a classic example in chemical engineering: an adiabatic plug flow reactor (PFR) (Snyder and Subramaniam, 1994; Fogler, 1999). In addition to being a fundamental unit operation in numerous chemical processes, the reactor system is associated with several modeling tasks that may be approached through a hybrid modeling lens. In the context of model calibration, kinetic and thermodynamic parameters inherent to the first-principles model must frequently be calibrated to reflect real data gathered from the reactor concentrations and temperatures. Similarly, data-driven models can be used to approximate certain parts of the model that may be not well described from physics, such as an equilibrium constant. Finally, the underlying differential equation model provides a basis for a PINN formulation, resulting in a full surrogate model that better generalizes to new points.

Specifically, we will look at the production of styrene from ethylbenzene (EB) with two side reactions (Rxns. (1)–(3)).

$$Ethylbenzene \leftrightarrow Styrene + H_2 \quad \text{(Rxn 1)}$$

$$Ethylbenzene \rightarrow Benzene + Ethylene \quad \text{(Rxn 2)}$$

$$Ethylbenzene + H_2 \rightarrow Toluene + Methane \quad \text{(Rxn 3)}$$

The reversible rate law for Rxn 1, as well as the irreversible side reaction rate laws Rxn (2)–(3) are given below with corresponding units (kmol/ m^3/s):

$$r_1 = \rho(1-\phi)exp\left(-0.08539 - \frac{10,925}{T}\right)\left(P_{EB} - \frac{P_{St}P_{H_2}}{K_{p1}}\right) \quad (26)$$

$$r_2 = \rho(1-\phi)exp\left(13.2392 - \frac{25,000}{T}\right)(P_{EB}) \quad (27)$$

$$r_3 = \rho(1-\phi)exp\left(0.2961 - \frac{11,000}{T}\right)(P_{EB}P_{H_2}) \quad (28)$$

These rate laws can then be reformulated to mole balances on all the species in the system, and combined with an energy balance to form the system of coupled differential equations shown below:

$$\frac{dF_{EB}}{dV} = -r_1 - r_2 - r_3 \quad (29)$$

$$\frac{dF_{St}}{dV} = r_1 \quad (30)$$

$$\frac{dF_{H_2}}{dV} = r_1 - r_3 \quad (31)$$

$$\frac{dF_{Be}}{dV} = \frac{dF_{Et}}{dV} = r_2 \quad (32)$$

$$\frac{dF_{Tol}}{dV} = \frac{dF_{Meth}}{dV} = r_3 \quad (33)$$

$$\frac{dT}{dV} = \frac{-r_1\Delta H_{rxn1} - r_2\Delta H_{rxn2} - r_3\Delta H_{rxn3}}{F_{EB}Cp_{EB} + F_{St}Cp_{St} + F_{H_2}Cp_{H_2} + F_{Be}Cp_{Be} + F_{Et}Cp_{Et} + F_{Tol}Cp_{Tol} + F_{Meth}Cp_{Meth} + F_{Steam}Cp_{Steam}} \quad (34)$$

In Eqs. (29)–(34), F_i and P_i represent the flow rate and partial pressure of each chemical species i, respectively. T is the temperature and r the reaction rates. The independent variable V represents an increment of reactor volume and increases linearly with axial distance down the reactor unit. For an explanation of all other parameters in the model and their values, the reader is referred to the Appendix. Data for each framework is collected by simulating the mechanistic equations above for 6 separate runs, which vary using initial temperatures in the range $T \in [850,950]$ K and flow rates of the reacting species in the range $F \in [0,5]$ mol/m^3. In addition, an inert mixture of steam enters the reactor at a flow rate of 48 mol/m^3, acting as a heat sink. The reactor is assumed to operate at a total pressure $P_T = 2.4$ atm with negligible pressure drop. Thus partial pressures can be calculated as $P_i = (F_i/F_T)P_T$ where F_T is the sum of flow rates of all chemical species. We will use this PFR system to demonstrate three methods: 1) Semi-parametric hybrid modeling, 2) Physics-Informed Machine Learning, 3) Model Calibration below.

6.2. Hybrid modeling

To illustrate the difference in HSM approaches, we visit the ethylbenzene reactor example when insufficient mechanistic knowledge limits model performance. The data-driven model used for illustrating each of the HSM approaches is a feed-forward neural network with one hidden layer and 10 hidden nodes. However, other nonlinear models could be used. To explore the nuances between mechanism correction and estimation, we consider scenarios wherein missing thermodynamic or kinetic information prohibit a fully mechanistic modeling approach.

Unknown Energy Balance

First, we consider the scenario where a model for ethylbenzene conversion is available but fails to accurately capture system

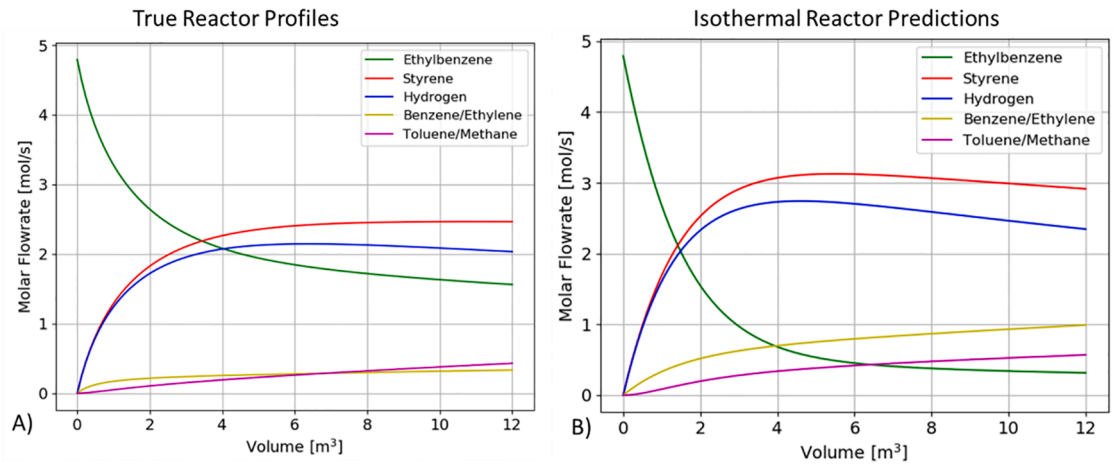

Fig. 9. True (Left) vs low-fidelity (Right) simulation of ethylbenzene conversion to styrene.

performance.

Fig. 9A shows the true progression of EB conversion in a tubular reactor assumed to be operating in plug-flow. The true temperature and flow rate profiles can be generated using the full set of equations found in the Case Study Description. In contrast, a low-fidelity mechanistic model which assumes isothermal conditions predicts the conversion profile in Fig. 9B. The low-fidelity predictions are simulated using the same mechanistic equations as in the true model, except Eq. (33) is replaced with $dT/dt = 0$. The task of this illustrative example is to improve the predictions of the low-fidelity mechanistic model.

A mechanism-estimation approach to model the unknown thermodynamics is to model the energy balance with a data-driven SM. This data-driven SM can be formulated as follows:

$$\frac{dT}{dt} = DD(T, F_k, \phi) \tag{35}$$

Here the data-driven SM $DD()$ represents a neural network (NN) with 6 inputs, which include temperature T and the $K = 5$ flow rates F_k of different chemical species in the reactor, where $k = 1,\ldots,K$. The NN has a single output dT/dt. The NN parameters are fitted by repeatedly integrating Eq. (35) with other mechanistic equations of the model, calculating the error in temperature predictions and updating the parameters of the NN using the error gradients until convergence. Note that this approach is feasible only by knowing the energy balance is the improperly formulated relationship.

An HSM for mechanism correction offers another approach to correcting the mechanistic model (Eqs. (11) and (12)). The data-driven SM is trained and simulated separately from the mechanistic model. The correction framework indexed for the EB system is outlined below.

$$\frac{dx_k}{dt} = f(x_k, c(t), \Theta) \tag{36}$$

$$x_{\Delta t+t,k} = DD\left(x_{t,k}, c_t, \phi\right) \tag{37}$$

Here $x_{t,k}$ represents each state variable k at time t, ϕ the parameters of the data-driven SM (i.e., the weights of a neural network) and Θ mechanistic parameters (i.e., activation energy). $c(t)$ could potentially represent a forcing variable such as a heating or cooling term, but these were not considered in this simple example. Training of the data-driven model is done using the flow rate and temperature predictions from the mechanistic model at position t as the inputs and the true concentrations/temperatures at position $t + dt$ as the outputs. In total, the data-driven SM therefore has 6 inputs and 6 outputs.

The performance of the two methods is depicted in Fig. 10. The predictions of both modeling methods are nearly identical and clearly improve the accuracy of the incomplete mechanistic model. The final mean squared error (MSE) between model predictions is slightly higher for the mechanism estimation approach than the mechanism hybrid correction approach: MSE = 3.396 mol/s vs MSE = 0.0775 mol/s, respectively. However, this should not be interpreted that the

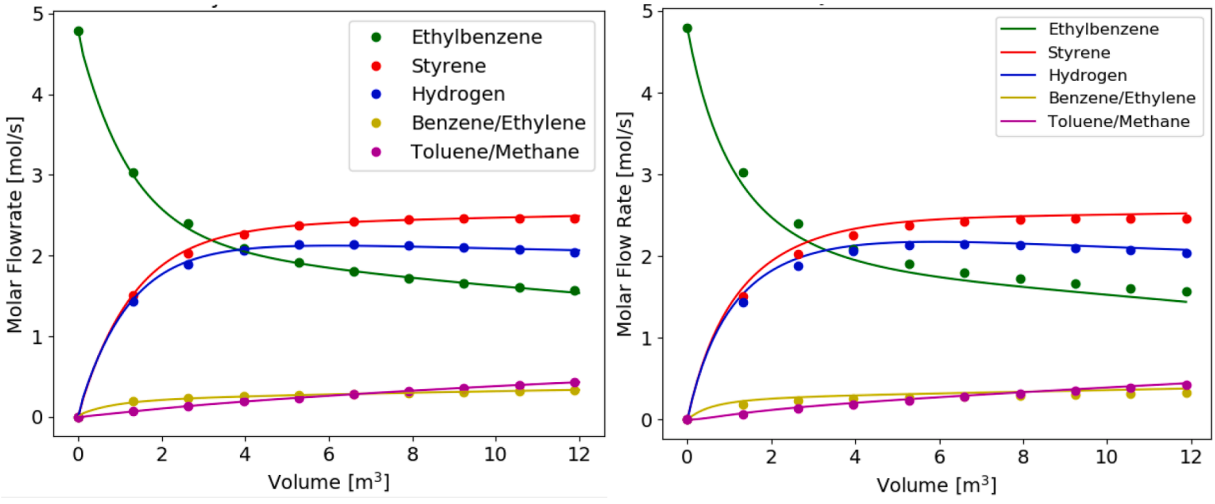

Fig. 10. Regression of hybrid correction model (left) and hybrid estimation model (right). HM predictions represented by solid lines. Data used for regression represented by points.

