Enhancing generalization and training efficiency of neural ordinary differential equations for chemical reactor modeling

Citation for the original published paper (version of record):
Hajireza, K., Eklund, D., Andersson, R. (2025). Enhancing generalization and training efficiency of neural ordinary differential equations for
chemical reactor modeling. Chemical Engineering Journal, 519.
http://dx.doi.org/10.1016/j.cej.2025.165183

Chemical Engineering Journal 519 (2025) 165183

Contents lists available at ScienceDirect

Chemical Engineering Journal

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

Enhancing generalization and training efficiency of neural ordinary differential equations for chemical reactor modeling

Kian Hajireza [a,b], David Eklund [b,*], Ronnie Andersson [a]

[a] Department of Chemistry and Chemical Engineering, Chalmers University of Technology, Gothenburg, SE-41296, Sweden
[b] Applied Digitalization, RISE Research Institutes of Sweden, Stockholm, SE-16440, Sweden

ARTICLE INFO

Keywords:
Neural ODE
Conservation laws
Physics-regularization
Pre-training
Design of experiments

ABSTRACT

Digitalization in chemical engineering enhances data accessibility and allows leveraging advancements in machine learning to develop predictive reactor models. Nevertheless, purely data-driven machine learning models lack fundamental physical and chemical principles, thereby limiting their interpretability and generalization capabilities. Adhering to the principles of scientific machine learning, the integration of fundamental constraints can overcome these limitations. In this study, we propose incorporating conservation laws as a soft constraint in the reactor model. These laws, which are not known *a priori*, are automatically discovered from data and integrated with neural ODEs. By assessing the null space of the dependent variables within the datasets, these laws are identified, thereby enhancing model generalization. The findings show that embedding physics-regularization not only improves generalization, especially when data is scarce, but also enhances robustness. Furthermore, we demonstrate that when sampling fails to capture key dynamics, the model can still accurately predict the evolution of individual species concentrations. This finding suggests that information loss resulting from inadequate sampling can be effectively mitigated by using the proposed method. By introducing noise into the training and validation datasets, we show that the methodology remains robust and consistently outperforms benchmarks across all different levels of noise studied. In data-rich scenarios where generalization is less of a concern, pre-training the model reduces the total computational time by 68 percent. Thus, the implementation of suitable strategies for reactor modeling can significantly improve accuracy, robustness, and computational efficiency.

1. Introduction

Mechanistic approaches have proven to be useful and successful in the context of modeling chemical reactors [1–3]. Nonetheless, a limitation of mechanistic approaches lies in the necessity that they require full mathematical description of the physico-chemical phenomena eg. kinetics, hydrodynamics, dispersion, mass transport, heat transfer, thermodynamics, etc. In certain scenarios, chemical reactions that take place are highly complex and non-elementary with several non-idealities such as axial dispersion, dead zones, catalyst deactivation or mass transfer limitations. Consequently, formulating a precise representation of the chemical reactor exclusively through mechanistic modeling proves to be inherently challenging and, in certain instances, virtually infeasible. Additionally, limitations such as lack of flexibility and expensive computations have led to the consideration of alternative methods [4]. An alternative to mechanistic modeling is using a data-driven approach. Data-driven methods, such as nonlinear regression [5] have limitations as they rely on assumptions about the structure of the rate law. This further motivates the use of alternative methods if such information is not available *a priori*. One such methodology is to use methods from deep learning [6] by utilizing neural networks, which are known to be universal approximators [7–10]. More recently, neural networks have been applied to scientific problems which has introduced the subfield of scientific machine learning [11], which seeks to use machine learning methodologies in conjunction with scientific data [12–14].

Recent publications have proposed novel approaches to scientific machine learning such as physics-informed neural networks (PINNs) [15,16], universal differential equations (UDEs) [17] and neural ordinary differential equations (neural ODEs) [18]. PINNs utilize both observed data and the governing equations, embedding both of them into the loss function. Forward PINNs are useful for problems and domains where the governing equations are known in their entirety, and therefore the applicability of forward PINNs are limited to problems where structural discovery is important. A UDE is characterized by

* Corresponding author.
E-mail address: david.eklund@ri.se (D. Eklund).

https://doi.org/10.1016/j.cej.2025.165183

Available online 4 July 2025

certain terms of the governing equation that are unknown, where the unknown terms can be approximated with a surrogate model such as a neural network, thus, integrating the governing equation with a neural network. A neural ODE approximates the entire system dynamics, i.e. right-hand side of the governing equation, as shown in Eq. (1). Here the right-hand side is approximated by a neural network, $\mathcal{NN}$, which is a function of its network parameters θ and the state variables y, where i is the index for a given state variable

$$\frac{dy_i}{dt} = \mathcal{NN}_i(\theta, y(t)). \tag{1}$$

Neural ODEs are capable of capturing continuous dynamics from solely ground truth temporal data and are therefore suitable for modeling continuous-time dynamics with observed data that have arbitrary measurements with respect to time. Thus, it can thoroughly recreate certain features such as curvature. Neural ODEs have been successfully applied to various chemical systems. For instance, Kim et al. [19] explored their use in modeling stiff chemical reaction systems, including Robertson's problem and a complex air pollution scenario in atmospheric chemistry. Additionally, Yin et al. [20] developed a generalized reactor model utilizing neural ODEs to predict chemical kinetics and residence-time distributions, thereby extending their application to non-ideal reactors. In another study, Fedorov et al. [21] demonstrated that incorporating mechanistic information, such as thermodynamic and kinetic data, into a neural ODE reduces the need for extensive observed data, highlighting the efficiency of integrating domain-specific knowledge. Neural ODEs have also been applied in mechanistic parameter estimation [22], optimization of kinetic parameters [23], industrial chemical applications [24], learning kinetic models with embedded thermodynamic and stoichiometric knowledge [25] and lastly, discovery of reaction pathways using a chemical reaction neural network (CRNN) [26].

1.1. Data-poor regime

Conducting experiments in chemistry can be costly due to various factors such as expensive catalysts, the use of scarcely available reactants, energy consumption, and the need for advanced instrumentation. This scenario motivates decreasing the number of experiments conducted by structurally employing statistical experimental design to gain maximum meaningful insights into the system's behavior. Moreover, if this is coupled with poor sampling, which could lead to scarce data, then using solely a data-driven approach for this problem could lead to difficulties obtaining a model with strong generalization capabilities. This encourages integrating mechanistic knowledge into the model such as a hybrid model [27,28]. By combining the two modeling methodologies, it is possible to leverage the strong predictive capabilities of neural networks and inducing prior structures from mechanistic understanding to constrain the parameter space of the neural network. This can be achieved by incorporating conservation laws into the model, for example by embedding the conservation laws into the loss function. Nevertheless, if the conservation laws are not explicitly known, then the conservation laws must be discovered. Related works [29] have used sparse identification of non-linear dynamics (SINDy) in conjunction with Koopman eigenfunctions to discover non-linear conservation laws. To achieve this, the authors had set up a library of candidate basis functions and computed the null space of the directional derivatives using singular value decomposition (SVD) to obtain the conservation laws. While a chemical reactor is characterized by a system of non-linear differential equations, the solution of dependent variables allows the identification of how conservation of mass persists in each instance of time. Identification of the null space using SVD allows the separation of noise from the underlying conservation to enforce this during training of the neural network in the neural ODE. The discovery of mass balances using SVD has been studied, however only in the context of determining phase proportions in minerals [30]. Döppel and Votsmeier [31] implemented a CRNN and used a molecular matrix to extract atom balance laws. Nevertheless, extracting conservation laws from the molecular matrix requires *a priori* information regarding the chemical formula of most of the species involved in the reactions. Limitations such as signal complexity, complex and large compounds or poor ionization efficiency may lead to difficulties having access to a molecular matrix which encourages the consideration of alternative methods for extracting information regarding the conservation laws. This work aims to discover formulations for the conservation of mass in the context of temporal data, which is omnipresent in experiments with batch reactors. It is reasonable to assume that the discovered formulation for the conservation of mass can be used as a form of constraint, which in theory should decrease the reliance on significant amounts of training data. Related works [32–34] have applied a hard and soft constraint, however the authors assumed prior knowledge such as chemical formulas of the individual species. Other related works [35] have used both a hard and soft constraint with *a priori* knowledge of the conservation laws that were applied. Using mechanistic information as soft constraints resembles a PINN, however our proposed method aims to discover the employed constraints unlike a forward PINN, where such knowledge is commonly known *a priori*.

