Extrapolation Problems in Kinetic Modelling of Catalytic Reactions with Neural Networks

Aleksandr Fedorov[*,†], David Linke[†]

Leibniz-Institut für Katalyse e. V., Albert-Einstein-Straße 29a, 18059 Rostock, Germany

Abstract

In the present study we investigated the behaviour of different types of neural networks in data extrapolation. The application is modelling a tubular chemical reactor in which a heterogeneous catalytic reaction (CO_2 hydrogenation to methanol) is performed. Since data are slow and expensive to measure we focused on small data sets for training. The different models (feed-forward neural network (NN), physics-informed NN, neural ordinary differential equation (ODE), kinetics-constrained neural ODE) were trained in a way to achieve approximately the same values of loss function in the cross-validation. Although the obtained models have the same generalization ability, the extrapolation capability varies significantly. Wherein, a neural network model that is additionally constrained by the general chemical and chemical engineering knowledge demonstrated much better extrapolation ability compared to unconstrained models. Methods how to validate the generalization of the neural network kinetic models without using additional experimental data were suggested and discussed.

Keywords

neural ODE, physics-informed neural networks, kinetics-constrained neural ODE, kinetic modelling, CO_2 hydrogenation

1. Introduction

The kinetic model development of catalytic reactions is one of the difficult but important part of chemical engineering for the process and industrial plant simulation [1]. The kinetic models of chemical reactions are ideally based on the knowledge of the reaction mechanisms. Due to complexity of the reaction mechanisms and the existence of a lot of parameters which have to be estimated from the data in data-driven kinetic modelling, several assumptions (postulating rate-determining stages, quasi-equilibrium approximation etc.) are usually used to convert the mechanisms to a set of reaction rate expressions for describing the dynamic of the process [2]. The complexity of this traditional approach is related to the necessity of having deep knowledge of the reaction mechanism as well as the difficulty of performing the screening of different possible assumptions. Notably, solving the inverse kinetic task (the estimation of kinetic model parameters from the data) is still a nontrivial task in the modelling process [3]. For these reasons, a machine learning approach is an attractive alternative for kinetic model development because only the data are needed to develop the models.

Neural networks are ones of popular methods of machine learning due to their flexibility which led to the development of a wide range of different types of neural network architectures [4, 5, 6, 7, 8] that are used in various fields. However, the overfitting problem of neural network models limits their wide usage especially in the case of small data that is typical available in the kinetic model development. Strategies for improving the neural network models significantly have been proposed, for example, by using the knowledge about the process via modifying the loss function [5, 9, 10] or by constraining the architecture of the models [11, 12, 13]. In our recently published work [13], we suggested a new approach for kinetic modelling with NNs. The approach is based on constraining neural ODE models by the general chemical and chemical engineering knowledge. This enabled us to obtain reliable NN models describing the complex process of CO_2 hydrogenation using only small data. During the method development we have found that different neural network models demonstrating the similar generalization ability in the cross validation differed significantly in their data prediction and extrapolation capabilities. Here, it is worth noting the difference between the generalization and extrapolation ability of the NN models. The generalization refers the ability to fit the test data that lies within the bounds of training dataset. By the extrapolation we mean in general the ability to predict the data at different residence time that lies out of the bounds of training data.

To investigate the observed phenomena in more details, in the present study we investigated the ability of different types of neural networks in the extrapolation of the kinetic catalytic data. For this, we generated small and large data sets for model training, validation, as well as for testing the model ability to extrapolate. The following different types of NN model were used: feed-forward

ITAT'24: Information technologies - Applications and Theory, September 20–24, 2024, Drienica, Slovakia

[*]Corresponding author.

[†]These authors contributed equally.

Aleksandr.Fedorov@catalysis.de (A. Fedorov); David.Linke@catalysis.de (D. Linke)

0000-0001-6434-6623 (A. Fedorov); 0000-0002-5898-1820 (D. Linke)

neural networks [4], physics-informed neural networks [5], neural ordinary differential equations (neural ODEs) [6], kinetics-constrained neural ODE [13]. To compare the neural networks based modelling approach with the traditional one, we also developed a simple power-law kinetic model assuming first order with respect to the reagents in the reaction rate expressions.