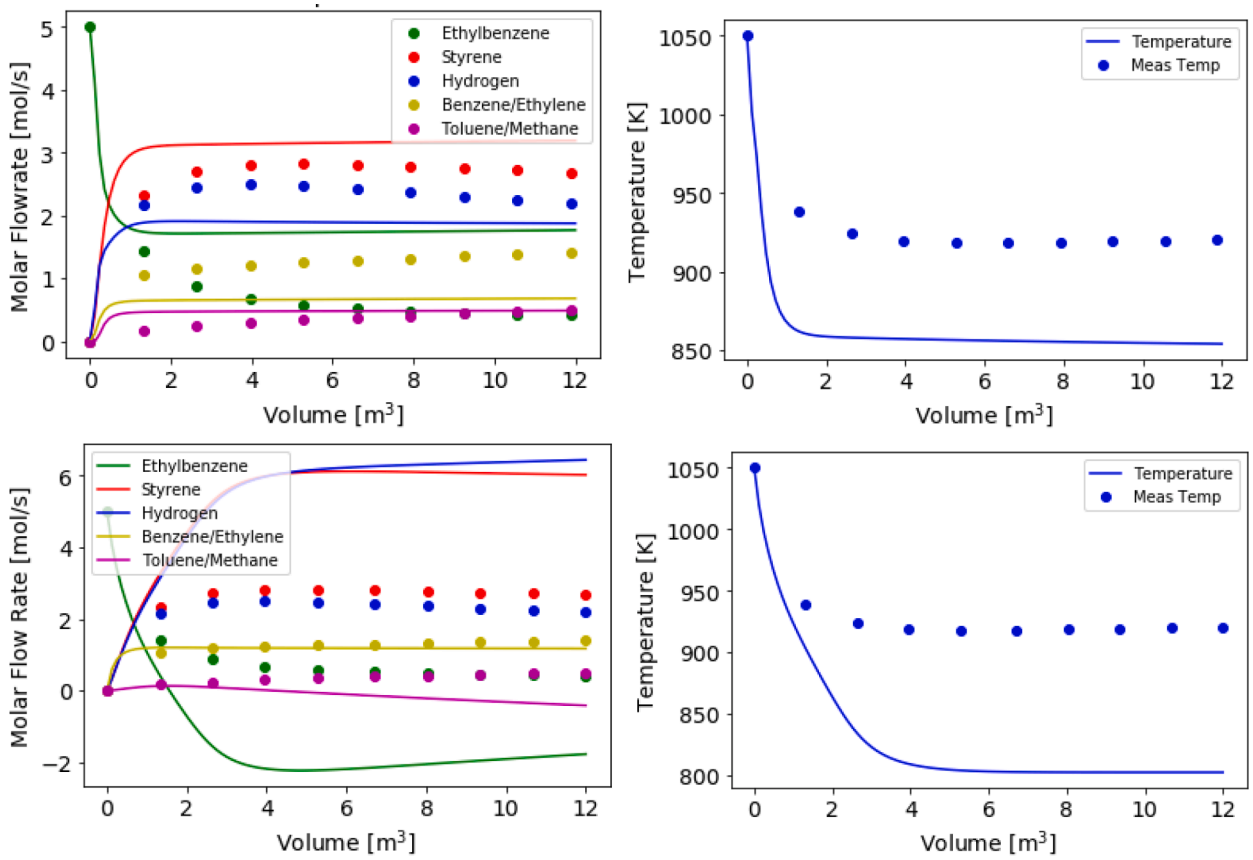

Fig. 11. Predictions of the hybrid correction (top) and hybrid estimation (bottom) of unknown energy balance for extrapolating conditions. HM predictions represented by solid lines. True data represented by points.

mechanism correction framework offers more accurate predictions in the general case. Rather than accuracy, in this case the advantage of the estimation approach over the correction scheme is the clearly defined phenomenological relationship the data-driven model is predicting. This model structure may be useful if it is later desired to propose a mechanistic formula for correcting the low-fidelity model. The data-driven model predictions could be used to regress parameters of the mechanistic model separately from the other equations in the model (Bradley and Boukouvala, 2021) enabling computationally tractable parameter estimation. A possible advantage of the correction framework is that there is no need for simulating the mechanistic model during estimation of the data-driven model parameters, which may be computationally demanding. While the differential formula for this model was simple, estimating data-driven parameters in a more complex PDE may not be tractable.

However, while the two methods offer accurate estimates for interpolating conditions, the same accuracy should not be expected for extrapolating conditions. Shown in Fig. 11 is the attempt to use either method for prediction when the inlet reactor is at a higher temperature of 1050 K. As seen in Fig. 11, since the data-driven component of the HMs rely strongly on temperature, neither method captures the true profile. Additional data would be required to train the HMs in the hotter reactor conditions prior to reliably using the HMs for predicting reactor performance.

Rate estimation

We also consider the scenario where one of the mechanistic model's

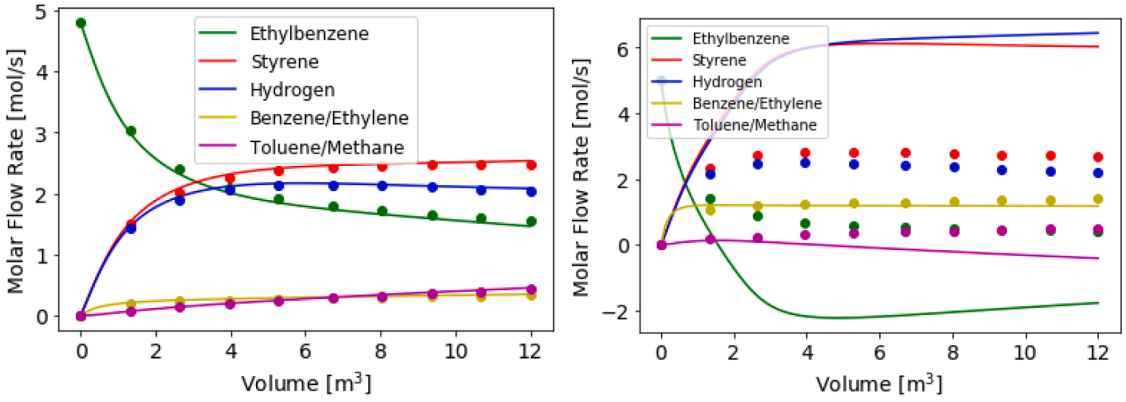

Fig. 12. HM predictions vs training data (left) and extrapolation data (right). DD model estimates Reaction 1. HM predictions represented by solid lines. True data represented by points.

kinetic rates (i.e., Eqs. (26)–(28)) are unknown. Such situations may arise when a reaction mechanism is too complex to model via first principles knowledge or the kinetics have shifted due to catalyst deactivation or equipment degradation. Without a formulation for the kinetic rates, simulation of the remaining mechanistic equations yields no useful information, so a mechanism correction formulation is not viable. Nevertheless, using a mechanism estimation formulation, the rate relationships can be modeled with a data-driven model, which here again is a Neural Network.

$$r = DD(T, F, \phi) \tag{38}$$

To keep this illustration simple, the data-driven part of the hybrid DE estimates either Rxn 1 or Rxn 3, assuming the kinetics are unknown. The Neural Network receives the 6 state variables as inputs and estimates a single unknown reaction, whereas the remaining two reactions are assumed known and follow the rate laws of Eqs. (26)–(28).

The fit of the HM when Reaction 1 is unknown is depicted in Fig. 12. As when the NN predicted the energy balance, the HM offers a satisfactory estimate of the training data and can be relied on to estimate conditions that are an interpolation of the training data. However, also depicted in Fig. 12 are the HM estimates for the case where in inlet temperature is higher ($T = 1050$ K) than during training. Once again, the HM fails to capture the faster kinetic profiles, predicting spurious relationships. Undoubtedly, the poor extrapolation of this HM is due to the strongly coupled nature of the differential equations. Since Reaction 1 strongly depends on temperature and the other reaction depend on the reactants of Reaction 1, poor prediction of this reaction results in bad estimation of all chemical species.

So far accuracy of the HMs have been similar to what would be expected of a purely data-driven model. The fitted model predicts well within the range of training data but not well outside this range. To offer a counter-example to this trend, the last case considers the situation where only Reaction 3 is unknown. Fig. 13 shows the fit of the HM when the Neural Network estimates the production of toluene and methane. Again, the HM fits the data well within the range of training data. More interestingly, the HM also predicts well the species concentration for a higher inlet temperature. This result is explained by the smaller role of the data-driven model in the HM. Although Reaction 3 is temperature dependent, it is a side reaction whose product concentrations do not play a role in other system reactions. Instead, the HM relies primarily on the correctly formulated mechanistic part of the model when extrapolating. While not always possible, limiting the role of the data-driven model increases the likelihood the HM will be reliable for unseen system conditions, which is the golden standard for modeling and perhaps the truest measure of model generalizability.

6.3. Physics-informed neural network

In this Section, we illustrate the merits of using a PINN surrogate in lieu of a standard black-box NN model. Here we can use the underlying mechanistic knowledge of the system to improve ML predictions outside of training points for cases where interpolation or extrapolation is necessary. Using the PINN structure described in Section 4.3, we can train our model using sparse data as well as adherence to the underlying mechanistic equations given by the ODE system. Fig. 14 below summarizes the inputs, outputs, and relevant physics relationships.

Importantly, the NN outputs are differentiated with respect to the input reactor coordinate (V). Using automatic differentiation, the resulting derivative values can be compared to Eqs. (29)–(34). Critically, this can be done at unlabeled data points, allowing for good surrogates with very few labeled data points. This is shown in the loss function below, where labeled and unlabeled data are denoted N_{sample} and N_{colloc}, respectively. Model predictions are given by y_i and ground truth by $\widetilde{y}_i$.

$$Loss = \frac{\mu}{N_{samples}} \sum_{i=1}^{N_{samples}} (y_i - \widetilde{y}_i)^2 + \frac{(1-\mu)}{N_{colloc}} \sum_{i=1}^{N_{colloc}} \left(\frac{dy_i}{dV} - \frac{d\widetilde{y}_i}{dV} \right)^2 \tag{39}$$

For the sake of comparison, a classic NN model is fit to the same data under identical training conditions. Some results are given below to illustrate the effect of the physics-based loss term. First, the two models are tested under very sparse data conditions, where we only have training points for inlet, outlet and midway concentrations ($y(V = 0, 6, 12)$). For the PINN, we also select 100 collocation points randomly throughout the V domain. In Fig. 15, the results for ethylbenzene concentration profiles for each model are depicted (Fig. 16).

While both models are able to fit the training data exactly, the PI-NN model has improved interpolating behavior than the black-box NN, especially in the more non-linear region ($V = [0, 6]$). This is due to the physics constraint applied at collocation points throughout the domain. We can also design an experiment for extrapolation. Here we provide training data at $V = 2, 6, 10$ and predict over $V = [0, 12]$.

Again, both models are able to predict points where training data is present, but only the PI-NN can extrapolate outside that region because it has mechanistic knowledge applied at collocation points during training. From the above results, it is clear that the PINN approach offers advantages over the purely black-box approach, however, it requires that accurate physics-based information is available. In this example, we have assumed a fully-mechanistic model is available and have built a NN surrogate that is trained with embedded mechanistic knowledge. A similar approach can be applied even if the mechanistic knowledge is partially available. For example, the physics-constraint violation term could include violation from a mass-balance constraint, or an energy balance constraint, if those are the only mechanistic equations that are

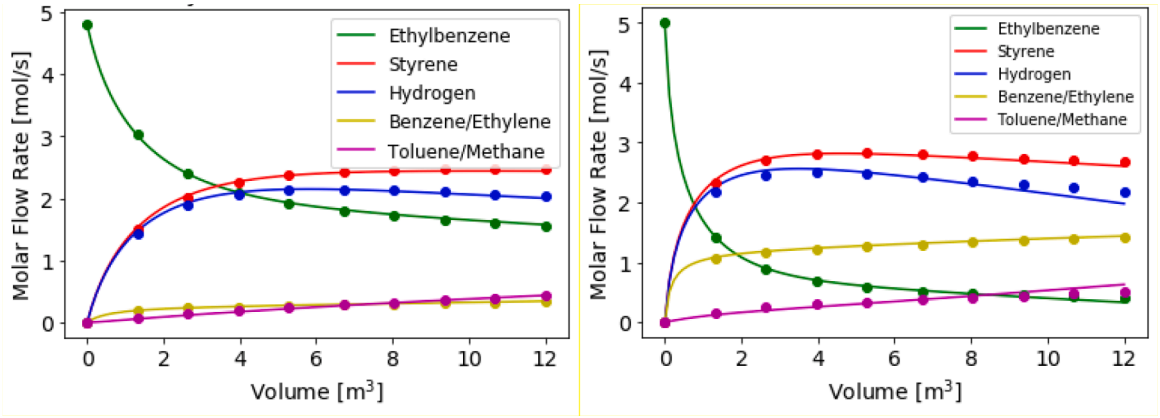

Fig. 13. HM predictions vs training data (left) and extrapolation data (right). DD model estimates Reaction 3. HM predictions represented by solid lines. True data represented by points.