In this work, we propose a strategy to efficiently train neural ODEs using scarcely available data. We discover formulations for the conservation of mass for a given chemical system using a limited amount of data and without making any prior assumptions regarding the structure of the system dynamics or about the chemical formula of the individual species. The discovered formulations for the conservation of mass are thereafter used as soft constraints in the form of physics-regularization for training the neural ODE. To the best of our knowledge, using discovered formulations for the conservation of mass without any *a priori* information as a form of soft regularization has not been previously studied. Lastly, our proposed method will also be evaluated with deteriorating quality of training data by introducing Gaussian noise.

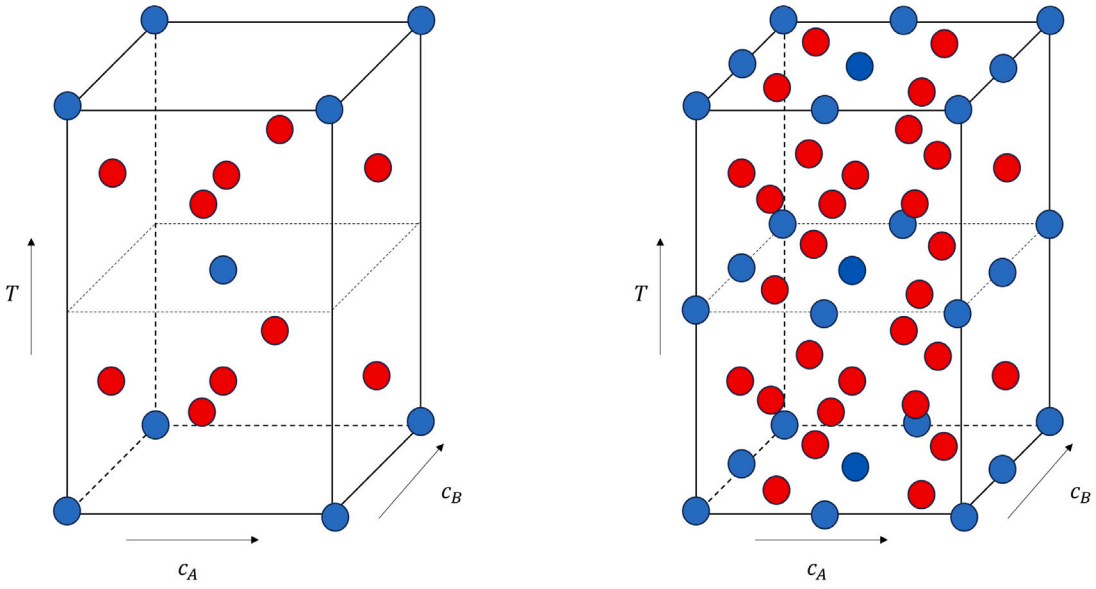
1.2. Data-rich regime

Backpropagation is employed to iteratively update the neural network parameters during training. The bottleneck of this approach is that backpropagation will be a function of the ODE solver. Namely, Eq. (2) has to be solved for each iteration and backpropagation is performed through the ODE solver:

$$\begin{aligned} y\left(t_1\right) &= y\left(t_0\right) + \int_{t_0}^{t_1} \mathcal{NN}(\theta, y(t))dt \\ &= \text{ODESolve}\left(y(t_0), \mathcal{NN}, t_0, t_1, \theta\right). \end{aligned} \tag{2}$$

For complex systems with many variables and initial conditions, this can be computationally cumbersome. An alternative would be train the universal approximator with the input and output combination of $\left\langle y_i, \frac{dy_i}{dt} \right\rangle$. This method was implemented by Roesch et al. [36], where collocation-based training was employed. In this approach, the neural network is trained utilizing the numerical derivatives of the dependent variables. This method thus circumvents the need for an ODE solver. In Goyal and Benner [37], a variant of collocation-based training was used for improved robustness. Furthermore collocation-based training has been demonstrated as a pre-training approach [38]. The study also investigated the theoretical time complexity when using pre-training. The works mentioned regarding collocation-based training have all used a single set of initial conditions for training.

In the current work, we employ pre-training together with a set of initial conditions using a full factorial design as training data. Additionally, previous studies regarding collocation-based pre-training have used number of epochs as the termination criterion and have therefore not reported the effect of pre-training on computational time. This work aims to convey the effect of pre-training on computational time.



(a) Data-poor regime (b) Data-rich regime

Fig. 1. Experimental design for 1(a) the data-poor regime and 1(b) the data-rich regime. The blue scatter points represent training datasets and the red scatter points represent validation datasets. The data-poor regime utilizes a 2^3 full factorial design with a center point to synthesize training datasets and the data-rich regime employs a 3^3 full factorial design to synthesize training datasets.

2. Materials and methods

2.1. Design of experiments

In machine learning applications, data is often split by random partitioning, cross-validation or bootstrapping. However, to achieve optimal predictive capability within the specific operating conditions defined in Fig. 1, a Design of Experiments (DoE) procedure is followed to structurally create a set of initial conditions that the neural network is trained on. A full factorial design is used to construct comprehensive sets of initial conditions for the state variables in order to capture any possible interactions between the factors, where a 2^3 full factorial design with a center point is used for the data-poor regime and a 3^3 full factorial design is used for the data-rich regime. Fig. 1 illustrates the experimental set-up used for this work. The set of factors that are varied include the concentration for species A and B and lastly, the temperature, T. All factors for the data-poor regime are varied on two levels, whereas the factors are varied on three levels for the data-rich regime as Table 1 shows. The initial conditions for the concentration of species C, D and E are all zero since species D and E are end-products and C is an intermediate product. To evaluate the performance of the model, an independent test set is constructed using a Monte-Carlo simulation to generate 1000 randomly sampled initial conditions within the upper and lower bounds of the initial values presented in Table 1. Each test dataset is sampled at 1000 evenly spaced points in time.

2.2. Neural ODEs and pre-training

As previously mentioned, to train neural ODEs, Eq. (2) needs to be solved for each epoch. To update the network parameters, a loss function is minimized shown in Eq. (3) through backpropagation

$$\mathcal{L}_{\text{data}}(\theta) = \sum_{j=1}^{N}\sum_{i=1}^{S}\sum_{k=1}^{\tau_{\text{obs}}} \frac{1}{NS\tau_{\text{obs}}}\left(y_{\text{obs},j,i,k} - \text{ODESolve}_i\left(y_{\text{obs},j,1:S,0}, \mathcal{NN}, t_0, t_k, \theta\right)\right)^2, \tag{3}$$

where N is the number of initial conditions, S is the number of state variables, τ_{obs} is the number of observed data points in time and y_{obs} is the observed ground truth data. A key point is to acknowledge that $y_{\text{obs},j,i,k}$ is a tensor, where j represents the jth initial condition, i represents the ith species and k represents the kth temporal data point. Performing backpropagation without the use of the adjoint sensitivity method [39] to compute the gradients of the loss function leads to a very complex computational graph, with the definition of the adjoint being presented in Eq. (4), where $\mathbf{z}(t)$ is the output of the hidden state:

$$\mathbf{a}(t) = \frac{\partial \mathcal{L}}{\partial \mathbf{z}(t)}. \tag{4}$$

The adjoint follows the differential equation shown in Eq. (5):

$$\frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t} = -\mathbf{a}(t)^{\mathrm{T}}\nabla_{\mathbf{z}}\mathcal{NN}(\mathbf{z}(t),\theta). \tag{5}$$

Chen et al. [18] showed that $\mathbf{a}(t_0)$ can be computed by performing integration backwards, starting from $\mathbf{a}(t_1)$. The challenge with this approach is that it presumes that information regarding the states $\mathbf{z}(t)$ have been stored during the forward pass and therefore would limit the inverse solver to evaluate at specific time points and not permit evaluation at different time points in comparison to the forward solver. Therefore, the backward solver recomputes backwards in time for any values of the hidden states $\mathbf{z}(t)$. The consequence of this is a constant memory cost, in contrast to the traditional backpropagation method used for neural networks. Ultimately, a final integral is computed to obtain the gradient of the loss with respect to the network parameters. This integral is shown in Eq. (6). It is worth emphasizing that the gradients, $\nabla_\theta\mathcal{NN}$ and $\nabla_{\mathbf{z}}\mathcal{NN}$, are computed through automatic differentiation [40]:

$$\nabla_\theta\mathcal{L} = \int_{t_1}^{t_0} \mathbf{a}(t)^{\mathrm{T}}\nabla_\theta\mathcal{NN}(\mathbf{z}(t),\theta)\mathrm{d}t. \tag{6}$$

Table 1
Initial values used to generate synthetic data and their corresponding levels in the experimental design in Fig. 1.