2. Experimental part

2.1. Data generation

For the data generation, the kinetic model of CO_2 hydrogenation to methanol was used from the work [14]. This kinetic model was based on the following chemical reactions:

$$CO_2 + 3H_2 \rightleftarrows CH_3OH + H_2O, \tag{1}$$

$$CO_2 + H_2 \rightleftarrows CO + H_2O, \tag{2}$$

$$CO_2 + 4H_2 \rightleftarrows CH_4 + 2H_2O. \tag{3}$$

A dual-site Langmuir-Hinshelwood kinetic model was used for describing the dynamics of the suggested reactions:

$$r_1 = \frac{k_1 \left(P_{CO_2} P_{H_2}^3 - \frac{P_{CH_3OH} P_{H_2O}}{K_{eq,1}} \right)}{P_{H_2}^2 \left(1 + K_{CO_2} \cdot P_{CO_2}\right) \left(1 + \sqrt{K_{H_2} \cdot P_{H_2}}\right)}, \tag{4}$$

$$r_2 = \frac{k_2 \left(P_{CO_2} P_{H_2} - \frac{P_{CO} P_{H_2O}}{K_{eq,2}} \right)}{\sqrt{P_{H_2}} \left(1 + K_{CO_2} \cdot P_{CO_2}\right) \left(1 + \sqrt{K_{H_2} \cdot P_{H_2}}\right)}, \tag{5}$$

$$r_3 = \frac{k_3 \sqrt{P_{CO_2} P_{H_2}}}{\left(1 + K_{CO_2} \cdot P_{CO_2}\right) \left(1 + \sqrt{K_{H_2} \cdot P_{H_2}}\right)}, \tag{6}$$

where r_i is the rate of the chemical reaction i; P_i is the partial pressure of compound i; k_i is the rate constant of the reaction i; $K_{eq,i}$ - the equilibrium constant of the reaction i; K_{CO_2} - the equilibrium constant of CO_2 adsorption; K_{H_2} - the equilibrium constant of dissociative adsorption of H_2. It is worth mentioning that the thermodynamic part of the reaction 3 was removed from the equation in the present study because the equilibrium constant of this reaction is very high ($\Delta_r G^\circ < 0$ and $|\Delta_r G^\circ|/RT \approx 19.6 \gg 1$, $K_{eq,3} = \exp\left(-\Delta_r G^\circ / RT\right) \approx 3.3 \cdot 10^8$). For comparison, the values of equilibrium constants of the reactions 1 and 2 are $2.6 \cdot 10^{-2}$ and $5.7 \cdot 10^{-6}$, respectively. The values are given for 573.15 K. The following equations were used to estimate the values of the equilibrium constants of the reactions 1 and 2:

$$\log_{10} K_{eq,1} = -\frac{2073}{T} + 2.029, \tag{7}$$

$$\log_{10} K_{eq,2} = \frac{3066}{T} - 10.592. \tag{8}$$

The rates constant were presented in the form of reparameterized Arrhenius equation:

$$k_i(T) = k_{i,0} \exp\left(\frac{E_i}{R}\left(\frac{1}{T_0} - \frac{1}{T}\right)\right), \tag{9}$$

where $k_{i,0}$ is the rate constant of reaction i at the reference temperature T_0 that was set to 573.15 K; E_i - activation energy of the reaction constant i; R - universal gas constant; T - temperature. The adsorption equilibrium constants were presented in the form of Van't Hoff equation:

$$K_i(T) = K_{i,0} \exp\left(\frac{\Delta H_i}{R}\left(\frac{1}{T_0} - \frac{1}{T}\right)\right), \tag{10}$$

where $K_{i,0}$ is the adsorption equilibrium constant at the reference temperature T_0 and ΔH_i is the molar change of the enthalpy. The parameters of the model were taken from the original work [14].