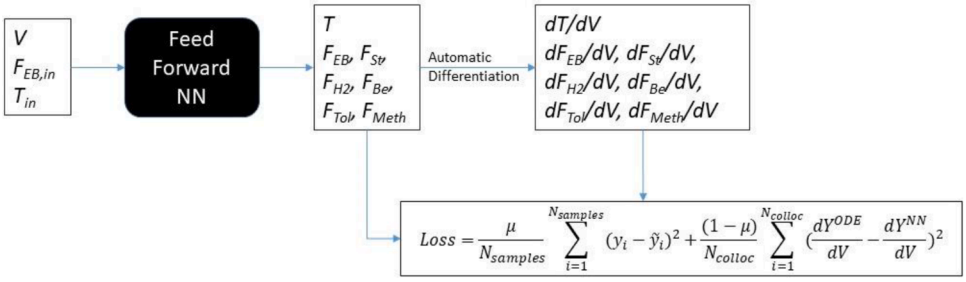

Fig. 14. PINN Representation of Ethylbenzene Reactor with Loss Function Formulation.

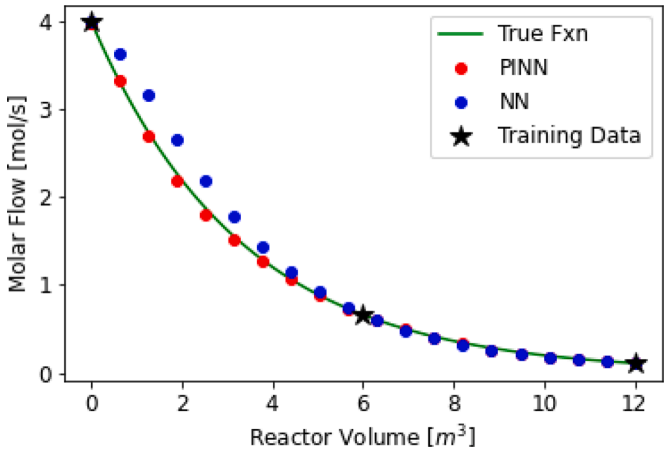

Fig. 15. Comparison of true simulation prediction (data) with black-box NN (NN), and Physics-Informed NN (PINN) predictions for Interpolation between training points.

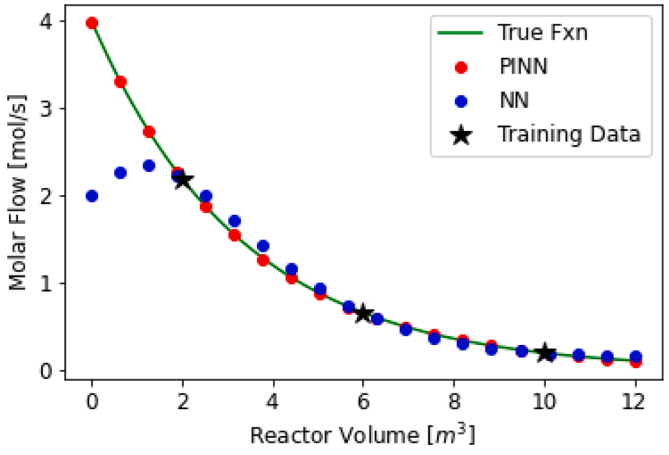

Fig. 16. Comparison of true simulation prediction (data) with black-box NN (NN), and Physics-Informed NN (PINN) predictions for Extrapolation from training points.

available. The difference between PINN and NN predictions would be further pronounced in applications with noisy data, as the differential constrain is essentially a regularization term that penalizes the objective function and reduces over-fitting and noise-fitting.

6.4. Model calibration

Model calibration requires two datasets: high-fidelity experimental data from a physical experiment and low-fidelity simulation data from a computer experiment. Since experimental data is not available for this

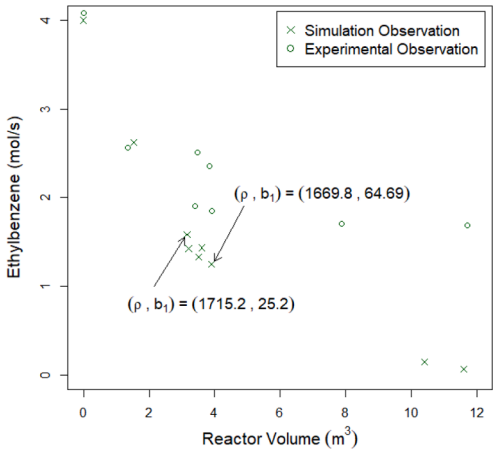

Fig. 17. Different guesses of calibration parameters to run computer simulation.

case study, we generated a high-fidelity dataset from the full Ethylbenzene reactor model. In addition, we applied the simplifying assumption: $\frac{dT}{dV} = 0$ to the Ethylbenzene reactor model and generated a low-fidelity simulation dataset. Since model calibration is widely used for the system where collecting data is expensive, we used a small dataset (9 low-fidelity simulation data Y_s and 8 high-fidelity observation data Y) for calibrating the model. Zero-mean Gaussian noise is added to the high-fidelity experimental data Y to see how the model calibration handles this scenario. We consider a *single variable input* (V: reactor

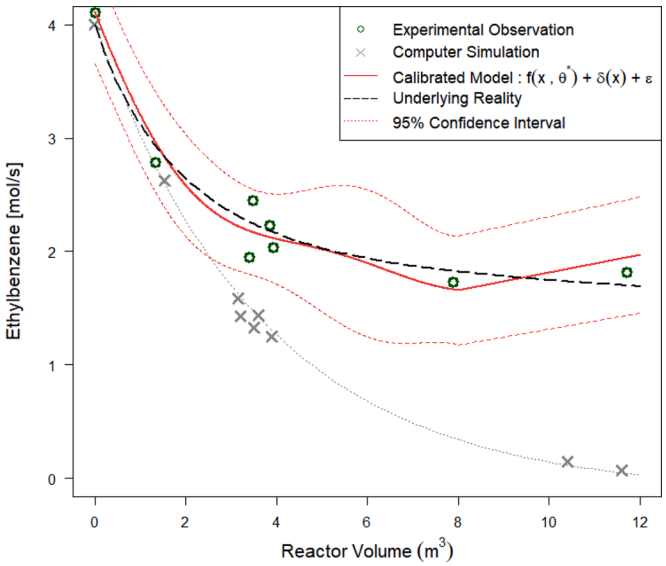

Fig. 18. Prediction performance of the calibrated model.

volume) – *single output* ($F_{Ethylbenzene}$: molar flow rate of ethylbenzene) system with *two calibration parameters* (ρ and b_1). Note that calibration parameters can be a physically interpretable parameter (ρ) or a tuning parameter (b_1) which has little to no physical meaning. We assume that domain knowledge is available for calibration parameters and that we know that there is a 95% chance of ρ be in a range of [1700, 2100] and b_1 be in a range of [−50,50]. By utilizing this domain knowledge, we sample calibration parameters $(\rho,\ b_1)$ from the prior distribution $\rho \sim N(1900,\ 100^2)$ and $b_1 \sim N(0,\ 25^2)$. Fig. 17 shows different runs of low-fidelity simulation data for different values of calibration parameters and the high-fidelity experimental data sampled for training the model. Note that the true value of calibration parameters is $\theta^* = (\rho,\ b_1) = (2137,\ -17.34)$ (listed in Appendix).

The modular Bayesian approach (Arendt et al., 2012) (Fig. 6) is applied for model calibration and the MCMC algorithm is used to sample the posterior predictive distribution of GP posterior hyperparameters (ϕ^*) and calibration parameters (θ^*). During the model calibration process, the variable input space and the calibration parameter space are scaled between [0,1] for numerical stability (Hinton and Rasmussen, 1997).

Fig. 18 shows that the low-fidelity computer model is calibrated to predict the high-fidelity experimental data with high accuracy, and all experimental observations lie within the 95% confidence interval of the calibrated model. Note that uncertainty is larger in the region where we have sparse observations, and one may perform additional experiments around the high uncertainty region. Likewise, investigation of uncertainty can help modelers to do informative experimental design.

Fig. 19 shows how each term in the calibrated model behaves in predicting high-fidelity experimental observations. It is observed that the $f(x, \theta^*; \phi_f^*)$ term (i.e., computer model replaced with GP) alone does not predict high-fidelity experimental observations well, while the discrepancy is mostly captured by the discrepancy function. The perfect agreement between the low-fidelity computer model and the high-fidelity experimental data is not guaranteed even when true calibration parameters are found, because the computer model itself $f(\cdot)$ is an approximation with simplifying assumptions (e.g., $\frac{dT}{dV} = 0$).

In this case study, the posterior mean values of calibrated parameters are estimated as $(\rho_{calibrated},\ b_{1,\ calibrated}) = (1846,\ 8.6)$, which do not converge to the true value $(\rho_{true},\ b_{1,true}) = (2137,\ -17.34)$. In fact, different combinations of calibration parameters $\theta^* = (\rho,\ b_1)$ and the hyperparameters of GP $\phi^* = [\phi_f^*, \phi_\delta^*, \sigma_m^{2*}]$ are possible for explaining the same system accurately. It thus becomes important to incorporate any prior knowledge (e.g., parameter priors, level of noise), into the training process as it is not guaranteed that the distinction between true response, true discrepancy and noise will be identifiable.

7. Perspectives and future outlook

7.1. Comparison between HSM and model calibration

Having reviewed the current state of HSM and model calibration, this perspectives piece now offers us the unique opportunity to weigh the relative merits of the two. In certain respects, model calibration can be summarized as the non-deterministic equivalent of hybrid models aimed at mechanism correction. They both seek to improve the performance of underperforming engineering model, generally by modeling the discrepancy between the engineering model and experimental data. Yet despite their parallel development and similar goals little attention has been given to juxtaposing the utility of the two options.

The model calibration formulation differs from hybrid mechanism correction in that it explicitly incorporates the effect of measurement noise. Moreover, in model calibration, the parameters in the mechanistic model can be seen as updated (from prior to posterior) conditioning on the multi-fidelity data, based on the Bayesian scheme. The parameters of the mechanistic model could be updated in a mechanism correction framework, but such cases are less frequent.