	Level	c_A [mol/L]	c_B [mol/L]	T [K]
Data-poor regime	+1	14	7	320
	−1	6	3	290
Data-rich regime	+1	14	7	320
	0	10	5	305
	−1	6	3	290

For very large problems this process is computationally demanding. This motivates the use of pre-training to reduce the expensive computational time by minimizing the number of epochs performing backpropagation through the ODE solver. Pre-training is used with the purpose of obtaining a lower total training time however with the requirement of achieving comparable generalization capabilities to the benchmark, which corresponds to a neural ODE without pre-training. Thus, when pre-training has been used, the neural network will have significantly more favorable network parameters compared to initializing a neural network using best practices. Pre-training is performed by training the network with respect to information about the numerical derivatives of the states together with the mean-squared error (MSE) as the loss function as defined in Eq. (7):

$$\mathcal{L}_{\text{deriv}}(\theta) = \frac{1}{NS\tau_{\text{obs}}} \sum_{j=1}^{N} \sum_{i=1}^{S} \sum_{k=1}^{\tau_{\text{obs}}} \left(y_{\text{deriv},j,i,k} - \mathcal{NN}_i\left(y_{\text{obs},j,1:S,k}, \theta\right) \right)^2. \tag{7}$$

Here y_{deriv} is the numerical derivative of the ground truth data. The numerical derivatives are estimated through spline interpolation and the finite difference method (FDM). The numerical derivatives are estimated through the central differencing scheme for the time points except for the first and final time points in order to obtain second order accuracy. Regarding the first and final time point, the numerical derivatives are estimated through forward and backward Euler method, respectively.

Algorithm 1 presents the algorithm used to perform pre-training and the termination criteria for the different training phases. Initially, pre-training is performed using Adam with a learning rate of 0.01 and switches to Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) when $\mathcal{L}_{\text{deriv,training}}$ is lower than $\dot{\mu}_{\text{threshold}}$ and $\mathcal{L}_{\text{data,validation}}$ is lower than $\mu_{\text{threshold}}$. The motivation for starting with Adam initially and not using BFGS directly is to improve robustness and stability properties since BFGS is appropriate for local minima search. Once pre-training has been concluded, the main training with Adam commences with a learning rate of 0.01, using the final network parameters from pre-training as initial parameters and minimizing Eq. (3). Lastly, the algorithm transitions to BFGS when $\mathcal{L}_{\text{data,training}}$ is less than $\kappa_{\text{threshold}}$. Subsequently, BFGS is employed to minimize the loss function with convergence defined as achieving a validation loss for the states below 0.005. The loss in the form of Eq. (3) for training data is represented as $\mathcal{L}_{\text{data,training}}$ and $\mathcal{L}_{\text{data,validation}}$ for the validation dataset. The loss in the form of Eq. (7) for the training data is represented as $\mathcal{L}_{\text{deriv,training}}$. It is worth emphasizing that $\dot{\mu}_{\text{threshold}}$, $\mu_{\text{threshold}}$, $\nu_{\text{threshold}}$, $\kappa_{\text{threshold}}$, $\upsilon_{\text{threshold}}$ are hyperparameters.

The benchmark for the data-rich regime is a neural ODE that uses Adam until the training loss is close to a minimum and switches to BFGS until convergence. In other words, it uses the same method as algorithm 1 but entirely omits the pre-training phase.

2.3. Discovering mass conservation and regularization

Based on formulations for conservation of mass, the solutions to the ODE system with given initial conditions will satisfy linear relations. That is, for any point in time, the data points $\boldsymbol{c} = [c_A, c_B, c_C, c_D, c_E]$ will satisfy relations in the following form, $\alpha c_A + \beta c_B + \gamma c_D + \delta c_D + \epsilon c_E = \phi$ where $\alpha, \beta, \gamma, \delta, \epsilon, \phi \in \mathbb{R}$. With $\boldsymbol{c} = [c_A, c_B, c_C, c_D, c_E]$ and $\boldsymbol{\xi} = (\alpha, \beta, \gamma, \delta, \epsilon)^T$, the equation is summarized in Eq. (8):

$$\mathbf{c}\boldsymbol{\xi} - \phi = 0. \tag{8}$$

To estimate the coefficients in $\boldsymbol{\xi}$ and discover the apparent number of linearly independent relations for each initial condition, we compute the numerical rank of the centered data matrix and estimate its null-space. The data matrix here excludes temperature and centered is defined as subtracting with the mean for each state variable for all datasets. The result is that we acquire a basis for the null-space $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_L$ where L is the numerical nullity (dimension of the kernel). The constant term in Eq. (8), is estimated as $m\boldsymbol{\xi}$ where m is the mean value of the data matrix over its rows. Below, we will use this notation for each initial condition $j \in \{1, \ldots, N\}$ and will apply the notation $\boldsymbol{\xi}_{j,l}$ and $\phi_{j,l}$ to denote the coefficients and constant term in the case for the jth initial condition.

Once the relations have been deduced, they can then be used as regularization as shown in Eq. (9). The linear relations evaluated on the training data should theoretically be as small as possible, therefore the resulting squared residual is minimized during training. τ_{null} is the number temporal points where the relation in Eq. (9) is evaluated thus, τ_{null} is a hyperparameter, which means it can be optimized, depending on the system dynamics. It is preferable that any deviations in $\mathcal{L}_{\text{data}}$ and the individual terms in Eq. (9) scale equally-weighted, which is achieved by penalizing the errors quadratically, and results in positive contributions from these terms in the loss function

$$\mathcal{L}_{\text{mass}}(\theta) = \sum_{l=1}^{L} \sum_{j=1}^{N} \sum_{k=1}^{\tau_{\text{null}}} \frac{\left(\text{ODESolve}_{(1:S-1)}\left(y_{\text{obs}}, j, 1:S, 0, \mathcal{NN}, t_0, t_k, \theta\right)\boldsymbol{\xi}_{j,l} - \phi_{j,l} \right)^2}{LN\tau_{\text{null}}}. \tag{9}$$

As the loss function, we use the sum of the terms in Eqs. (3) and (9) as shown below, where λ_{mass} is a hyperparameter:

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{data}}(\theta) + \lambda_{\text{mass}}\mathcal{L}_{\text{mass}}(\theta). \tag{10}$$

The embedding of conservation laws into the loss function should encourage the network to make predictions that are physically consistent. As benchmark, we employ a neural ODE that solely minimizes $\mathcal{L}_{\text{data}}$.

The early-stopping criterion is used as the termination criterion for training the neural networks for the data-poor regime. Attention should be drawn to the fact that since we are utilizing early-stopping as our termination criterion, the validation loss also incorporates the formulations for conservation of mass.

2.4. Data-generation

A mechanistic batch reactor model is constructed to generate synthetic data to train the neural network. The batch reactor system consists of three reactions and five chemical species according to the reaction scheme shown in Eqs. (11)–(13):

$$\mathrm{A} + \mathrm{B} \longrightarrow \mathrm{C}, \tag{11}$$

$$\mathrm{C} \longrightarrow 2\,\mathrm{E}, \tag{12}$$

$$2\,\mathrm{A} \longrightarrow \mathrm{D}. \tag{13}$$

Algorithm 1: Training with pre-training. The boxed region represents the algorithm used for the benchmark.

Input: Temporal state measurements
Output: Prediction of state variables
Parameters: θ, $\dot{\mu}_{\text{threshold}}$, $\mu_{\text{threshold}}$, $\nu_{\text{threshold}}$, $\kappa_{\text{threshold}}$, $\upsilon_{\text{threshold}}$

Initialize model parameters, θ;
Estimate numerical derivatives using FDM;
converged=false;
while $\mathcal{L}_{\text{deriv,training}} > \dot{\mu}_{\text{threshold}}$ ***or*** $\mathcal{L}_{\text{data,validation}} > \mu_{\text{threshold}}$ **do**
  Perform pre-training with Adam by minimizing $\mathcal{L}_{\text{deriv,training}}$;
end
while $\mathcal{L}_{\text{data,validation}} > \nu_{\text{threshold}}$ **do**
  Perform pre-training with BFGS by minimizing $\mathcal{L}_{\text{deriv,training}}$;
end

while $\mathcal{L}_{\text{data,training}} > \kappa_{\text{threshold}}$ **do**
  Perform training with Adam by minimizing $\mathcal{L}_{\text{data,training}}$;
end
while *not converged* **do**
  Perform training using BFGS by minimizing $\mathcal{L}_{\text{data,training}}$;
  if $\mathcal{L}_{\text{data,validation}} > \upsilon_{\text{threshold}}$ **then**
    converged=true;
  end
end

Details regarding the batch reactor model and their corresponding numerical values are provided in Appendix A.