To generate the data for the present study, the following system of ODE was integrated:

$$\frac{dF_i}{d\tau} = \sum_{j=1}^{N} \nu_{ij} r_j, \tag{11}$$

where F_i is the molar flow of the compound i; τ is the residence time, N - is the number of the chemical reactions; ν_{ij} is the stoichiometric coefficient of the compound i in the reaction j (positive for the products and negative for reagents). The system of ODE represents the mathematical model of 1D plug flow reactor for catalytic reaction assuming the absence of heat and mass transfer limitations. The integration of the system 11 allows one to estimate the molar flows of compounds along the reactor (at different residence time).

Table 1 shows the reaction conditions used for generating the training data. For each reaction conditions, 5 points of the residence time (0.05, 0.1, 0.2, 0.4, 1.0) were used for the data generation. Thus, the total number of different reaction conditions was 13 ·5 = 65. The total inlet molar flow was set to a constant value of 1 for each reaction conditions (only CO_2 and H_2 were used in the inlet flow). Thus, the dataset represent the dependencies between the outlet molar flows of each compound (obtained by the integration of the system of ODE 11) and the reaction condition (initial molar flows of CO_2 and H_2, temperature, total pressure, and the residence time). The partial pressure of compounds (required for the estimation the reaction rates 4, 5, and 6.) was estimated by the following equation assuming the ideal gas law:

Table 1
The reaction conditions used for the training data generation.

#	Temperature, ˚C	Pressure, bar	H_2:CO_2 ratio
1	**200**	30	3
2	**250**	30	3
3	**300**	**30**	**3**
4	**350**	30	3
5	**400**	30	3
6	300	**20**	3
7	300	**25**	3
8	300	**35**	3
9	300	**40**	3
10	300	30	**1.5**
11	300	30	**2**
12	300	30	**4**
13	300	30	**6**

$$P_i = \frac{P \cdot F_i}{\sum_j F_j}, \tag{12}$$

where P is the total pressure; P_i - the partial pressure of compound i; F_i - the molar flow of compound i.

The CO_2 conversion X_{CO_2} and selectivity of i-compound S_i, which are the commonly used measures of reactor and catalyst performance, were estimated by the following equations that relate the state at reactor inlet to the state at reactor outlet:

$$X_{CO_2} = \frac{F^{in}_{CO_2} - F^{out}_{CO_2}}{F^{in}_{CO_2}}, \tag{13}$$

$$S_i = \frac{F^{out}_i}{F^{in}_{CO_2} - F^{out}_{CO_2}}. \tag{14}$$

To validate and investigate the extrapolation ability of the kinetic models, 4 different test datasets were generated. The 4 test datasets differed in the range of the residence time selection (0-1, 1-2, 2-5, and 5-10) to investigate the extrapolation ability of the investigated kinetic models. Each dataset had 1000 points which were generated by randomly selecting values from the ranges of the reaction conditions (temperature in 200-400 ˚C, pressure in 20-40 bar, H_2:CO_2 ratio in 1.5-6.0) using a uniform distribution. To train physics-informed neural network model, an additional dataset (labeled as the zero set) was created. The zero set was generated by a similar procedure applied for the test data generation except for setting the residence time to be 0. The zero dataset had also 1000 points.

2.2. Model architecture and training

Different variants of artificial neural networks were used to fit the data to CO_2 hydrogenation. It is important to note that the architecture and parameters of NN models (the number of layers and neurons, a type of activation function etc.) were chosen to achieve approximately the same value of loss function (Equation 17) for 10-fold cross-validation during the model training.