The obvious advantage of the Bayesian scheme is the straightforward interpretation of uncertainty information, streamlining conclusions that are based on variable sensitivity and prediction intervals. In contrast, the deterministic correction scheme has been favored in engineering circles largely for its lower-bar in terms of technical know-how—there being no requirement to specify prior distributions, sampling schemes, or interpreting posterior probabilities. In addition, by avoiding Gaussian Processes, the deterministic mechanism correction scheme avoids the cubically-increasing compute cost frequently cited to be a problem when

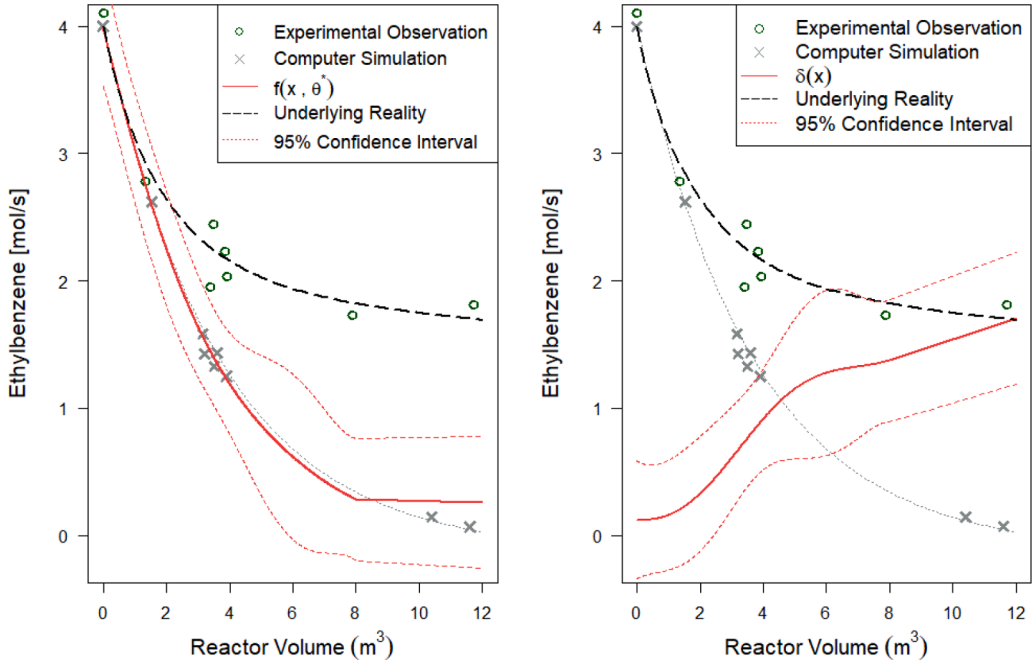

Fig. 19. The behavior of each term in Model Calibration. (Left): Computer model replaced with GP $f(x, \theta^*; \phi_f^*)$ (Right): Discrepancy function $\delta(x; \phi_\delta^*)$. Here, θ^* is the posterior calibration parameters and ϕ_f^* and ϕ_δ^* are the GP posterior hyperparameters to the computer model f and the discrepancy function δ, respectively.

calibrating models with large amounts of data.

Model Correction HSM methods and MC approaches both include the incorporation and training of an "error" or "discrepancy" function. An important debate can be made on how adding the correction model compares to finding an improved or augmented mechanistic model, by adding more parameters within the mechanistic model. Of course, if the available knowledge and resources exist to improve the mechanistic model to mimic reality, this should be preferred. Often this could be done in a manual trial-and-error approach, or via automated methods for model discrimination (e.g., see Olofsson et al., 2018) . However, the hybrid structures discussed in this paper are most appropriate for cases where this is either impossible (e.g., when a high-fidelity model is a black-box simulation that cannot be accessed or edited), or impractical at the given time (e.g., fully mechanistic model development is not possible due to missing knowledge, lack of time and computational resources).When modeling the discrepancy, both methods suffer from the same limitations on interpretability of the discrepancy model and low extrapolation accuracy. When these are demanded, the modeler may ultimately be required to revert to mechanistic approaches that modify the engineering model in ways that yield additional fundamental knowledge about the system (Joseph and Melkote, 2009).

7.2. Extrapolation, identifiability and interpretability

For a model to be useful, it must predict with reasonable accuracy system conditions different from those considered during model training. These predictions may fall within the range of data used during training (i.e. interpolation) beyond the range of training data (i.e. extrapolation). Each of the hybrid frameworks have been demonstrated to offer superior predictions than a purely data-driven model, (Lee et al., 2002; Van Can et al., 1996; Kennedy and O'Hagan, 2001b) at least for interpolation tasks. In addition, in cases where the mechanistic model is incomplete or poorly formulated, simply refitting the mechanistic model has been shown to underperform a hybrid model based on model calibration (Kennedy and O'Hagan, 2001b), correction (Aguiar and Filho, 2001; Sun et al., 2020; Keskitalo and Leiviskä, 2015), or estimation (Van Can et al., 1998; Georgieva et al., 2003). However, the improved generalizability of hybrid approaches over a purely data-driven approach should not be confused with the extrapolation potential of purely mechanistic models. When a mechanistic model is well-formulated, the physical meaning (i.e., interpretability) of its parameter values can be leveraged to hypothesize how the mechanistic model will perform on an entirely new set of conditions. For the majority of the hybrid frameworks in this survey, the final output model is a data-driven model. Thus, their generalizability is likewise confined to the conditions covered by the training data. For example, hybrid correction models are known to have the same limited generalizability as a purely data-driven model (Van Can et al., 1996; van Can et al., 1997) and are thus rarely applied when good model extrapolation is required, such as in system optimization (Yang et al., 2020). Improving their generalizability requires either enlarging the range of conditions used to sample data for model-building or more efficient coverage of the conditions used for training—for example, through careful design of experiments.

A notable exception to the general view above is the hybrid estimation framework. The HM estimation scheme aims to limit the application of the data-driven model to only the data regions and model functions that require adjustment or mechanistic knowledge is missing. As a consequence, HSMs based on mechanism estimation have been shown to have some ability to predict accurately beyond observed data (i.e., extrapolation). This makes this formulation far more attractive to applications in optimization. However, this extrapolation potential is contingent upon HM predictions relying on the accurately specified mechanistic relationships when extrapolating (Braake et al., 1998; Van Can et al., 1998). An excellent illustration of this principle can be found in (Yang et al., 2011). These authors modeled an unknown kinetic rate in the toluene nitration process via a data-driven SM while assuming a mechanistic SM for mass transfer is known. Since the kinetic rate was not dependent on the concentration or volume ratio of toluene these process variables could be changed without a significant decrease in the HM accuracy. Nevertheless, there are cases where the extrapolation accuracy of the hybrid estimation model may resemble that of a purely data-driven approach. For example, if a process model consists of highly coupled nonlinear conservation balance equations, separating the effects of data-driven and mechanistic SMs is not always obvious or possible (Quaghebeur et al., 2021). For such scenarios, mechanistic correction and estimation frameworks may be indistinguishable and perform equally well (see Section 7.1 of the Motivating Example in this work, for example). More work is needed to delineate how or even when SMs may be merged to preserve interpretability, such as a recent graphical framework proposed in Lee et al. (2020).

In the case of Physics-Informed ML, while embedded physical knowledge cannot eliminate these challenges, several literature studies (Raissi et al., 2019; Kim et al., 2020; Fraces et al., 2020) demonstrate appreciable improvements for the extrapolation ability of the PI-ML models by acting as a regularization term. In (Haghighat and Juanes, 2020b), a simple PDE model example clearly illustrates this fundamental trade-off. Here we use a NN to fit an "unknown" function: $f(x, y) = \sin(x)\sin(y)$. We know the data should follow this underlying PDE: $f_{xx} + f_{yy} + 2f = 0$, therefore, we can formulate a NN that learns the function solely from data points and a PINN which embeds the PDE knowledge. We only have data for the domain x,$y = [-\pi,\pi]$ but want to make predictions over $[-2\pi,2\pi]$. Using identical NN structures, data, and training epochs, the performance of both methods is compared in Fig. 20.

In both, the functional representation is distorted the farther away from the training domain (shown as inner four squares above). However, the PINN clearly better represents the pattern outside of the domain compared to the NN. A less obvious point is that the NN does better within the training regime because it is overfitting. Another observation frequently absent in the literature is that not all modes of extrapolation are equivalent. While in the above example, PINNs show good generalization beyond the range of spatial coordinates used during training, in general their ability to rigorously extrapolate to new situations, say with different initial or boundary conditions, is limited. This observation can be explained by the use of independent variables (often spatial and time coordinates), which confines its validity to a particular spatio-temporal trajectory. Further research is needed to systematically explore which mathematical functions are most helpful for regularization/extrapolation and if non-differential functions can provide similar benefits.

7.3. Uncertainty quantification

Developing standard methods for quantifying the uncertainty of the assumptions in hybrid frameworks remains an open problem. Studies that characterize the reliability, identifiability, and sensitivity of HSMs would be a useful step in this direction. Optimization of hybrid models

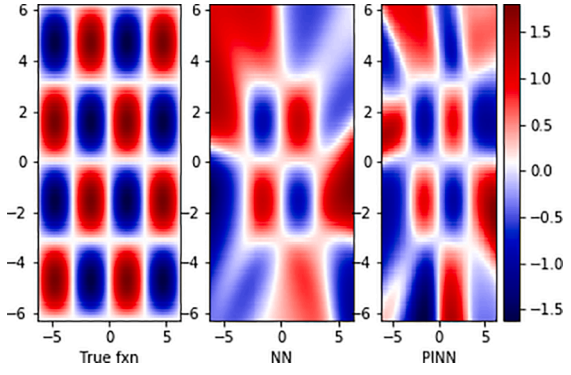

Fig. 20. Extrapolation Ability of PINNs (Haghighat and Juanes 2020a).

using uncertainty information was tackled in (Kahrs and Marquardt, 2007) for algebraic systems and recently extended to dynamic systems by Bae et al. (2020). They established two criteria to measure the validity space: 1) a convex-hull criteria created a region around the measured data to define a space of trusted inputs to the hybrid model and 2) a confidence interval criteria constrained the HM's final predictions. Both could be added to the optimization formulation, either as constraints or as part of the objective function, to limit the predictions of the data-driven model in regions where the data-driven model was required to extrapolate. However, as noted by the authors in Kahrs and Marquardt (2007), the convex-hull fails to be informative when data is sparse. A more nonlinear characterization of the data range used for training was demonstrated through a clustering technique (Simutis et al., 1995; Teixeira et al., 2005). By assuming a Gaussian distribution of cluster points, Simutis et al. were able define a criterion for the extrapolation of both the input and output space. More recently, Pinto et al. (2019) used a bagging technique to fit an ensemble of NNs, which enabled construction of a confidence interval for the final HM. Another promising approach is to use topological data analysis along with a nonlinear classifier to construct validity regions. In (Schweidtmann et al., 2021), Schweidtmann's group showed how these validity regions could constrain data-driven models during optimization, suggesting that the method would be similar for hybrid models. Finally, recent work applied global sensitivity analysis to diagnose the uncertainty of the HM with the notable result that the true sensitivities of the mechanistic model could not be recovered when the data-driven Neural Network was integrated with the differential equations (Francis-Xavier et al., 2021). Additional research is needed to understand how to apply these UQ techniques for model adaption and optimization for models with multiple SMs. Further progress in characterizing uncertainty for hybrid submodeling frameworks will likely result from employing statistical techniques previously developed from the Bayesian perspective of the model calibration framework.

In fact, the model calibration scheme may offer the most comprehensive treatment of uncertainty. This may be useful in, for example, an experimental design program, especially when the simulation or experiment is expensive. By using a Bayesian-based approach with design validation metrics (Chen et al., 2007), the modeler may determine what system conditions will yield the most informative design under a framework similar to model calibration. Recently, the model calibration with the emphasis on the uncertainty is discussed under three problems the forward, inverse, and the validation problem (Lee et al., 2019). Another use case where the calibration emphasis on distribution can be advantageous is when we can take domain knowledge into account as a form of prior settings for different parameters and discrepancy function. This offers a more straightforward interpretation of the discrepancy model and hyperparameters based on original confidence in the mechanistic model parameters. However, as previously stated, when little information is known *a priori*, setting prior distributions can be difficult. If little prior information is available, many modelers may choose a simple prior distribution (e.g., uniform distribution) (Chen and Wang, 2017).