For reference, we explicitly provide the linear conservation relations that happen to hold for the given system that is studied. There exists two linearly independent mass balances that are presented in Eqs. (14) and (15), where $c_{i,0}$ is the initial condition for the state variable i:

$$c_A - c_B + 2c_D = c_{A,0} - c_{B,0} + 2c_{D,0}, \tag{14}$$

$$2c_B + 2c_C + c_E = 2c_{B,0} + 2c_{C,0} + c_{E,0}. \tag{15}$$

2.5. Software

In this study, we evaluated computational efficiency using Python (3.10.9) with the PyTorch library for implementing neural networks, and torchdiffeq for solving neural ODEs, alongside the corresponding packages in Julia. We observed a significant reduction in computational time with Julia. Therefore, for this work, the Julia programming language (1.10.1) is utilized to develop the reactor model as efficiently as possible. We employed DifferentialEquations.jl for solving differential equations with Verner's 7/6 Runge–Kutta method for both generating the ground truth data and solving the neural ODEs. Additionally, we utilized Lux.jl for defining neural networks, DiffEqFlux.jl for solving the neural ODEs, Zygote.jl for performing automatic differentiation, LinearAlgebra.jl for computing the null spaces, OptimizationOptimisers.jl for performing optimization with Adam and lastly, Optim.jl for performing optimization with BFGS for refined local optimization.

3. Results and discussion

3.1. Network architectures

The choice of network architecture is based on stability and achieving an accurate generalization with a reasonable training time. With respect to the choice of activation function, $tanh(x)$ was evaluated; however, its convergence time was longer compared to ReLU-based activation functions such as GELU and Swish. Overall, Swish exhibited better stability characteristics relative to GELU, and was thus selected as the activation function for this study. The selection of the number of neurons is guided by the principles of the Fletcher-Gloss method [41], which is given by Eq. (16) where n_{input} represents the number of inputs to the neural network, n_{output} represents the number of outputs of the neural network and n_{hidden} represents the number of neurons in the hidden layer. With six inputs and outputs, this configuration supports the selection of 12 neurons in the hidden layer:

$$2\sqrt{n_{\text{input}}} + n_{\text{output}} \leq n_{\text{hidden}} \leq 2n_{\text{input}} + 1. \tag{16}$$

We investigate neural network architectures with hidden layers ranging from 1 to 3 as Table 2 displays. The test accuracy is computed through Eq. (17), where $SS_{\text{res,test}}$ equals the residual sum of squares for the test dataset and $SS_{\text{tot,test}}$ equals the total sum of squares for the test dataset:

$$R^2_{\text{test}} = 1 - SS_{\text{res,test}}/SS_{\text{tot,test}}. \tag{17}$$

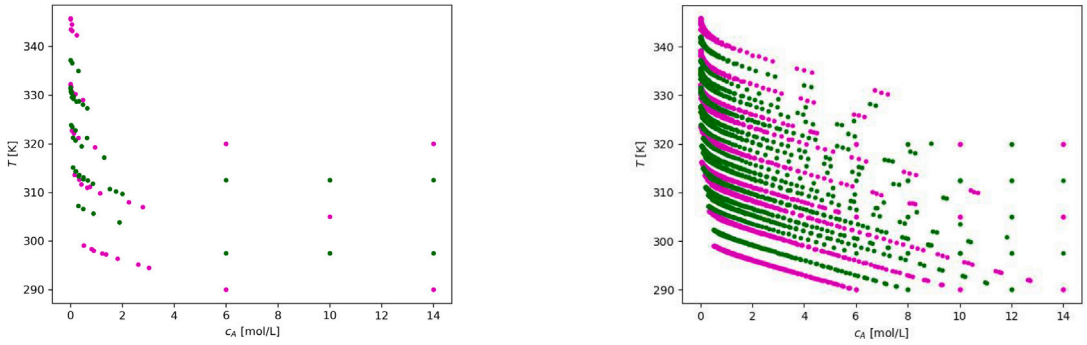
Table 2 demonstrates that a fair level of accuracy can be achieved with a single hidden layer. The median test accuracy improves when increasing the number of hidden layers from 1 to 2. Although increasing the number of hidden layers from 2 to 3 improves the test accuracy there is an increased computational cost. Therefore, for this study, two hidden layers are selected for the network architecture. Finally, we incorporate a pre-processing layer to our neural network that normalizes the inputs by subtracting them by their respective means and dividing by their standard deviations. Xavier initialization is used to initialize the network parameters.

3.2. Data regimes

Consider Eq. (A.1), where $\frac{dc_D}{dt}$ is a function of two variables, c_A and T. Fig. 2 presents an illustration of the two-data regimes for c_D as a function of its independent variables, c_A and T. The trained neural network will learn along the trajectories of the data points in the figure, thus minimizing the distance between the ground truth data and predicted values of the neural ODE. This figure further motivates the chosen structure for the experimental design, since theoretically the model should be capable of predicting along other trajectories in between the two extremes of the magenta data points. Comparing Fig. 2(a) to Fig. 2(b), the data in Fig. 2(a) is scarce and also sparse, which suggests an appropriate constraint should be introduced for improved

Table 2
Computational time and accuracy for different number of hidden layers with 12 neurons in each hidden layer with 5 observed data points and employing the experimental design presented in Fig. 1(a).

Number of hidden layers	R^2_{test}			Computational time (h)		
	Average	Median	Std Dev	Average	Median	Std Dev
1	0.89	0.89	0.06	0.27	0.24	0.12
2	0.89	0.92	0.06	0.36	0.29	0.18
3	0.91	0.93	0.06	0.55	0.49	0.24



(a) Data-poor regime (b) Data-rich regime

Fig. 2. A contour scatter plot of all observed data points of c_D for 2(a) data-poor regime and 2(b) data-rich regime. The data points in magenta represent training datasets and the data points in green represent validation datasets.

predictive power by regularizing the loss function. Meanwhile, the extensive amounts of data presented in Fig. 2(b) supports the rationale of introducing a method aimed at reducing the computational time.

3.3. Data-poor regime

3.3.1. Discovering conservation laws

By using the methods presented in Section 2.3, the conservation laws are discovered for the system described in Section 2.4. The conservation laws are not unique and will therefore be a linear combination of Eqs. (14) and (15). An example is that for the case with five temporal data points and the initial condition set of $(+1, -1, -1)$ in the experimental design, one obtained conservation law is presented in Eq. (18):

$$0.02c_A + 0.65c_B + 0.68c_C + 0.04c_D + 0.34c_E = 2.25. \tag{18}$$

It is possible to construct a matrix that vertically concatenates the numerical coefficients in Eqs. (14) and (15) together with the discovered scalars in $\boldsymbol{\xi}$ and the constant term as shown in Eq. (19):

$$\mathbf{C} = \begin{bmatrix} 1 & -1 & 0 & 2 & 0 & c_{A,0} - c_{B,0} + 2c_{D,0} \\ 0 & 2 & 2 & 0 & 1 & 2c_{B,0} + 2c_{C,0} + c_{E,0} \\ \alpha_1 & \beta_1 & \gamma_1 & \delta_1 & \epsilon_1 & \phi_1 \\ \alpha_2 & \beta_2 & \gamma_2 & \delta_2 & \epsilon_2 & \phi_2 \end{bmatrix}. \tag{19}$$

The total number of rows, minus two, corresponds to the number of discovered relations. To assess the accuracy of the discovered relations, we evaluate the numerical rank of this matrix. The numerical rank of this matrix can be determined by the number of singular values that exceed a specified threshold, with singular values that are less than this threshold indicating linear dependence among rows or columns. In the absence of noise in the observed data, it is expected that the numerical rank of the matrix below should be 2 since the objective is to discover 2 linearly independent relations. We provide the singular values of $\mathbf{C}$ in Appendix B for the case when the standard deviation of noise is 0.01

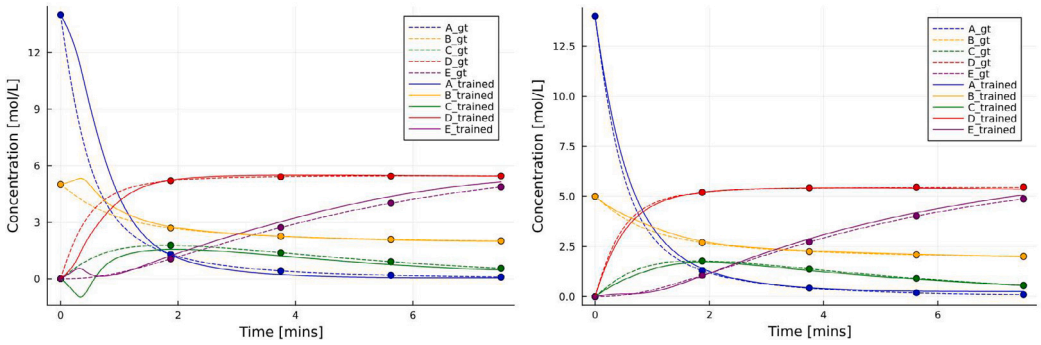
3.3.2. Effect of physics-regularization

All numerical results presented in this section represent the outcome of 20 independent runs i.e. replicates, each initialized with different network parameters. In the context of ablation studies, the benchmark serves as a reference point to assess the impact of the physics-regularization term in Eq. (10). By conducting a comparison of model performance against the benchmark, we can evaluate how physics-regularization together with the discovery of conservation laws contributes to the overall effectiveness of the model.