The first model was the simple feed-forward neural network denoted as **NN**. This model has 4 inputs (temperature, pressure, the inlet fraction of CO_2 and the residence time). Temperature was transformed in the following form and used as an input for neural network models:

$$T_{in} = \exp\left[\frac{1.1 \cdot 10^4}{R}\left(\frac{1}{T_0} - \frac{1}{T}\right)\right]. \tag{15}$$

By presenting the temperature in a such way, T_{in} was in a range of $\approx$ 0.61-1.41. The inlet molar fraction of carbon dioxide $x^{in}_{CO_2}$ was calculated by the following equation:

$$x^{in}_{CO_2} = \frac{1}{1+m}, \tag{16}$$

where m is the H_2:CO_2 ratio. The pressure was normalized by dividing by the maximum value of 40 bar. The **NN** model has 4 outputs for the molar flow of CO_2, CO, CH_4, and CH_3OH. The **NN** model has 1 hidden layer with 20 nodes and hyperbolic tangent activation function as well as the output layer with sigmoid activation function. To train the **NN** model, the following loss function was used:

$$loss_1 = \sum_{i \in \text{training set}} \left(\frac{F^{pred}_i - F^{exp}_i}{F_{i,scale}}\right)^2, \tag{17}$$

where F^{pred}_i is the molar flow of the compound i in the outlet predicted by the model; F^{exp}_i is the molar flow of the compound i in the outlet simulated by the kinetic model; $F_{i,scale}$ is a characteristic scale for compound i that was calculated based on the maximum value for the molar flow of compound i in the training dataset. This scaling was also helpful to mitigate the stiffness of neural ODE models [15] during the model training.

The second model was a physics-informed neural network (**PINN**). The architecture of this model was similar to the **NN** model except for a modification of the outlet of the **NN** model:

$$out^{PINN}_i = \frac{out^{NN}_i \cdot x^{in}_{CO_2}}{\sum_j out^{NN}_j}, \tag{18}$$

where out^{PINN}_i is the i-outlet of **PINN** model; out^{NN}_i is the i-outlet of **NN** model. For the training **PINN** model, we used the knowledge that there is no change in the molar flows of compounds when the residence time is to equal to 0. For this, an additional zero dataset was also used for training of **PINN** model and the corresponding loss function was:

$$loss_2 = loss_1 + \sum_{i \in \text{zero set}} \left(\frac{F_i^{pred} - F_i^{exp}}{F_{i,scale}} \right)^2. \quad (19)$$

The next model (**NODE**) was based on the neural ODE [6] that uses neural networks for approximating the right part of ODEs:

$$\frac{dF_i}{d\tau} = \text{ANN}(P_i, T), \quad (20)$$

where ANN is the feed-forwards neural network. The inputs of ANN were the partial pressure of compounds and the re-parameterized temperature (Equation 15). The **NODE** model has 7 inputs (partial pressure of compounds - CO_2, CO, CH_4, CH_3OH, H_2, H_2O as well as the re-parameterized temperature (Equation 15)). The **NODE** model consists of two hidden layers with hyperbolic tangent and exponential activation functions. The number of nodes was 3 for both layers. The output of the ANN was the linear function. The loss function 17 was used for training **NODE** model.

The kinetics-constrained neural ODE approach was used for developing the fourth model (**KCNODE**). The idea of the approach is to use the general knowledge about the process and approximate the rates of chemical reactions by the following equation:

$$r_i = k_i(T) \cdot \prod_j \left(P_i^j\right) \cdot \left(1 - \frac{Q}{K_{eq,i}}\right) \cdot \text{ANN}(P_i, T), \quad (21)$$

where $k_i(T)$ is the rate constant of the reaction i defined by Equation 9; P_i^j is the partial pressure of the reagent j in the reaction i. Q - the reaction quotient. For example, in the case of reaction 1 the corresponding quotient is:

$$Q_1 = \frac{P_{CH_3OH} P_{H_2O}}{P_{CO_2} P_{H_2}^3}. \quad (22)$$

The **KCNODE** model consists of 1 hidden layer with 3 nodes (hyperbolic tangent activation function) and the outlet layer which has sigmoid activation function. The sigmoid function is chosen for the output layer because it ensures positive value, and, thus, Equation 21 aligns with thermodynamic. So, if ($Q < K_{eq}$), the rate is positive (forward reaction), and if ($Q > K_{eq}$), the rate is negative (backward reaction).