7.4. Hard constraints, soft constraints, gradient pathologies in PI-ML

One important distinction to draw from existing literature in PI-ML and for future work in this area is that of soft and hard constraints. Simply stated, a soft constraint is one that is penalized in the optimization algorithm but does not need to be satisfied in order to converge to an optimal solution. Hard constraints restrict the feasible space of the objective function and must be fully satisfied for the algorithm to converge. Strengths of the soft constraint include simple implementation, compatibility with ML training algorithms, robustness to noise in training data, and the ability to enforce physical knowledge that may not be fully accurate but still improves model performance. Soft constraints fit naturally within the ML field as a regularization technique for deep models. Their weakness in a traditional context is a lack of constraint guarantees and fuzziness about their full contribution to a model. Recent works have contributed some clarification and novel methodologies for dealing with the hyperparameters in these models in a more generalizable way to improve PINN convergence even with gradient pathologies present (Wang et al., 2022; Maddu et al., 2021).

A PI-ML model with hard constraints could provide formal guarantees which may be attractive in many applications, but it is much less clear how to formulate these problems without losing the advantages of ML models. The Lagrangian-dual method referenced (Fioretto et al., 2020; Fioretto et al., 2020) in Section 5.3 is one way to enforce hard constraints, if the Karush-Kuhn-Tucker (KKT) conditions can be satisfied. However, it is not guaranteed that a feasible solution will be found and the constraints are applied to discrete input points rather than a continuous domain. Using a fine input grid to approximate continuous constraint results in a very large problem that may be overly constrained to find the optimum. While some studies (Mohan et al., 2020; Fioretto et al., 2020; Zhang et al., 2018) have shown how this may work in practice, much work still needs to be done to clarify where hard constraints fit within this field.

7.5. Future outlook

In addition to big picture open questions reviewed in these perspectives, there are many topics that could use further investigation, a few of which are summarized below.

- Further investigation on the discrepancy function is needed in HSM and MC to improve interpretability and extrapolation performance. Constraining a discrepancy function in a way of following physics-based knowledge will be an interesting topic.
- Systematic and robust methods for regularization of multiple physics-based terms in PI-ML methods are needed, such that better guarantees of physics-violations are provided and less ad-hoc tuning is required.
- Incorporation of probabilistic knowledge into a parametric hybrid modeling framework is a promising topic. Non-parametric surrogates can give flexibility and uncertainty information to the parametric hybrid modeling framework, while the parametric functional form helps to constrain the exploration space that is physically valid that the non-parametric function fails to capture.
- Constraining the training of machine learning models with physics-based knowledge is a popular approach in hybrid modeling. However, limited study is present on discussion between soft constraints versus hard constraints. Systematic research on this topic will help researchers choose an appropriate hybrid model in practical applications.

8. Conclusion

This perspective piece has reviewed historical trends and recent contributions towards developing frameworks that merges physics-based and data-driven knowledge for modeling tasks, especially for processes generating dynamic data. We reviewed three different areas, Physics-Informed Machine Learning, Hybrid Submodeling and Model Calibration, attempting to highlight the major efforts, capabilities, similarities and pros/cons of each area. In reality, all areas are so diverse and rapidly growing, that it is hard to make concrete conclusions over the superiority of one approach over the other. This was never our goal, as we believe that in all areas tremendous achievements have been made, and these are all areas that deserve further attention. In Table 3, we summarize some of the key components, methods and advantages and limitations of the majority of methods belonging into each category, as defined in this review paper.

From the above summary, we make the following observations:

Table 3
Summary of three areas reviewed in this paper (PIML, HSM, MC) with respect to Knowledge Availability, Purpose/methodology and Advantages and Limitations.

	Physics-Informed Machine-Learning	Hybrid SubModeling	Model Calibration
Knowledge Availability	Differential/ Algebraic Model and (Optional) data	Incomplete Differential/ Algebraic Model and (Optional) data	Incomplete Differential/ Algebraic model (Low-fidelity data) and High-fidelity data
Purpose/ methodology	Use ML model as a basis function, optimize ML parameters to fit underlying DE model	Train DD model to estimate unknown relationships constrained by a first-principles model	Predict the true system response and parameters by introducing a discrepancy term to the low-fidelity model, using Bayesian-guided updating.
Advantage (+) Limitation (-)	(+) Compatibility with efficient ML computational libraries (+) Simple construction of inverse/forward problem structures (+) Final model is a computationally cheap surrogate (-) Poor generalization to new conditions (-) No guarantee of exact constraint satisfaction	(+) Improved prediction accuracy (over a low-fidelity model) (+) Accurate extrapolation for physics-dependent part of the model (-) Separability of data-driven and mechanistic part of model not always feasible (-) Interpretability and extrapolation limited by data-driven component	(+) Embedded uncertainty quantification (+) Ability to incorporate domain knowledge as a form of a prior (-) Multiple optimal solutions, so calibration parameters values not always physically interpretable (-) High computational cost when dataset becomes large

- While not a panacea for all modeling tasks, hybrid methods may be key to reliable, fast process characterization when either first-principles alone or data alone cannot offer robust predictive power.
- In recent years software for implementing HMs has in many ways matured enough for user-friendly implementation of HMs.
- While HMs outperform purely data-driven approaches, they are more limited in their ability to reveal additional mechanistic insight about a system than a purely first-principles approach. Modelers should take care when extrapolating that the data-driven component does not overly influence the output of the model. Notwithstanding, HM approaches that yield mechanistic insights are certainly welcome.
- Separating mechanisms into different submodels was envisioned early, but no general framework for quantifying the contributions of each SM is available. Efficient, general frameworks for quantifying the validity (i.e., uncertainty) of models with multiple SMs would help define which and how many SMs to use for a given system.
- The Bayesian component of model calibration can be a useful tool for exploring the multi-fidelity data space. Probabilistic info can be leveraged to reduce the number of experiments and qualify the validity of the calibrated model. However, less work has been done on Bayesian analysis of dynamic systems. Studies that streamline this analysis would certainly encourage the calibration framework's regular use in the chemical processes field, where dynamic, and often unexpected, changes in the system are commonplace.

CRediT authorship contribution statement

William Bradley: Writing – review & editing, Visualization. **Jinhyeun Kim:** Writing – review & editing, Methodology. **Zachary Kilwein:** Writing – review & editing, Methodology. **Logan Blakely:** Writing – review & editing. **Michael Eydenberg:** . **Jordan Jalvin:** Writing – review & editing. **Carl Laird:** Writing – review & editing. **Fani Boukouvala:** Writing – review & editing, Supervision, Funding acquisition.

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.

Funding

The authors gratefully acknowledge funding received from RAPID/ NNMI Grant #GR10002225, Georgia Tech start-up grant and NSF CBET grants (1336386 and 1944678).

Disclaimer

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Appendix

Case Study Parameters
The parameter values for base case ethylbenzene reaction and their meaning are listed below:
Catalyst Bed
$\rho = 2137[\mathrm{kg/m^3\ pellet}]$
$\phi = 0.4$
Equilibrium Constant
$K_{p1} = exp\{b_1 + \frac{b_2}{T} + b_3 \ln(T) + [(b_4 T + b_5)T + b_6]T\}[\mathrm{atm}]$

b_1	-17.34	b_4	-2.314×10^{-10}
b_2	-1.302×10^{4}	b_5	1.302×10^{-6}
b_3	5.051	b_6	-4.931×10^{-3}

Heats of Reaction

$\Delta H_{rxn1} = 118$ [kJ/mol ethylbenzene]

$\Delta H_{rxn2} = 105.2$ [kJ/mol ethylbenzene]

$\Delta H_{rxn3} = -53.9$ [kJ/mol ethylbenzene]

Heat Capacities [J/mol/K]

Methane	68	Styrene	27
Ethylene	90	Ethylbenzene	299
Benzene	201	Hydrogen	30
Toluene	249	Steam	40

References

Abadi, M., Agarwal A., Barham P., Brevdo E., Chen Z., Citro C., Corrado G.S., Davis A., Dean J., and Devin M.. 2016. 'Tensorflow: large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467*.

Abbas, M., Ilin, A., Solonen, A., Hakkarainen, J., Oja, E., Järvinen, HJ., 2016. Empirical evaluation of bayesian optimization in parametric tuning of chaotic systems. Int. J. Uncertain. Quantif. 6, 467–485.

Abonyi, J., Madar, J., Szeifert, F., Roy, R., Köppen, M., Ovaska, S., Furuhashi, T., Hoffmann, F., 2002. Combining first principles models and neural networks for generic model control. Soft Computing and Industry: Recent Applications. Springer London, London.

Agarwal, M., 1997. Combining neural and conventional paradigms for modelling, prediction and control. Int. J. Syst. Sci. 28, 65–81.

Aguiar, H.C., Maciel Filho, R., 2001. Neural network and hybrid model: a discussion about different modeling techniques to predict pulping degree with industrial data. Chem. Eng. Sci. 56, 565–570.

Arendt, P., Apley, D., Chen, W., 2012. Quantification of model uncertainty: calibration, model discrepancy, and identifiability. J. Mech. Des. 134, 100908.

Arnold, F., King, R., 2021. State–space modeling for control based on physics-informed neural networks'. Eng. Appl. Artif. Intell. 101, 104195.

Bae, J., Lee, H., Jeong, D.H., Lee, J.M., 2020. Construction of a valid domain for a hybrid model and its application to dynamic optimization with controlled exploration. Ind. Eng. Chem. Res. 59, 16380–16395.

Bangi, M.S.F., Kao, K., Kwon, JS-Il, 2022. Physics-informed neural networks for hybrid modeling of lab-scale batch fermentation for β-carotene production using Saccharomyces cerevisiae. Chem. Eng. Res. Des. 179, 415–423.

Bangi, M.S.F., Kwon, JS-Il, 2020. Deep hybrid modeling of chemical process: application to hydraulic fracturing. Comput. Chem. Eng. 134, 106696.

Bayarri, MJ., Berger, JO., Paulo, R., Sacks, J., Cafeo, JA., Cavendish, J., Lin, C.H., Tu, J., 2007. A framework for validation of computer models. Technometrics 49, 138–154.

Baydin, A.G., Pearlmutter, BA., Radul, A.A., Siskind, J.M., 2017. Automatic differentiation in machine learning: a survey. J. Mach. Learn. Res. 18, 5595–5637.

Bengio, Y., Delalleau O., and Roux N.. 2005. The curse of highly variable functions for local kernel machines.

Berg, J., and Nystrom K.. 2017. 'Neural network augmented inverse problems for PDEs', arXiv: Machine Learning.

Bhat, K.S, Mebane, DS., Mahapatra, P., Storlie, CB., 2017. Upscaling uncertainty with dynamic discrepancy for a multi-scale carbon capture system. J. Am. Stat. Assoc. 112, 1453–1467.

Bhosekar, A., Ierapetritou, M., 2018. Advances in surrogate based modeling, feasibility analysis, and optimization: a review. Comput. Chem. Eng. 108, 250–267.

Bishop, CM., 2006. Pattern Recognition and Machine Learning. Springer.

Bollas, G.M., Papadokonstadakis, S., Michalopoulos, J., Arampatzis, G., Lappas, A.A., Vasalos, I.A., Lygeros, A., 2003. Using hybrid neural networks in scaling up an FCC model from a pilot plant to an industrial unit. Chem. Eng. Process. 42, 697–713.

Braake, H.A.B., van Can, H.J.L., Verbruggen, H.B., 1998. Semi-mechanistic modeling of chemical processes with neural networks'. Eng. Appl. Artif. Intell. 11, 507–515.