Fig. 3 demonstrates the effectiveness of physics-regularization compared to the benchmark. Since the observed data points are uniformly sampled, the key dynamics that take place in the initial phases of the reaction are not properly captured. It is critical to observe that there is a significant amount of information loss due to the poor sampling of observed data points. This is circumvented by utilizing physical constraints which help the model to accurately predict the evolution of individual species concentrations. Thus, adding physics-based information such as the conservation of mass enables the model to efficiently utilize constraints for regions that are not properly populated with observed data points. Due to the sparsity of the observed data points, the benchmark predicts negative concentrations in the initial phase of the reaction. Despite the sparse arrangement of the observed data points, our proposed method manages to make physically coherent predictions that also cohere well to the ground truth trajectories.

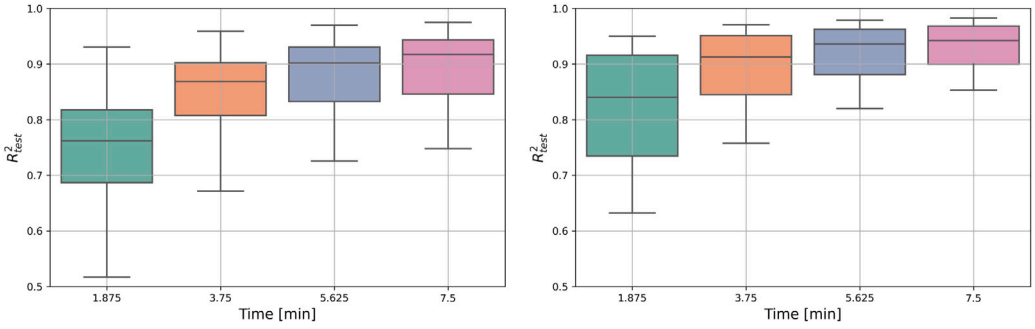
Given that the key dynamics occurs early in the reaction, assessing the accuracy up to a specific time point is of particular interest. Fig. 4 presents a box plot of the test accuracy up to different temporal data points. The figure demonstrates that for the key dynamics, our proposed model achieves both a higher median test accuracy and an improved best accuracy across the various time points. The motivation for our proposed model obtaining a better test accuracy in the key dynamics is that by using a logarithmic sampling for the collocation points, it is possible to use the discovered conservation laws to constrain the model, in order for the model to make physically feasible predictions. As Fig. 3 shows, the benchmark fails to properly capture the key dynamics, while our proposed method performs better to capture the key dynamics.

One aspect to consider in regards to the physics-regularization term presented in its entirety in Eq. (9) is that this term can be sampled in an arbitrary number of collocation points and in an arbitrary distribution of sampled collocation points. Sampling in a higher number of points



(a) Benchmark (b) With physics-regularization

Fig. 3. Concentration profile for 3(a) benchmark and 3(b) when physics-regularization is applied for a selected validation dataset. The dashed lines represent ground truth trajectories, the solid lines represent the prediction of the neural ODE and the scattered dots represent the specific time points for the observed data points.



(a) Benchmark (b) With physics-regularization

Fig. 4. A box-plot of the test accuracy up to different observed temporal data points for the various replicates of 4(a) benchmark and 4(b) when physics-regularization is applied.

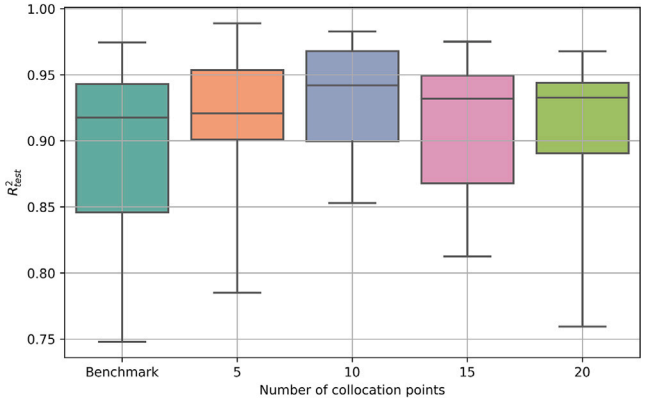


Fig. 5. Box plot for the test accuracy for a different number of collocation points with logarithmic distribution and $\lambda_{\text{mass}} = 1$ and 5 observed data points. The benchmark has zero collocation points.

has the consequence of longer training times due to backpropagation through the ODE solver. Different number of collocation points are considered ranging between 5–20 collocation points with increments of 5 collocation points as elucidated in Fig. 5. Despite increasing the number of collocation points to 20, the test accuracy decreases and peaks at 10 collocation points. The likely explanation for this is that with 10 logarithmically spaced collocation points, the sampled points manage to capture a significant portion of the key dynamics thus, providing the model with rich information of the key dynamics. By increasing the collocation points to 15 and 20 the model does not gain a lot of additional information, which consequences in a reduced test accuracy.

Regarding the distribution of points, there exists numerous different ways of choosing the distribution of sampled collocation points such as deterministic or random sampling. Some examples of deterministic sampling include sampling the collocation points uniformly or logarithmically. An example of random sampling methodologies include using latin hypercube sampling (LHS). Table 3 presents the test accuracy results using various sampling techniques for the collocation points.

As Table 3 shows, logarithmic spacing between the sampled collocation points result in the strongest predictive power. Using LHS and uniform sampling results in a larger standard deviation, which further motivates choosing a logarithmic spacing between the collocation points. The likely reason as to why logarithmic spacing is the most suitable is that it populates the earlier parts of the reaction dynamics compared to the later parts. Most reactions that take place with the initial conditions presented in Section 2.1, have a significant portion of their key dynamics with rapid changes earlier during the reaction, thus, reaching steady-state during the later parts of the sampling. With logarithmic sampling, it is possible to capture the key dynamics more efficiently since it emphasizes sampling earlier in the reactions and thus, capturing rapid changes, while minimizing unnecessary computations during steady-state. Notably, regardless of the chosen sampling methodology, the regularized model displays a better test accuracy than the benchmark. The sampling distribution presented here is system-specific and may differ for other chemical systems, such as reactions exhibiting time-delayed dynamics.

The optimal values for λ_{mass} are determined using a grid search over a predefined range of values between $[10^{-2}, 10]$, selecting the value for the hyperparameter that maximizes the validation accuracy.

Table 3
Test accuracy for different collocation sampling techniques with 10 collocation points, $\lambda_{mass} = 1$ and 5 observed data points.

Collocation sampling technique	R^2_{test}		
	Average	Median	Std Dev
Logarithmic sampling	0.93	0.94	0.04
Uniform sampling	0.91	0.93	0.06
LHS	0.92	0.93	0.06
Benchmark	0.89	0.92	0.06

Table 4
Test accuracy and computational time for physics-regularized approach and benchmark for various number of observed data points. 10 logarithmically distributed collocation points are utilized for the case with physics-regularization.

Method	Number of observed data points	R^2_{test}			Computational time (h)		
		Average	Median	Std Dev	Average	Median	Std Dev
Physics-regularized	5	0.93	0.94	0.04	0.36	0.34	0.14
	6	0.95	0.95	0.03	0.41	0.40	0.14
	7	0.96	0.97	0.02	0.50	0.42	0.23
	8	0.97	0.97	0.01	0.43	0.36	0.21
	10	0.97	0.97	0.02	0.46	0.41	0.25
Benchmark	5	0.89	0.92	0.06	0.36	0.29	0.18
	6	0.94	0.95	0.03	0.36	0.29	0.20
	7	0.95	0.97	0.05	0.42	0.39	0.17
	8	0.95	0.97	0.07	0.42	0.34	0.23
	10	0.96	0.97	0.04	0.36	0.29	0.18

The chosen values for λ_{mass} are unity for all the different numbers of temporal data points of $(5, 6, 7, 8, 10)$. As Table 4 indicates, using physics-regularization leads to improved test accuracy. The largest effect of physics-regularization compared to the benchmark is observed when data is scarce, i.e. a low number of observed data points. The results indicate a general trend in which increasing the number of observed data points enhances the model's generalization capability, as anticipated. Despite increasing the number of data points, the benchmark does not achieve a better average test accuracy than the physics-regularized approach for any number of observed data points. The physics-regularized approach demonstrates a lower standard deviation than the benchmark, indicating that our proposed method can also be used a form of standard deviation reduction. The explanation for this occurrence is that when using a constraint, the flexibility of the neural network is reduced. The consequence of this is a decreased likelihood for the neural network to explore extreme regions of the solution space, which leads to more consistent results and thus, a lower standard deviation. Nevertheless, since Eq. (9) requires backpropagation through the ODE solver, it is expected that there will be a larger computational cost to our proposed method. Our proposed technique incurs a minor increased computational cost, as shown in Table 4. In the worst case scenario for 10 observed data points, our method requires 41% longer median computational time than the benchmark. Notably, the best case scenario for our method demonstrates comparable median computational times to the benchmark such as for 7 and 8 observed data points. The general trend for both methods indicate that computational time increases up to a certain number of observed data points and thereafter begins to decrease. This can be explained that by increasing the number of data points, the information in the data will become more rich since it captures more of the key dynamics which results in an increased convergence rate and thus, a lower computational cost.