It is worth noting that the activation energies and the rate constants were also parameters of the **KCNODE** model and were varied during the training of the neural network model, along with the weights and biases of the neural network layers.

The last model denoted as Power-Law (**PL**) model replaces the ANN part in the **KCNODE** model by simple power law rate expressions. Thus, it represents a simple conventional kinetic model that is based on the assumption that first order kinetics can sufficiently describe the reactions. The **PL** model was selected as baseline model and used for the comparison with the NN models. The loss function 17 was used for training **KCNODE** and **PL** models.

2.3. Hardware and software specifications

Dormand-Prince-Shampine method (DOPRI5) method was used for the integration of the neural ODE [16]. Training neural network models was carried out by minimizing the loss functions using ADAM [17] optimizer with a learning rate of 0.005. L_2-regularization was used. The parameter of the regularization was set to 10^{-6}. All calculation were implemented in the Python programming language (version 3.9.12) [18]. The scientific libraries NumPy [19] (version 1.23.0), SciPy [20] (version 1.8.1), Pandas [21] (version 1.23.0), Scikit-learn [22] (version 1.1.1) were used for data analysis and evaluation. Pytorch [23] (version 1.12.0) and Torchdyn [24] (version 1.0.3) were used for building and training neural networks models. Matplotlib [25] (version 3.7.2) was used to visualize the results.

3. Results and Discussion

The first step of the present study was to obtain the trained neural network models with the similar generalization ability. In our work, we have chosen the value of the loss function in the 10-folds cross validation (CV) as a metric of the generalization. Firstly, we estimated the CV value for the PL model which was $1.7 \cdot 10^{-3}$. To achieve a similar CV value of the loss function for all neural network models, we trained our models by minimizing the corresponding loss function. Figure 1 shows the dependency between the value of 10-folds CV of the loss function and the number of epochs. It can be seen that the CV loss function decreases with increasing number of epochs for all the neural network models except for the **NODE** one where we observe a significant rise in the value of the loss function after around 2000 epochs. From the obtained data we have found the number of iterations which is needed for training to achieve the corresponding value of loss function ($1.7 \cdot 10^{-3}$ in our case). It is worth noting that we did not manage to achieve this target value for the **NODE** model. For this model, the minimum value of the loss function ($2.1 \cdot 10^{-3}$) was therefore chosen which is still close to the targeted value. Thus, we managed to obtain a set of different neural network models with similar generalization ability.

To validate the generalization ability of machine learning models, an additional test dataset is typically used. However, for kinetic model development, chemical and

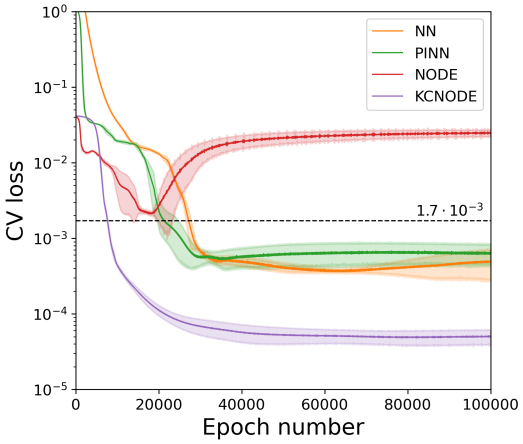

Figure 1: The dependencies between the value of 10-fold CV and the number of epochs during training for different models. In the case of neural ODE-based models (**NODE** and **KCNODE**), the number of epochs was multiplied with 10 for better visualization. The training process was repeated 5 times to estimate the standard deviation that is shown as error bar in the figure.

engineering knowledge can be utilized for validation. We simulated virtual data using the obtained models and compared them in Figure2. The analysis revealed prediction issues with the **NN** and **NODE** models, particularly with the carbon balance and molar flow predictions. The **NN** model predicted non-zero CH_3OH flow at zero residence time, and the NODE model predicted negative molar flows. These issues are illustrated by plotting CO_2 conversion and product selectivity, which should range from 0 to 1 but did not for **NN** and **NODE** models. In contrast, knowledge-integrating models (**PINN** and **KCNODE**) did not exhibit these problems.