Bradley, W., Boukouvala, F., 2021. Two-stage approach to parameter estimation of differential equations using neural ODEs. Ind. Eng. Chem. Res.

Butler, K.T., Davies, D.W., Cartwright, H., Isayev, O., Walsh, A., 2018. Machine learning for molecular and materials science'. Nature 559, 547–555.

Cai, G., and Mahadevan S.. 2017. 'Model Calibration with Big Data.' in.

Carmassi, M., Barbillon P., Chiodetti M., Keller M., and Parent E.. 2018. CaliCo: a R package for Bayesian calibration.

Chen, L., Hontoir, Y., Huang, D., Zhang, J., Morris, A.J, 2004. Combining first principles with black-box techniques for reaction systems. Control Eng. Pract. 12, 819–826.

Chen, L., Dai, S., Pu, Y., Zhou, E., Li, C., Su, Q., Chen, C., Carin, L., 2018a. Symmetric variational autoencoder and connections to adversarial learning. In: Proceedings of the International Conference on Artificial Intelligence and Statistics. PMLR, pp. 661–669.

Chen, RT.Q., Rubanova Y., Bettencourt J., and Duvenaud D.. 2018. "Neural ordinary differential equations." In arXiv e-prints.

Chen, W., Xiong, Y., Tsui, K.L., Wang, S., 2007. A design-driven validation approach using Bayesian prediction models. J. Mech. Des. 130.

Chen, Y., Ierapetritou, M., 2020. A framework of hybrid model development with identification of plant-model mismatch. AlChE J. 66, e16996.

Chen, Z., Wang, Bo, 2017. How priors of initial hyperparameters affect Gaussian process regression models. Neurocomputing 275.

Cozad, A., Sahinidis, N.V., Miller, D.C., 2014. Learning surrogate models for simulation-based optimization. AlChE J. 60, 2211–2227.

Czarnecki, W.M., Osindero, S., Jaderberg, M., Swirszcz, G., Pascanu, R., 2017. Sobolev training for neural networks'. Adv. Neural Inf. Process. Syst. 30.

de Azevedo, C.R, Peres, J., von Stosch, M., 2015. An efficient method for the numerical integration of measured variable dependent ordinary differential equations. Eng. Appl. Artif. Intell. 38, 24–33.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Commun. ACM 51, 107–113.

Deb, C., Zhang, F., Yang, J., Lee, S.E., Shah, K.W., 2017. A review on time series forecasting techniques for building energy consumption. Renew. Sustain. Energy Rev. 74, 902–924.

Dhillon, A., Verma, G.K., 2020. Convolutional neural network: a review of models, methodologies and applications to object detection. Prog. Artif. Intell. 9, 85–112.

Diaconis, P., 2009. The Markov chain Monte Carlo revolution. Bull. Am. Math. Soc. 46, 179 textendash205.

Dissanayake, M.W.M.G., Phan-Thien, N., 1994. Neural-network-based approximations for solving partial differential equations. Commun. Numer. Methods Eng. 10, 195–201.

Dong, G., Liu, H., 2018. Feature Engineering For Machine Learning and Data Analytics. CRC Press.

Duarte, B., Saraiva, P.M., Pantelides, C., 2004. Combined mechanistic and empirical modelling. Int. J. Chem. React. Eng.

Duvenaud, D., Nickisch H., and Rasmussen C.. 2011. 'Additive Gaussian Processes'.

Fabrizio, E., Monetti, V., 2015. Methodologies and advancements in the calibration of building energy models'. Energies 8, 2548–2574.

Fiedler, B., Schuppert, A., 2008. Local identification of scalar hybrid models with tree structure. IMA J. Appl. Math. 73, 449–476.

Fioretto, F., Mak, T.W.K., Van Hentenryck, P., 2020a. Predicting AC optimal power flows: combining deep learning and lagrangian dual methods. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 630–637.

Fioretto, F., Hentenryck P.V., Terrence W.K.M., Tran C., Baldo F., and Michele J. arXiv preprint arXiv:.09394 Lombardi. 2020. 'Lagrangian Duality for Constrained Deep Learning'.

Fogler, H.S, 1999. Elements of Chemical Reaction Engineering, 3rd ed. Prentice Hall PTR, Upper Saddle River, N.J. [1999]©1999.

Fraces, C.G., Papaioannou A., and Hamdi J. arXiv preprint arXiv:.05172 Tchelepi. 2020. 'Physics informed deep learning for transport in porous media. Buckley Leverett Problem'.

Francis-Xavier, F., Kubannek, F., Schenkendorf, R., 2021. Hybrid process models in electrochemical syntheses under deep uncertainty'. Processes 9.

Freund, R.M. 2004. 'Applied lagrange duality for constrained optimization'.

Gattiker, J., Myers K., Williams B., Higdon D., Carzolio M., and Hoegh A.. 2015. 'Gaussian process-based sensitivity analysis and Bayesian model calibration with GPMSA.' in.

Georgieva, P., Meireles, M.J., Feyo de Azevedo, S., 2003. Knowledge-based hybrid modelling of a batch crystallisation when accounting for nucleation, growth and agglomeration phenomena. Chem. Eng. Sci. 58, 3699–3713.

Gibert, K., Sànchez-Marrè, M., Izquierdo, J., 2016. A survey on pre-processing techniques: relevant issues in the context of environmental data mining. AI Commun. 29, 627–663.

Glassey, J., Stosch, M.V., 2018. Hybrid Modeling in Process Industries. CRC Press, Milton.

Goebel, R., Sanfelice, R.G., Teel, A.R., 2009. Hybrid dynamical systems. IEEE Control Syst. Mag. 29, 28–93.

Goldberg, Y.J., 2017. Synthesis lectures on human language technologies. Neural Netw. Methods Nat. Lang. Process. 10, 1–309.

Gorbach, N.S., Bian, A.A., Fischer, B., Bauer, S., Buhmann, J.M., 2017. Model selection for Gaussian process regression. Pattern Recognit. 10496, 306–318 (Gcpr 2017).

Gramacy, RB., Apley, DW., 2015. Local gaussian process approximation for large computer experiments. J. Comput. Graph. Stat. 24, 561–578.

Grossberg, S.J., 1988. Neural networks. Nonlinear Neural Netw. Princ. Mech. Archit. 1, 17–61.

GU, M. 2018. 'Robust calibration of imperfect mathematical models', R package version 0.5.0.