An additional aspect to consider is how our proposed method performs when the data quality deteriorates by introducing absolute Gaussian noise. We have made adjustments to the absolute Gaussian noise by scaling it by a factor of 10 when applied to temperature, to account for the differences in magnitude between temperature and concentration. Furthermore, all negative noisy values are excluded to ensure physical consistency, this applies solely to the concentration datasets. We introduce Gaussian noise with standard deviations of 0.01, 0.02 and 0.05. The test dataset used to evaluate test accuracy when trained on noisy data, as described in Section 2.1, is devoid of Gaussian noise. This is done to capture the coherence between the ground truth trajectories in the full factorial design and the predicted trajectories. The validation dataset incorporates Gaussian noise consistent with that of the training dataset. Table 5 presents a comparison of the benchmark to our proposed model. In Appendix B, the discovery of conservation laws for noisy data is presented and the singular values for $\mathbf{C}$ in Eq. (19) are significantly larger than the case for ideal data. This suggests that the identified conservation laws are significantly worse approximations rather than the approximative relations identified in the ideal case. Despite this, Table 5 shows that a good test accuracy is obtained with our proposed method for small standard deviations of noise. With a standard deviation of 0.01 for Gaussian noise, both methods obtain better test accuracies than the ideal case due to Gaussian noise acting as implicit regularization through controlled perturbations. The controlled variabilities can prevent the model from overfitting to the precise structure of the ideal data, which encourages the neural network to learn the general structure of the data. Additionally, for this reason our proposed method achieves a standard deviation that is lower than the ideal case. As the standard deviation of Gaussian noise increases, it is observed that the test accuracy of both methods begin to deteriorate. Notably, when estimating the null space of the centered ground truth concentrations, only one relation is identified for larger levels of noise for the majority of the datasets, rather than two relations which is discovered for lower standard deviations of noise. Despite this, our proposed method exhibits a better test accuracy than the benchmark for all noise levels.

Model training was conducted using synthetic data designed to replicate real-world conditions with concentrations measured by gas chromatography and temperatures recorded via thermocouples. To evaluate the robustness of the physics-regularized model, two noise components were introduced to the actual signals: (i) a relative Gaussian error (1%–5% relative standard deviation) to simulate the precision of gas chromatography, and (ii) absolute homoscedastic Gaussian noise (up to 0.5 °C) added to the temperature profile. This controlled augmentation of noise at varying levels enables a detailed and comprehensive assessment of the model's ability to learn the system dynamics from data of different qualities. The results of the analysis is summarized in Table 6. Consistent with results from ideal datasets and datasets with absolute noise, our method shows better model generalization

Table 5
Test accuracy and computational time for physics-regularized approach and benchmark for various standard deviations of absolute Gaussian noise. 10 logarithmically distributed collocation points are utilized for the case with physics-regularization with $\lambda_{mass} = 1$. 5 observed data points are used as training and validation datasets.

Method	Noise standard deviation	R^2_{test}			Computational time (h)		
		Average	Median	Std Dev	Average	Median	Std Dev
Physics-regularized	0.01	0.94	0.94	0.02	0.41	0.45	0.22
	0.02	0.93	0.92	0.04	0.34	0.34	0.09
	0.05	0.85	0.87	0.09	0.35	0.36	0.11
Benchmark	0.01	0.89	0.93	0.09	0.32	0.32	0.10
	0.02	0.89	0.92	0.07	0.22	0.23	0.05
	0.05	0.81	0.84	0.11	0.27	0.24	0.09

Table 6
Test accuracy and computational time under varying noise conditions. Temperature measurements use absolute Gaussian noise, while concentration measurements use relative noise proportional to the true concentration value. 10 logarithmically distributed collocation points are utilized for the case with physics-regularization with $\lambda_{mass} = 1$. 5 observed data points are used as training and validation datasets.

Method	Noise Parameter (σ) Temp: $\sigma_T = 10\sigma$ | Conc: $\sigma_C = \sigma \times c_i$	R^2_{test}			Computational time (h)		
		Average	Median	Std Dev	Average	Median	Std Dev
Physics-regularized	0.01	0.92	0.94	0.04	0.30	0.28	0.09
	0.02	0.91	0.93	0.05	0.33	0.29	0.16
	0.05	0.79	0.85	0.14	0.31	0.29	0.09
Benchmark	0.01	0.90	0.91	0.04	0.38	0.37	0.11
	0.02	0.89	0.91	0.07	0.25	0.23	0.07
	0.05	0.80	0.83	0.15	0.25	0.21	0.12

Table 7
Test accuracy and computational time for the physics-regularized approach and benchmark for various numbers of observed data points when introducing Gaussian noise with a noise parameter of $\sigma = 0.05$. Temperature measurements use absolute Gaussian noise, while concentration measurements use relative noise proportional to the true concentration value. 10 logarithmically distributed collocation points are utilized for the case with physics-regularization with $\lambda_{mass} = 1$. 5 observed data points are used as training and validation datasets.

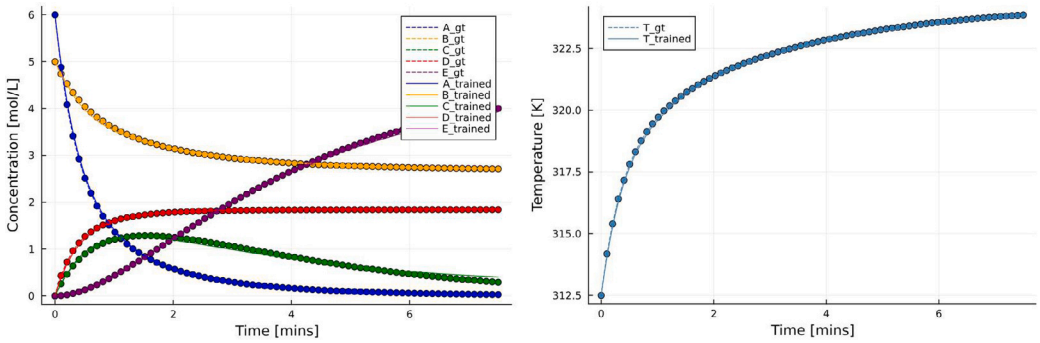
Method	Number of observed data points	R^2_{test}			Computational time (h)		
		Average	Median	Std Dev	Average	Median	Std Dev
Physics-regularized	5	0.79	0.85	0.14	0.31	0.29	0.09
	6	0.87	0.91	0.10	0.27	0.27	0.08
	7	0.90	0.92	0.06	0.31	0.29	0.08
	8	0.91	0.93	0.05	0.38	0.39	0.09
	10	0.92	0.94	0.08	0.33	0.33	0.11
Benchmark	5	0.80	0.83	0.15	0.25	0.21	0.12
	6	0.84	0.90	0.13	0.33	0.26	0.20
	7	0.84	0.89	0.13	0.21	0.20	0.05
	8	0.87	0.92	0.13	0.23	0.23	0.07
	10	0.89	0.92	0.08	0.27	0.26	0.08

in terms of higher median R^2_{test} values and lower standard deviations compared to the benchmark. The general trend of the model accuracy is that when the noise increases, the test accuracy decreases. However, for the largest noise level, the difference between the two methods diminishes. This phenomenon can be attributed to the limited number of temporal data points. With more data, a physics-regularized model can better detect the underlying conservation laws. Consequently, to further illustrate the robustness of the physics-regularized method, we evaluate its performance with an increased volume of observed data points.

As shown in Table 7, model accuracy increases monotonically with the number of observed data points and an increased model accuracy for the physics-regularized model compared to the benchmark. Additionally, the physics-regularized method shows a lower standard deviation demonstrating its usefulness for practical applications.