To assess the approximation ability of the obtained models, the series of the different test datasets was generated. The datasets differed in the residence time range (from 0-1 to 5-10). The values of estimated loss function for the test datasets are presented in Table 2. One can see that the values of the loss function for the dataset with the residence time range of 0-1 are similar for all the models. However, when we try to predict data outside of the training dataset range, we observe that the values of the loss function increase with increasing residence time range for all the models. This increase varies significantly from model to model. The highest errors in the data prediction for a residence time range of 5-10 were found for the **NN** and **PINN** models. This is due to the fact that the models were trained on the dataset where the residence time varied in the range of 0-1. Another limitation of the **NN** and **PINN** models is that both models only represent the solution of the reactor model. It means that these models describe the solution of the system of ODE but not the kinetic model represented in the form of rate equations. This imposes serious restrictions on using the resulting models for up-scaling (modelling another type of reactor or extending the models by adding the diffusion/heat transfer).

The neural ODE approach does not have such a limitation since the neural ODE models represent the approximation of the right part of ODE. From Table 2 one can see that better extrapolation ability is observed for neural ODE models (**NODE** and **KCNODE**) compared to **NN** and **PINN** ones. Wherein, the loss function for the dataset generated with a residence time range of 5-10 was 0.66 for **KCNODE** model and around 5 times lower then the one for **NODE** model (3.4). Thus, the neural ODE model additionally constrained by the general chemical and chemical engineering knowledge demonstrates much better extrapolation ability. To compare the **KCNODE** model with a traditional approach, Table 2 shows the values of loss function for **PL** model. One can see that both **KCNODE** and **PL** models show similar predicting ability in fitting the test data. In addition, another kinetics-constrained neural ODE model (**KCNODE***) was obtained by training after 10000 epochs to achieve the minimum value of the CV loss function to compare with other models. The values of loss function for different test data are also presented in Table 2. On can see, that the the loss function for the dataset generated with a range of residence time in 5-10 is around 0.2 and decreased compared to **KCNODE** model.

4. Conclusions

In the present work, we investigated the behaviour of different neural network models (feed-forward NN, physics-informed NN, neural ODE, kinetics-constrained neural ODE) when applied for modelling catalytic data describing the process of CO_2 hydrogenation to methanol. To compare models under the same conditions, we trained the models in a way to achieve a similar value of the loss function in the 10-folds cross validation. Although the obtained models had similar generalization capability, we showed that only neural ODE model additionally constrained by the general chemical and chemical engineering knowledge demonstrate a good fitting of the data in the context of the residence time. Moreover, its capability to extrapolate was comparable to the traditional modelling approach. Due to the extrapolation ability, the application of neural ODE models for accelerated kinetic model development can be expected to grow.