Gusmão, G.S., Retnanto A.P., Cunha S.C., and Medford A.J.. 2020. 'Kinetics-Informed Neural Networks', arXiv preprint arXiv:2011.14473.
Haghighat, E., and Juanes R.. 2020a. 'SciANN: a Keras wrapper for scientific computations and physics-informed deep learning using artificial neural networks', arXiv preprint arXiv:2005.08803.
Haghighat, E., and Ruben J. arXiv preprint arXiv:.08803 Juanes. 2020b. 'SciANN: a Keras wrapper for scientific computations and physics-informed deep learning using artificial neural networks'.
Hajirahimi, Z., Khashei, M., 2020. Sequence in hybridization of statistical and intelligent models in time series forecasting. Neural Process. Lett. 1–21.
Hankin, RK.S., 2005. Introducing BACCO, an R bundle for bayesian analysis of computer code output. J. Stat. Softw. 14, 1–21.
Hart, W.E., Laird, C.D., Watson, J.P., Woodruff, D.L., Hackebeil, G.A., Nicholson, B.L., Siirola, J.D., 2017. Pyomo-optimization Modeling in Python. Springer.
Higdon, D., Kennedy, M., Cavendish, JC., Cafeo, JA., Ryne, RD., 2004. Combining field data and computer simulations for calibration and prediction. SIAM J. Sci. Comput. 26, 448–466.
Hinton, GE., and Rasmussen C.. 1997. "Evaluation of gaussian processes and other methods for non-linear regression." In.
Hochreiter, S., 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain., Fuzziness Knowl. Based Syst.
Huang, C., Ren, Yi, McGuinness, E., Losego, M., Lively, R., Joseph, V., null, nEd., 2021. Bayesian optimization of functional output in inverse problems. J. Name Optim. Eng. Medium X.
Jagtap, AD., Kawaguchi, K., Karniadakis, GEm, 2020a. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks'. J. Comput. Phys. 404, 109136.
Jagtap, AD., Kharazmi, E., Karniadakis, GEm, 2020b. Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. Comput. Meth. Appl. Mech. Eng. 365, 113028.
Jia, Xiaowei, Willard J., Karpatne A., Read J.S., Zwart J.A., Steinbach M., and Vipin J. arXiv preprint arXiv:.11086 Kumar. 2020. 'Physics-guided machine learning for scientific discovery: an application in simulating lake temperature profiles'.
Jidling, C., Wahlström N., Wills A., and Schön T.. 2017. Linearly constrained Gaussian processes.
Jin, R., Chen W., and Sudjitanto A.. 2002. On sequential sampling for global metamodeling in engineering design.
Joseph, V.R., Yan, H., 2015. Engineering-driven statistical adjustment and calibration. Technometrics 57, 257–267.
Joseph, V.R, Melkote, SN., 2009. Statistical adjustments to engineering models. J. Qual. Technol. 41, 362–375.
Kadeethum, T., Jørgensen, T.M., Nick, H.M., 2020. Physics-informed neural networks for solving nonlinear diffusivity and Biot's equations. PLoS One.
Kahrs, O., Marquardt, W., 2007. The validity domain of hybrid models and its application in process optimization. Chem. Eng. Process. 46, 1054–1066.
Kars, O., Marquadt, W., 2008. Incremental identification of hybrid process models. Comput. Chem. Eng. 32, 694–705.
Kalyanaraman, J., Fan, Y., Labreche, Y., Lively, RP., Kawajiri, Y., Realff, MJ., 2015. Bayesian estimation of parametric uncertainties, quantification and reduction using optimal design of experiments for CO_2 adsorption on amine sorbents. Comput. Chem. Eng. 81, 376–388.
Kalyanaraman, J., Kawajiri, Y., Lively, RP., Realff, MJ., 2016. Uncertainty quantification via bayesian inference using sequential monte carlo methods for CO_2 adsorption process. AIChE J. 62, 3352–3368.
Karniadakis, AD.JGEm, 2020a. Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. Commun. Comput. Phys. 28, 2002–2041.
Karniadakis, GEm, Kevrekidis, IG., Lu, Lu, Perdikaris, P., Wang, S., Yang, L., 2021. Physics-informed machine learning. Nat. Rev. Phys. 3, 422–440.
Karniadakis, G.P., Em, G., 2020b. 'Physics-informed learning machines for partial differential equations: gaussian processes versus neural networks'. Nonlinear Syst. Complex. 32, 323–343.
Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., Kumar, V., 2017. Theory-guided data science: a new paradigm for scientific discovery from data. IEEE Trans. Knowl. Data Eng. 29, 2318–2331.
Kennedy, M.C., O'Hagan, A., 2001a. Bayesian calibration of computer models. J. R. Stat. Soc. Ser. B Stat. Methodol. 63, 425–450.
Kennedy, M., and O'Hagan A.. 2006. "Supplementary details on bayesian calibration of computer models." In.
Kennedy, MC., O'Hagan, A., 2001b. Bayesian calibration of computer models. J. R. Stat. Soc. Ser. B 63, 425–464 (Statistical Methodology).
edited by Keskitalo, J., Leiviskä, K., Angelov, P., Atanassov, K.T., Doukovska, L., Hadjiski, M., Jotsov, V., Kacprzyk, J., et al., 2015. Artificial neural network ensembles in hybrid modelling of activated sludge plant. Intelligent Systems'2014. Springer International Publishing, Cham, pp. 683–694. edited by.
Kevrekidis, P.G., Cuevas-Maraver, J., Saxena, A., 2020. Emerging Frontiers in Nonlinear Science. Springer.
Kim, J., Lee K., Lee D., Sheo Y.J., and Noseong J arXiv preprint arXiv:.02681 Park. 2020. 'DPM: a novel training method for physics-informed neural networks in extrapolation'.
Kim, Y.J., Park, C.S., 2016. Stepwise deterministic and stochastic calibration of an energy simulation model for an existing building. Energy Build. 133, 455–468.
Kim, Y., Choi Y., Widemann D., and Zohdi T.. 2020. 'A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder', arXiv preprint arXiv:2009.11990.
Kissas, G., Yang, Y., Hwuang, E., Witschey, W.R., Detre, J.A., Perdikaris, P., 2020. Machine learning in cardiovascular flows modeling: predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks. Comput. Methods Appl. Mech. Eng.
Krogh, A., 2008. What are artificial neural networks? J Nat. Biotechnol. 26, 195–197.
L'Heureux, A., Grolinger, K., Elyamany, H.F., Capretz, M.A.M., 2017. Machine learning with big data: challenges and approaches. IEEE Access 5, 7776–7797.
Lagaris, I.E., Likas, A., Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. IEEE Trans. Neural Netw. 9, 987–1000.
Lange-Hegermann, M. 2020. 'Linearly constrained Gaussian processes with boundary conditions', *ArXiv*, abs/2002.00818.
Lee, D.S., Jeon, C.O., Park, J.M., Chang, K.S., 2002. Hybrid neural network modeling of a full-scale industrial wastewater treatment process. Biotechnol. Bioeng. 78, 670–682.
Lee, D.S., Vanrolleghem, PA., Park, J.M., 2005. Parallel hybrid modeling methods for a full-scale cokes wastewater treatment plant. J. Biotechnol. 115, 317–328.
Lee, D., Jayaraman, A., Kwon, JS., 2020. Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling. PLoS Comput. Biol. 16, e1008472.
Lee, G., Kim, W., Oh, H., Youn, B.D., Kim, N., 2019a. Review of statistical model calibration and validation—from the perspective of uncertainty structures. Struct. Multidiscip. Optim. 60.
Lee, JH., Shin, J., Realff, MJ., 2018. Machine learning: overview of the recent progresses and implications for the process systems engineering field. Comput. Chem. Eng. 114, 111–121.
Lee, K., Cho, H., Lee, I., 2019b. Variable selection using Gaussian process regression-based metrics for high-dimensional model approximation with limited data. Struct. Multidiscip. Optim. 59, 1439–1454.
Lee, S., Dietrich, F., Karniadakis, G.E., Kevrekidis, I.G., 2019c. Linking Gaussian process regression with data-driven manifold embeddings for nonlinear data fusion. Interface Focus 9, 20180083.
Li, K., Mahapatra, P., Sham Bhat, K., Miller, DC., Mebane, DS., 2017. Multi-scale modeling of an amine sorbent fluidized bed adsorber with dynamic discrepancy reduced modeling. React. Chem. Eng. 2, 550–560.
Lin, Li-H, Joseph, V., 2019. Transformation and additivity in Gaussian processes. Technometrics 62, 1–27.
Ling, Y., Mullins, J., Mahadevan, S., 2014. Selection of model discrepancy priors in Bayesian calibration. J. Comput. Phys. 276, 665–680.
Linkletter, C., Bingham, D., Hengartner, N., Higdon, D., Ye, KQ., 2006. Variable selection for Gaussian process models in computer experiments. Technometrics 48, 478–490.
Liu, F., Bayarri, M., Berger, J., 2009. Modularization in Bayesian analysis, with emphasis on analysis of computer models. Bayesian Anal. 4.
Liu, H., Ong Y., Shen X., and Cai J.. 2018. When Gaussian process meets big data: a review of scalable GPs.
Lopez, P.C., Udugama, IA., Thomsen, ST., Roslander, C., Junicke, H., Mauricio-Iglesias, M., Gernaey, KV., 2020. Towards a digital twin: a hybrid data-driven and mechanistic digital shadow to forecast the evolution of lignocellulosic fermentation. Biofuels Bioprod. Biorefin. 14, 1046–1060.
Lu, J., Yao Ke, and Gao F.. 2009. 'Process similarity and developing new process models through migration', 55: 2318–28.
Lunderman, S., Morzfeld, M., Posselt, DJ., 2021. Using global Bayesian optimization in ensemble data assimilation: parameter estimation, tuning localization and inflation, or all of the above. Tellus A Dyn. Meteorol. Oceanogr. 73, 1–16.
Luo, L., Gao, F., 2015. Model migration through bayesian adjustments. IFAC-PapersOnLine 48, 112–116.
Lutter, M., Ritter C., and Peters J.. 2019. 'Deep lagrangian networks: using physics as model prior for deep learning', *arXiv preprint arXiv:1907.04490*.
MacKay, DJ.C., 2003. Information theory, inference, and Learning Algorithms. Cambridge University Press: Cambridge, UK; New York.
Maddu, S., Sturm, D., Müller, C.L., Sbalzarini, I.F., 2021. Inverse dirichlet weighting enables reliable training of physics informed neural networks. Mach. Learn. Sci. Technol.
Manfren, M., Aste, N., Moshksar, R., 2013. Calibration and uncertainty analysis for computer models – a meta-model based approach for integrated building energy simulation'. Appl. Energy 103, 627–641.
Matsunawa, T., Yu, B., Pan, D., 2015. *Optical* Proximity Correction With Hierarchical Bayes model. SPIE.
McBride, K., Medina, E.I.S., Sundmacher, K., 2020. Hybrid semi-parametric modeling in separation processes: a review. Chem. Ing. Tech. 92, 842–855.
McBride, K., Sundmacher, K., 2019. Overview of surrogate modeling in chemical process engineering. Chem. Ing. Tech. 91, 228–239.
McCann, M.T., Hwan Jin, K., Michael, J., 2017. Convolutional neural networks for inverse problems in imaging: a review. IEEE Signal Process. Mag. 34, 85–95. Unser.
McIntire, M., Ratner, D., Ermon, S., 2016. Sparse Gaussian processes for Bayesian optimization. In: Proceedings of the UAI.
Meng, X., Karniadakis, GEm, 2020. A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems. J. Comput. Phys. 401, 109020.
Meng, Y., Yu, S., Zhang, J., Qin, J., Dong, Z., Lu, G., Pang, H., 2019. Hybrid modeling based on mechanistic and data-driven approaches for cane sugar crystallization. J. Food Eng. 257, 44–55.
Misyris, G.S., Venzke A., and Spyros J. arXiv preprint arXiv:.03737 Chatzivasileiadis. 2019. 'Physics-Informed Neural Networks for Power Systems'.
Mitusch, SK., Funke S., and Kuchta M.. 2021. 'Hybrid FEM-NN models: combining artificial neural networks with the finite element method', ArXiv, abs/2101.00962.

Mohan, A.T., Lubbers N., Livescu D., and Chertkov M.. 2020. 'Embedding hard physical constraints in neural network coarse-graining of 3D turbulence', arXiv preprint arXiv:2002.00021.

Mowlavi, S., and Nabi S.. 2021. 'Optimal control of PDEs using physics-informed neural networks', *arXiv preprint arXiv:2111.09880.*.

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. 0.

Oliveira, R., 2004. Combining first principles modelling and artificial neural networks: a general framework. Comput. Chem. Eng. 28, 755–766.

Olofsson, S., Peter Deisenroth, M., Misener, R., 2018. Design of experiments for model discrimination using Gaussian process surrogate models. Computer Aided Chemical Engineering. Elsevierlocation|child::sb:host[1]/sb:edited-book/sb:publisher/sb:location)"_$^!"boolean(child::sb:host/sb:pages)">.

Pakravan, Samira, Mistani, Pouria A, Aragon-Calvo, Miguel Angel, Gibou, Frederic, 2020. Solving inverse-PDE problems with physics-aware neural networks. arXiv preprint. *arXiv:2001.03608*.

Palomo, J., Paulo, R., Garcia-Donato, G., 2015. SAVE: an R package for the statistical analysis of computer models. J. Stat. Softw. 64.

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Trans. Knowl. Data Eng. 22, 1345–1359.

Pang, G., Lu, Lu, Karniadakis, GEm, 2019. fpinns: fractional physics-informed neural networks'. SIAM J. Sci. Comput. 41, A2603–A2A26.

Paszke, A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., and Antiga L.. 2019. 'Pytorch: an imperative style, high-performance deep learning library', *arXiv preprint arXiv:1912.01703*.

Peherstorfer, B., Willcox, K., Gunzburger, M., 2018. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. SIAM Rev. 60, 550–591.

Perdikaris, P., Venturi, D., Karniadakis, GEm, 2016. Multifidelity information fusion algorithms for high-dimensional systems and massive data sets. SIAM J. Sci. Comput. 38, B521–BB38.

Perdikaris, P., Venturi, D., Royset, J.O., Karniadakis, GEm, 2015. Multi-fidelity modelling via recursive co-kriging and Gaussian–Markov random fields'. In: Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, 471, 20150018.

Piironen, J., Vehtari, A., 2016. Projection predictive model selection for Gaussian processes. In: Proceedings of the IEEE 26th International Workshop on Machine Learning for Signal Processing. MLSP, pp. 1–6.

Pinto, J., Azevedo, C.R., Oliveira, R., Stosch, M., 2019. A bootstrap-aggregated hybrid semi-parametric modeling framework for bioprocess development. Bioprocess. Biosyst. Eng. 42, 1853–1865.

Plumlee, M., 2017. Bayesian calibration of inexact computer models. J. Am. Stat. Assoc. 112, 1274–1285.

Potharst, R., and Feelders A.J.. 2002. 'Classification trees for problems with monotonicity constraints', 4: 1–10.

Psichogios, DC., Ungar, LH., 1992. A hybrid neural network-first principles approach to process modeling. AlChE J. 38, 1499–1511.

Qi Zhi, He, Barajas-Solano, D., Tartakovsky, G., Tartakovksy, AM., 2020. Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. Adv. Water Resour. 141.

Qin, S.J, Chiang, LH., 2019. Advances and opportunities in machine learning for process data analytics. Comput. Chem. Eng. 126, 465–473.

Quaghebeur, W., Nopens, I., De Baets, B., 2021. Incorporating unmodeled dynamics into first-principles models through machine learning. IEEE Access 9, 22014–22022.

Quaghebeur, W., Torfs, E., Baets, BDe, Nopens, I., 2022. Hybrid differential equations: integrating mechanistic and data-driven techniques for modelling of water systems. Water Res. 213, 118166.

Rackauckas, C., Ma Y., Dixit V., Guo X., Innes M., Revels J., Nyberg J., and Ivaturi V.D.. 2018. 'A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions', ArXiv, abs/1812.01892.

Rackauckas, C., Ma Y., Martensen J., Warner C., Zubov K., Supekar R., Skinner D., and Ramadhan A.. 2020. "Universal differential equations for scientific machine learning." In arXiv e-prints, arXiv:2001.04385.

Rackauckas, C., Singhvi A., Ma Y., Hatherly M., Jones S.P., Caine C., Saba E., TagBot J., and Olver S.. 2020. 'SciML/differentialequations. jl: v6. 15.0'.

Raissi, M. 2019. 'Physics informed neural networks (github Repository)'.

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.

Raissi, M., Perdikaris, P., Karniadakis, GEm, 2017a. Inferring solutions of differential equations using noisy multi-fidelity data. J. Comput. Phys. 335, 736–746.

Raissi, M., Karniadakis, G., 2017b. Machine learning of linear differential equations using Gaussian processes. J. Comput. Phys. 348, 683–693.