3.4. Data-rich regime

Development of modern online chemical sensors have lead to an increasingly faster sampling rate, which means access to process data has become ubiquitous. To leverage the power of online chemical sensors, this section studies a data-rich regime with 75 evenly spaced data points in time. In conjunction with more datasets than the data-poor regime as elucidated in Figs. 1 and 2, it is expected that neural ODEs should obtain a strong predictive power for the test datasets generated using a Monte-Carlo simulation. Fig. 6 presents the concentration and temperature profiles for the data-rich scenario. Although pre-training is used to obtain the results in Fig. 6, it is worth emphasizing that all replicates for both our proposed method in the data-rich regime and the benchmark demonstrates a test accuracy was larger than 0.985. As shown in Fig. 7, utilizing pre-training results in a significant reduction in computational time. By employing pre-training, the median computational time was reduced by 68%. Notably, there is solely one replicate for when pre-training is utilized that incurs a higher computational time than for any benchmark replicate. This elucidates the effectiveness of utilizing pre-training when training neural ODEs. Fig. 7 also shows that by employing pre-training, a smaller standard deviation can be obtained for the computational times in regards to the smaller interquartile range compared to the benchmark. Thus, by



(a) Concentration profile (b) Temperature profile

Fig. 6. Subfigures showing the results obtained using pre-training: 6(a) concentration profile and 6(b) temperature profile. The solid lines represent the prediction of the neural ODE, the dashed lines represent the ground truth trajectories and the scattered points represent the observed ground truth data points.

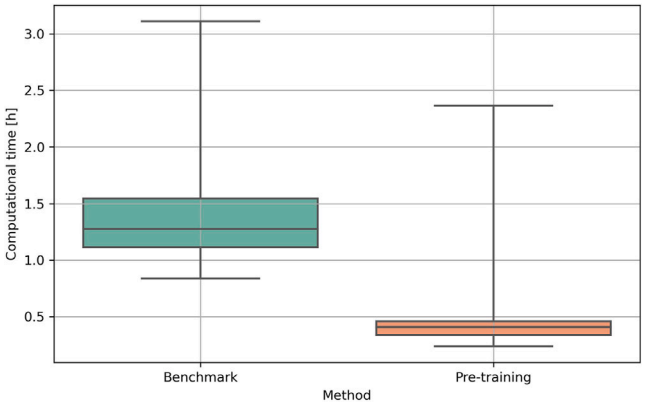


Fig. 7. Computational time for benchmark and when using pre-training.

utilizing pre-training, it is possible to obtain more robust training times compared to the benchmark.

4. Conclusion

This study investigates enhancing scientific machine learning models by incorporating fundamental constraints, automatically discovered from data to create a hybrid reactor model. The results illustrate how unknown conservation laws can be discovered from data and embedding them into neural ODEs as soft constraints to enhance the generalization and robustness of chemical reactor models. These conservation laws, which are not known *a priori*, are automatically discovered from data and integrated with neural ODEs. By assessing the null space of the ground truth data for the concentration of the chemical species, these laws are identified, thereby enhancing model generalization. The findings show that embedding physics-regularization not only improves generalization when data is scarce, but also enhances robustness. Furthermore, we demonstrate that when sampling fails to capture key dynamics, the model can still accurately predict the evolution of individual species concentrations. This finding suggests that information loss resulting from inadequate sampling can be effectively mitigated by employing our proposed method. With a logarithmic distribution of collocation points, the earlier parts of the reaction are populated with collocation points to ensure that the conservation of mass is adhered to in the time points where observed data points are not available and the key dynamics are taking place. The robustness of the proposed method was further verified by introducing noise into the training and validation datasets. Across all tested noise levels, our proposed method consistently outperformed the benchmark, demonstrating its strength. The method's reliance on mass conservation assumes that all chemical species are measured and known. In practical applications, this constraint requires robust species detection methods to ensure all relevant chemical components are measured to enable the use of physics-regularization via discovery of conservation laws. Normal level of noise is manageable as shown in the results section. An outlier in measurements is removed from the analysis, similar to other regression techniques. This underscores the benefit of the neural ODE methodology as it does not require uniform temporal measurements, which is more challenging with other techniques such as Long-Short-Term Memory (LSTM) networks. Despite our proposed method discovering only one formulation for conservation of mass for high levels of noise, it consistently yields better generalization capability than the benchmark. The demonstrated robustness of our proposed method when trained on noisy data highlights its potential for broad applicability to real-world datasets.

Owing to the significant advancements of digitalization in chemistry, we also investigate the scenario of a data-rich regime where generalization capabilities are not the main concern, instead the objective is to efficiently minimize the total computational time without losing any predictive power. By introducing pre-training prior to the main training we manage to cut the total computational time by 68% without comprising accuracy. This demonstrates the method's practicality in real-world applications in contexts where digitalization is a prevailing focus.

CRediT authorship contribution statement

Kian Hajireza: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **David Eklund:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Ronnie Andersson:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: David Eklund reports financial support was provided by Swedish Foundation for strategic research. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Financial support from the Swedish Foundation for Strategic Research, Sweden (Grant : FID20-0002) is gratefully acknowledged.

Table A.8
Physical and chemical parameters.

Variable	Symbol	Value	Unit
Reaction Enthalpy 1	ΔH_1	−12.0	kJ mol^{-1}
Reaction Enthalpy 2	ΔH_2	−9.60	kJ mol^{-1}
Reaction Enthalpy 3	ΔH_3	−24.0	kJ mol^{-1}
Pre-exponential Factor 1	$k_{0,1}$	3.00	$\times$ 10^{7}L/mol/min
Pre-exponential Factor 2	$k_{0,2}$	3.00	$\times$ 10^{7}min^{-1}
Pre-exponential Factor 3	$k_{0,3}$	3.00	$\times$ 10^{7}L/mol/min
Activation Energy 1	$E_{a,1}$	51.00	kJ mol^{-1}
Activation Energy 2	$E_{a,2}$	49.00	kJ mol^{-1}
Activation Energy 3	$E_{a,3}$	50.00	kJ mol^{-1}
Heat Capacity	c_p	4.0	kJ kg^{-1} K^{-1}
Density	ρ	2.0	kg L^{-1}

Table B.9
The first singular value for different training datasets for data corrupted by Gaussian noise with a standard deviation of 0.01.

Number of observed data points	Smallest $\times$ 10^{-5}	Median $\times$ 10^{-3}	Largest $\times$ 10^{-2}
5	5.11	1.12	2.21
6	3.22	1.39	2.46
7	4.12	1.40	3.40
8	4.38	1.67	2.65
10	6.63	1.69	2.20
20	6.94	1.79	2.68
40	8.71	1.50	3.72

Table B.10
The second singular value for different training datasets for data corrupted by Gaussian noise with a standard deviation of 0.01.

Number of observed data points	Smallest $\times$ 10^{-4}	Median $\times$ 10^{-2}	Largest $\times$ 10^{-1}
5	10.4	1.19	4.18
6	6.77	1.65	2.69
7	4.28	2.11	3.16
8	5.21	2.57	2.77
10	6.92	3.18	2.98
20	13.8	3.87	2.41
40	16.5	2.97	3.94

Table B.11
The third singular value for different training datasets for data corrupted by Gaussian noise with a standard deviation of 0.01.

Number of observed data points	Smallest	Median	Largest
5	1.05	1.08	3.32
6	1.05	1.09	3.32
7	1.05	1.09	3.32
8	1.05	2.60	3.32
10	1.05	2.60	3.32
20	1.06	3.04	3.32
40	1.07	3.04	3.32

Appendix A. Synthetic data generation

The list of physical and chemical parameters used to generate data using Eq. (A.1) are presented in Table A.8.

Eq. (A.1) shows the governing equations that are used to generate synthetic data:

$$\begin{aligned}
\frac{dc_A}{dt} &= -k_1 c_A c_B - 2k_3 c_A^2,\\
\frac{dc_B}{dt} &= -k_1 c_A c_B,\\
\frac{dc_C}{dt} &= k_1 c_A c_B - k_2 c_C,\\
\frac{dc_D}{dt} &= k_3 c_A^2,\\
\frac{dc_E}{dt} &= 2k_2 c_C,\\
\frac{dT}{dt} &= -\frac{k_1 c_A c_B \Delta H_1 + k_2 c_C \Delta H_2 + k_3 c_A^2 \Delta H_3}{\rho c_p}.
\end{aligned} \tag{A.1}$$

The reaction rate coefficient is expressed through an Arrhenius expression as shown in Eq. (A.2), where the subscript r denotes the reaction number:

$$k_r = k_{0,r} exp\left(-\frac{E_{a,r}}{RT}\right). \tag{A.2}$$

Appendix B. Singular values and conservation laws

The singular values for the matrix in Eq. (19) have been computed for different number of observed data points. Tables B.9–B.11 demonstrate that for small standard deviations of Gaussian noise it is possible in certain cases to discover approximations of the relations for the conservation laws. However it should be noted that the singular values are considerably larger than the case when no noise is present which was in the order of magnitude of 10^{-14}. The singular values in Tables B.9–B.11 are a result of 100 different replicates, where the smallest, median and largest singular values are determined for all replicates. With reference to Table B.9 and B.10, it is challenging to assert that the correct conservation law has been identified due to that the largest singular value in both cases have a large magnitude, therefore the discovered relations are approximations to the true conservation laws.