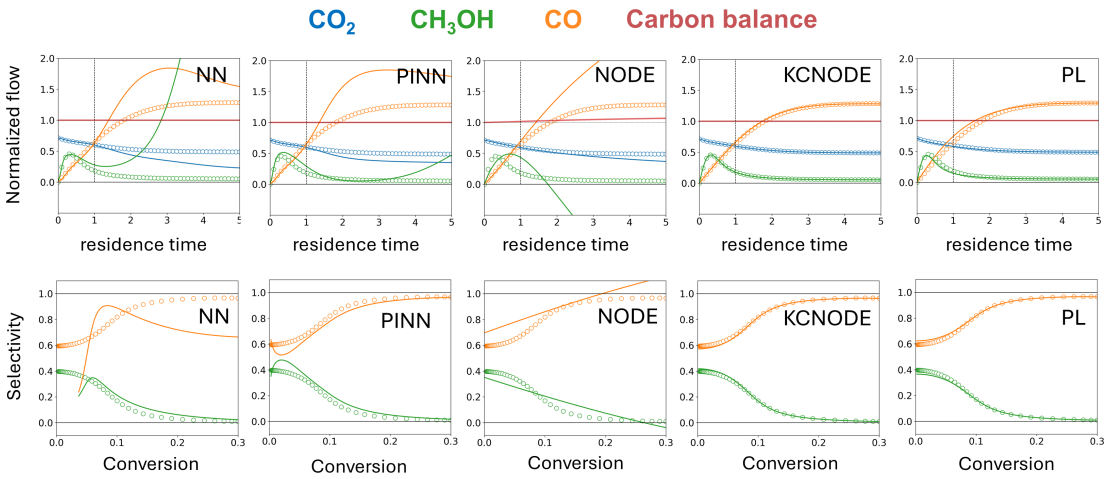

Figure 2: The observed (dots) and fitting (lines) dependencies between: the normalized molar flows of compounds $F_i/F_{i,scale}$ and the residence times τ (top); the CO_2 conversion and product selectivity (bottom) estimated by Equations 14 and 13, respectively. Carbon balance was defined as the sum of the molar flow of all carbon-containing compounds. The following reaction condition was used for the simulation: temperature was 375 °C, the total pressure was 32 bar, H_2:CO_2 ratio was 2.5.

Table 2
The comparison of the obtained models in the predicting the test datasets differing in the range of residence time selection. The standard deviation (relative value) is given in brackets

model	loss function				
	CV	0-1	1-2	2-5	5-10
NN	$1.7 \cdot 10^{-3}$ (±38%)	$2.6 \cdot 10^{-3}$ (±18%)	0.15 (±65%)	11 (±89%)	75 (±82%)
PINN	$1.7 \cdot 10^{-3}$ (±44%)	$1.7 \cdot 10^{-3}$ (±18%)	$9.6 \cdot 10^{-2}$ (±25%)	11 (±81%)	186 (±85%)
NODE	$2.1 \cdot 10^{-3}$ (±4.1%)	$2.8 \cdot 10^{-3}$ (±17%)	$2.0 \cdot 10^{-2}$ (±7.9%)	0.48 (±21%)	3.4 (±27%)
KCNODE	$1.7 \cdot 10^{-3}$ (±9.8%)	$4.7 \cdot 10^{-3}$ (±14%)	$2.5 \cdot 10^{-2}$ (±11%)	0.13 (±8.0%)	0.66 (±7.2%)
PL	$1.7 \cdot 10^{-3}$ (±0.1%)	$5.3 \cdot 10^{-3}$ (±0.09%)	$2.5 \cdot 10^{-2}$ (±0.07%)	0.13 (±0.4%)	0.67 (±0.5%)
KCNODE*	$5.0 \cdot 10^{-5}$ (±23%)	$4.4 \cdot 10^{-4}$ (±7.7%)	$3.9 \cdot 10^{-3}$ (±5.2%)	$3.5 \cdot 10^{-2}$ (±4.7%)	0.20 (±4.7%)

5. Acknowledgments

Financial support from German Research Foundation (DFG) through the project NFDI4Cat (DFG no. 441926934) is gratefully acknowledged.

References

[1] Yablonskii, G. S. Yablonskii, V. I. Bykov, V. I. Elokhin, A. N. Gorban, Kinetic models of catalytic reactions, Comprehensive Chemical Kinetics, Elsevier Science & Technology, 2014.

[2] L. Brübach, D. Hodonj, P. Pfeifer, Kinetic analysis of co2 hydrogenation to long-chain hydrocarbons on a supported iron catalyst, Industrial & Engineering Chemistry Research 61 (2022) 1644–1654. doi:10.1021/acs.iecr.1c04018.

[3] S. Matera, W. F. Schneider, A. Heyden, A. Savara, Progress in accurate chemical kinetic modeling, simulations, and parameter estimation for heterogeneous catalysis, ACS Catalysis 9 (2019) 6624–6647. doi:10.1021/acscatal.9b01234.