Rasmussen, C.E., Williams, C.K.I., 2005. Gaussian processes for machine learning. Gaussian Processes for Machine Learning, pp. 1–247.

Rico-Martínez, R., Krischer, K., Kevrekidis, I.G., Kube, M.C., Hudson, J.L., 1992. Discrete-vs. continuous-time nonlinear signal processing of Cu electrodissolution data. Chem. Eng. Commun. 118, 25–48.

Rüden, 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., Schücker, J., 2019. Informed machine learning – a taxonomy and survey of integrating knowledge into learning systems. arXiv Mach. Learn.

Salvatier, J., Wiecki, TV., Fonnesbeck, C., 2016. Probabilistic programming in python using PyMC3. PeerJ Comput. Sci. 2, e55.

Sansana, J., Joswiak, MN., Castillo, I., Wang, Z., Rendall, R., Chiang, LH., Reis, MS., 2021. Recent trends on hybrid modeling for Industry 4.0′. Comput. Chem. Eng. 151, 107365.

Santner, TJ., Williams, BJ., Notz, W., 2003. *The* Design and Analysis of Computer Experiments. Springer, New York.

Sargsyan, K., Najm, H.N., Ghanem, R., 2015. On the statistical calibration of physical models. Int. J. Chem. Kinet. 47, 246–276.

Särkkä, S. 2011. "Linear operators and stochastic partial differential equations in gaussian process regression." In, 151–58. Berlin, Heidelberg: Springer Berlin Heidelberg.

Savitsky, T., Vannucci, M., Sha, N., 2011. Variable selection for nonparametric gaussian process priors: models and computational strategies'. Stat. Sci. A Rev. J. Inst. Math. Stat. 26, 130–149.

Schäfer, P., Caspari, A., Mhamdi, A., Mitsos, A., 2019. Economic nonlinear model predictive control using hybrid mechanistic data-driven models for optimal operation in real-time electricity markets: in-silico application to air separation processes. J. Process Control 84, 171–181.

Schäfer, P., Caspari, A., Schweidtmann, AM., Vaupel, Y., Mhamdi, A., Mitsos, A., 2020. The potential of hybrid mechanistic/data-driven approaches for reduced dynamic modeling: application to distillation columns. Chem. Ing. Tech. 92, 1910–1920.

Schubert, J., Simutis, R., Dors, M., Havlik, I., Lübbert, A., 1994. Bioprocess optimization and control: application of hybrid modelling. J. Biotechnol. 35, 51–68.

Schulz, E., Speekenbrink, M., Krause, A., 2018. A tutorial on Gaussian process regression: modelling, exploring, and exploiting functions. J. Math. Psychol. 85, 1–16.

Schweidtmann, AM., Weber, JM., Wende, C., Netze, L., Mitsos, A., 2021. Obey validity limits of data-driven models through topological data analysis and one-class classification. Optim. Eng.

Simutis, R., Havlik, I., Schneider, F., Dors, M., Lübbert, A., Munack, A., Schügerl, K., 1995. Artificial neural networks of improved reliability for industrial process supervision. Computer Applications in Biotechnology. Pergamon, Amsterdam.

Sirignano, J., Spiliopoulos, K., 2018. DGM: a deep learning algorithm for solving partial differential equations. J. Comput. Phys. 375, 1339–1364.

Snelson, E., and Ghahramani Z.. 2005. Sparse Gaussian Processes using Pseudo-inputs.

Snyder, JD., Subramaniam, B., 1994. A novel reverse flow strategy for ethylbenzene dehydrogenation in a packed-bed reactor. Chem. Eng. Sci. 49, 5585–5601.

Stein, A., Corsten, L.C.A., 1991. Universal kriging and cokriging as a regression procedure. Biometrics 575–587.

Su, H-Te, Bhat, N., Minderman, P.A., McAvoy, T.J., 1992. Integrating neural networks with first principles models for dynamic modeling. IFAC Proc. Vol. 25, 327–332.

Sun, B., Yang, C., Wang, Y., Gui, W., Craig, I., Olivier, L., 2020. A comprehensive hybrid first principles/machine learning modeling framework for complex industrial processes. J. Process Control 86, 30–43.

Tagade, P., Hariharan, KS., Basu, S., Verma, M.K.S., Kolake, S.M., Song, T., Oh, D., Yeo, T., Doo, S., 2016. Bayesian calibration for electrochemical thermal model of lithium-ion cells. J. Power Sources 320, 296–309.

Tagade, PM., Jeong, B.M., Choi, H.L., 2013. A Gaussian process emulator approach for rapid contaminant characterization with an integrated multizone-CFD model. Build. Environ. 70, 232–244.

Tascikaraoglu, A., Uzunoglu, M., 2014. A review of combined approaches for prediction of short-term wind speed and power. Renew. Sustain. Energy Rev. 34, 243–254.

Teixeira, A., Cunha, A.E., Clemente, J.J., Moreira, J.L., Cruz, H.J., Alves, P.M., Carrondo, M.J.T., Oliveira, R., 2005. Modelling and optimization of a recombinant BHK-21 cultivation process using hybrid grey-box systems. J. Biotechnol. 118, 290–303.

Thompson, ML., Kramer, MA., 1994. Modeling chemical processes using prior knowledge and neural networks'. AlChE J. 40, 1328–1340.

Tipireddy, R., and Tartakovsky A.. 2018. Physics-informed machine learning method for forecasting and uncertainty quantification of partially observed and unobserved states in power grids.

Tsay, C., 2021. Sobolev trained neural network surrogate models for optimization. Comput. Chem. Eng. 153, 107419.

van Can, H.J.L., te Braake, H.A.B., Bijman, A., Hellinga, C., Ch, K., Luyben, A.M., Heijnen, J.J., 1999. An efficient model development strategy for bioprocesses based on neural networks in macroscopic balances: part II. Biotechnol. Bioeng. 62, 666–680.

van Can, H.J.L., te Braake, H.A.B., Hellinga, C., Luyben, K.C.A.M., 1997. An efficient model development strategy for bioprocesses based on neural networks in macroscopic balances. Biotechnol. Bioeng. 54, 549–566.

Van Can, HJ.L., Hellinga, C., Luyben, KCh.A.M., Heijnen, JJ., Braake, HA.B.Te, 1996. Strategy for dynamic process modeling based on neural networks in macroscopic balances. AlChE J. 42, 3403–3418.

Van Can, HJ.L., Braake, HA.B.Te, Dubbelman, S., Hellinga, C., Luyben, KCh.A.M., Heijnen, JJ., 1998. Understanding and applying the extrapolation properties of serial gray-box models. AlChE J. 44, 1071–1089.

Venkatasubramanian, V., 2019. The promise of artificial intelligence in chemical engineering: is it here, finally? AlChE J. 65, 466–478.

Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A., 2008. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103.

von Stosch, M., Davy, S., Francois, K., Galvanauskas, V., Hamelink, J.M., Luebbert, A., Mayer, M., Oliveira, R., O'Kennedy, R., Rice, P., Glassey, J., 2014a. Hybrid modeling for quality by design and PAT-benefits and challenges of applications in biopharmaceutical industry. Biotechnol. J. 9, 719–726.

von Stosch, M., Oliveira, R., Peres, J., Azevedo, S.F., 2012. A general hybrid semi-parametric process control framework. J. Process Control 22, 1171–1181.

von Stosch, M., Oliveira, R., Peres, J., Azevedo, S.F., 2014b. Hybrid semi-parametric modeling in process systems engineering: past, present and future. Comput. Chem. Eng. 60, 86–101.
Wahlström, N., Kok, M., Schön, T.B., Gustafsson, F., 2013. Modeling magnetic fields using Gaussian processes. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3522–3526.
Wang, R., Walters R., and Rose J. arXiv preprint arXiv:.03061 Yu. 2020. 'Incorporating symmetry into deep dynamics models for improved generalization'.
Wang, S., Teng Y., and Perdikaris P.. 2020. 'Understanding and mitigating gradient pathologies in physics-informed neural networks', ArXiv, abs/2001.04536.
Wang, S., Yu, X., Perdikaris, P., 2022. When and why PINNs fail to train: a neural tangent kernel perspective. J. Comput. Phys. 449, 110768.
Wang, X., Chen, J., Liu, C., Pan, F., 2010. Hybrid modeling of penicillin fermentation process based on least square support vector machine. Chem. Eng. Res. Des. 88, 415–420.
Wang, Y., Yue X., Tuo R., Hunt J., and Shi J.. 2019. Effective model calibration via sensible variable identification and adjustment, with application to composite fuselage simulation.
Willard, J., Jia X., Xu S., Steinbach M., and Kumar V.. 2020. 'Integrating physics-based modeling with machine learning: a survey', ArXiv, abs/2003.04919.
Willis, MJ., Stosch, M., 2017. Simultaneous parameter identification and discrimination of the nonparametric structure of hybrid semi-parametric models. Comput. Chem. Eng. 104, 366–376.
Wilson, A., Gilboa E., Nehorai A., and Cunningham J.. 2013. 'GPatt: fast multidimensional pattern extrapolation with Gaussian processes', ArXiv, abs/1310.5288.
Wilson, A., and Adams R.. 2013. 'Gaussian process covariance kernels for pattern discovery and extrapolation', Proceedings of the 30th International Conference on Machine Learning, ICML 2013.
Wilson, ZT., Sahinidis, NV., 2017. The ALAMO approach to machine learning. Comput. Chem. Eng. 106, 785–795.
Wipf, D., and Nagarajan S.. 2007. A new view of automatic relevance determination.
Wu, T., Movellan, J., 2012. Semi-parametric Gaussian process for robot system identification. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 725–731.
Wu, Z., Rincon, D., Christofides, P.D., 2020. Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. J. Process Control 89, 74–84.
Xia, Y., Feng, G., Wang, J., 2008. A novel recurrent neural network for solving nonlinear optimization problems with inequality constraints. IEEE Trans. Neural Netw. 19, 1340–1353.
Xiong, Y., Chen, W., Tsui, K.L., Apley, DW., 2009. A better understanding of model updating strategies in validating engineering models. Comput. Methods Appl. Mech. Eng. 198, 1327–1337.
Yan, F., Qi, Y., 2010. Sparse Gaussian process regression via L1 penalization. In: Proceedings of the ICML.
Yang, A., Martin, E., Morris, J., 2011. Identification of semi-parametric hybrid process models. Comput. Chem. Eng. 35, 63–70.
Yang, S., Wong, SW.K., Kou, S.C., 2021. Inference of dynamic systems from noisy and sparse data via manifold-constrained Gaussian processes. Proc. Natl. Acad. Sci. 118, e2020397118.
Yang, S., Navarathna, P., Ghosh, S., Wayne Bequette, B., 2020. Hybrid modeling in the era of smart manufacturing. Comput. Chem. Eng. 140, 106874.
Yang, X., Tartakovsky G., and Tartakovsky A.. 2018. Physics-informed kriging: a physics-informed gaussian process regression method for data-model convergence.
Yang, Y., and Perdikaris P.. 2018. 'Physics-informed deep generative models', arXiv preprint arXiv:1812.03511.
Yi, G., Shi, J.Q., Choi, T., 2011. Penalized Gaussian process regression and classification for high-dimensional nonlinear data. Biometrics 67, 1285–1294.
Zendehboudi, S., Rezaei, N., Lohi, A., 2018. Applications of hybrid models in chemical, petroleum, and energy systems: a systematic review. Appl. Energy 228, 2539–2566.
Zhang, Li, Wang, F., Sun, T., Xu, B., 2018. A constrained optimization method based on BP neural network. Neural Comput. Appl. 29, 413–421.