Data availability

Data will be made available on request.

References

[1] X. Shi, Y. Xiang, L.-X. Wen, J.-F. Chen, CFD analysis of liquid phase flow in a rotating packed bed reactor, Chem. Eng. J. 228 (2013) 1040–1049, http://dx.doi.org/10.1016/j.cej.2013.05.081.

[2] M. Tatar, V. Vashisth, M. Iqbal, T. Butterworth, G. van Rooij, R. Andersson, Analysis of a plasma reactor performance for direct nitrogen fixation by use of three-dimensional simulations and experiments, Chem. Eng. J. 497 (2024) 154756, http://dx.doi.org/10.1016/j.cej.2024.154756.

[3] S. Soltani, R. Andersson, B. Andersson, Enhancement of time resolution in transient kinetics, Chem. Eng. J. 264 (2015) 188–196, http://dx.doi.org/10.1016/j.cej.2014.10.098.

[4] V. Venkatasubramanian, The promise of artificial intelligence in chemical engineering: Is it here, finally? AIChE J. 65 (1) (2019).

[5] R.-L. Tseng, P.-H. Wu, F.-C. Wu, R.-S. Juang, A convenient method to determine kinetic parameters of adsorption processes by nonlinear regression of pseudo-nth-order equation, Chem. Eng. J. 237 (2014) 153–161, http://dx.doi.org/10.1016/j.cej.2013.10.013.

[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444, http://dx.doi.org/10.1038/nature14539.

[7] G. Cybenko, Approximation by superpositions of a sigmoidal function. Math cont sig syst (MCSS) 2:303-314, Math. Control Signals Systems 2 (1989) 303–314, http://dx.doi.org/10.1007/BF02551274.

[8] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Netw. 3 (5) (1990) 551–560, http://dx.doi.org/10.1016/0893-6080(90)90005-6.

[9] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257, http://dx.doi.org/10.1016/0893-6080(91)90009-T.

[10] M. Leshno, V.Y. Lin, A. Pinkus, S. Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Netw. 6 (6) (1993) 861–867, http://dx.doi.org/10.1016/S0893-6080(05)80131-5.

[11] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, S. Lee, Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence, 2019, http://dx.doi.org/10.2172/1478744.

[12] K. Yamaguchi, Y. Matsukawa, Y. Numazawa, H. Aoki, Modeling of water gas shift reaction using neural network trained on detailed kinetic mechanisms, Chem. Eng. J. 491 (2024) 151659, http://dx.doi.org/10.1016/j.cej.2024.151659.

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

[14] J. Jawad, A.H. Hawari, S. Javaid Zaidi, Artificial neural network modeling of wastewater treatment and desalination using membrane processes: A review, Chem. Eng. J. 419 (2021) 129540, http://dx.doi.org/10.1016/j.cej.2021.129540.

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

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

[17] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal differential equations for scientific machine learning, 2021, arXiv:2001.04385.

[18] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural ordinary differential equations, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS '18, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 6572–6583.

[19] S. Kim, W. Ji, S. Deng, Y. Ma, C. Rackauckas, Stiff neural ordinary differential equations, Chaos: An Interdiscip. J. Nonlinear Sci. 31 (9) (2021) http://dx.doi.org/10.1063/5.0060697.

[20] J. Yin, J. Li, I. Karimi, X. Wang, Generalized reactor neural ODE for dynamic reaction process modeling with physical interpretability, Chem. Eng. J. 452 (2022) 139487, http://dx.doi.org/10.1016/j.cej.2022.139487.

[21] A. Fedorov, A. Perechodjuk, D. Linke, Kinetics-constrained neural ordinary differential equations: Artificial neural network models tailored for small data to boost kinetic model development, Chem. Eng. J. 477 (2023) 146869, http://dx.doi.org/10.1016/j.cej.2023.146869.

[22] W. Bradley, F. Boukouvala, Two-stage approach to parameter estimation of differential equations using neural ODEs, Ind. Eng. Chem. Res. 60 (2021) http://dx.doi.org/10.1021/acs.iecr.1c00552.

[23] X. Su, W. Ji, J. An, Z. Ren, S. Deng, C.K. Law, Kinetics parameter optimization of hydrocarbon fuels via neural ordinary differential equations, Combust. Flame 251 (2023) 112732, http://dx.doi.org/10.1016/j.combustflame.2023.112732.

[24] F. Sorourifar, Y. Peng, I. Castillo, L. Bui, J. Venegas, J. Paulson, Physics-enhanced neural ordinary differential equations: Application to industrial chemical reaction systems, Ind. Eng. Chem. Res. 62 (2023) http://dx.doi.org/10.1021/acs.iecr.3c01471.

[25] T. Kircher, F.A. Döppel, M. Votsmeier, Global reaction neural networks with embedded stoichiometry and thermodynamics for learning kinetics from reactor data, Chem. Eng. J. 485 (2024) 149863, http://dx.doi.org/10.1016/j.cej.2024.149863.

[26] W. Ji, F. Richter, M.J. Gollner, S. Deng, Autonomous kinetic modeling of biomass pyrolysis using chemical reaction neural networks, Combust. Flame 240 (2022) 111992, http://dx.doi.org/10.1016/j.combustflame.2022.111992.

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

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

[29] E. Kaiser, J.N. Kutz, S.L. Brunton, Discovering conservation laws from data for control, in: 2018 IEEE Conference on Decision and Control, CDC, IEEE, 2018, pp. 6415–6421, http://dx.doi.org/10.1109/CDC.2018.8618963.

[30] X. Li, C. Zhang, R.R. Almeev, F. Holtz, GeoBalance: An excel VBA program for mass balance calculation in geosciences, Geochemistry 80 (2) (2020) 125629, http://dx.doi.org/10.1016/j.chemer.2020.125629.

[31] F.A. Döppel, M. Votsmeier, Robust mechanism discovery with atom conserving chemical reaction neural networks, Proc. Combust. Inst. 40 (1–4) (2024) 105507, http://dx.doi.org/10.1016/j.proci.2024.105507.

[32] B. Klumpers, T. Luijten, S. Gerritse, E. Hensen, I. Filot, Direct coupling of microkinetic and reactor models using neural networks, Chem. Eng. J. 475 (2023) 145538, http://dx.doi.org/10.1016/j.cej.2023.145538.

[33] T. Kumar, A. Kumar, P. Pal, A physics-constrained neuralode approach for robust learning of stiff chemical kinetics, in: NeurIPS Machine Learning and the Physical Sciences Workshop, 2023.

[34] D. Rehman, J.H. Lienhard, Physics-informed deep learning for multi-species membrane separations, Chem. Eng. J. 485 (2024) 149806, http://dx.doi.org/10.1016/j.cej.2024.149806.

[35] T. Beucler, M. Pritchard, S. Rasp, J. Ott, P. Baldi, P. Gentine, Enforcing analytic constraints in neural networks emulating physical systems, Phys. Rev. Lett. 126 (9) (2021) 098302, http://dx.doi.org/10.1103/PhysRevLett.126.098302.

[36] E. Roesch, C. Rackauckas, M. Stumpf, Collocation based training of neural ordinary differential equations, Stat. Appl. Genet. Mol. Biol. 20 (2021) http://dx.doi.org/10.1515/sagmb-2020-0025.

[37] P. Goyal, P. Benner, Neural ordinary differential equations with irregular and noisy data, R. Soc. Open Sci. 10 (2023) http://dx.doi.org/10.1098/rsos.221475.

[38] S. Gong, Q. Meng, Y. Wang, L. Wu, W. Chen, Z. Ma, T.-Y. Liu, Incorporating NODE with pre-trained neural differential operator for learning dynamics, Neurocomputing 528 (2023) 48–58, http://dx.doi.org/10.1016/j.neucom.2023.01.040.

[39] L.S. Pontryagin, E. Mishchenko, V. Boltyanskii, R. Gamkrelidze, The Mathematical Theory of Optimal Processes, Wiley-Interscience, 1962.

[40] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (153) (2018) 1–43.

[41] I.N.d. Silva, D.H. Spatti, R.A. Flauzino, Redes Neurais Artificiais Para Engenharia E Ciências Aplicadas, Artliber, 2010.