[4] G. Bebis, M. Georgiopoulos, Feed-forward neural networks, IEEE Potentials 13 (1994) 27–31. doi:10.1109/45.329294.

[5] 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, Journal of Computational Physics 378 (2019) 686–707. doi:https://doi.org/10.1016/j.jcp.2018.10.045.

[6] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018. URL: https:

//proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[7] G. Corso, H. Stark, S. Jegelka, T. Jaakkola, R. Barzilay, Graph neural networks, Nat. Rev. Methods Primers 4 (2024).

[8] Y. Yu, X. Si, C. Hu, J. Zhang, A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures, Neural Computation 31 (2019) 1235–1270. doi:10.1162/neco_a_01199.

[9] G. S. Gusmão, A. P. Retnanto, S. C. da Cunha, A. J. Medford, Kinetics-informed neural networks, Catalysis Today 417 (2023) 113701. doi:https://doi.org/10.1016/j.cattod.2022.04.002, transient Kinetics Seminar.

[10] Y. Weng, D. Zhou, Multiscale physics-informed neural networks for stiff chemical kinetics, The Journal of Physical Chemistry A 126 (2022) 8534–8543. doi:10.1021/acs.jpca.2c06513, pMID: 36322833.

[11] F. Sorourifar, Y. Peng, I. Castillo, L. Bui, J. Venegas, J. A. Paulson, Physics-enhanced neural ordinary differential equations: Application to industrial chemical reaction systems, Industrial & Engineering Chemistry Research 62 (2023) 15563–15577. doi:10.1021/acs.iecr.3c01471.

[12] T. Kircher, F. A. Döppel, M. Votsmeier, Global reaction neural networks with embedded stoichiometry and thermodynamics for learning kinetics from reactor data, Chemical Engineering Journal 485 (2024) 149863. URL: https://www.sciencedirect.com/science/article/pii/S1385894724013482. doi:https://doi.org/10.1016/j.cej.2024.149863.

[13] 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, Chemical Engineering Journal 477 (2023) 146869. doi:https://doi.org/10.1016/j.cej.2023.146869.

[14] S. Ghosh, J. Sebastian, L. Olsson, D. Creaser, Experimental and kinetic modeling studies of methanol synthesis from co_2 hydrogenation using in_2o_3 catalyst, Chemical Engineering Journal 416 (2021) 129120. doi:https://doi.org/10.1016/j.cej.2021.129120.

[15] S. Kim, W. Ji, S. Deng, Y. Ma, C. Rackauckas, Stiff neural ordinary differential equations, Chaos: An Interdisciplinary Journal of Nonlinear Science 31 (2021) 093122. doi:10.1063/5.0060697.

[16] J. Dormand, P. Prince, A family of embedded runge-kutta formulae, Journal of Computational and Applied Mathematics 6 (1980) 19–26. URL: https://www.sciencedirect.com/science/article/pii/0771050X80900133. doi:https://doi.org/10.1016/0771-050X(80)90013-3.

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

[18] G. Van Rossum, F. L. Drake, Python 3 Reference Manual, CreateSpace, Scotts Valley, CA, 2009.

[19] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. Del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, T. E. Oliphant, Array programming with NumPy, Nature 585 (2020) 357–362.

[20] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, SciPy 1.0: fundamental algorithms for scientific computing in python, Nat. Methods 17 (2020) 261–272.

[21] W. McKinney, et al., Data structures for statistical computing in python, in: Proceedings of the 9th Python in Science Conference, volume 445, Austin, TX, 2010, pp. 51–56.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, Journal of machine learning research 12 (2011) 2825–2830.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[24] M. Poli, S. Massaroli, A. Yamashita, H. Asama, J. Park, S. Ermon, Torchdyn: Implicit models and neural numerical methods in pytorch (????).

[25] J. D. Hunter, Matplotlib: A 2d graphics environment, Computing in science & engineering 9 (2007) 90–95.