Systems & Control Transactions

**Research Article - Peer Reviewed Conference Proceeding**
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

# Physics-Informed Graph Neural Networks for Modeling Spatially Distributed Dynamically Operated Processes

**Md Meraj Khalid[a,b], Luisa Peterson[a], Edgar Ivan Sanchez Medina[a,*], and Kai Sundmacher[a,b,*]**

[a] Process Systems Engineering, Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstraße 1, 39106 Magdeburg, Germany
[b] Chair for Process Systems Engineering, Otto-von-Guericke University, Universitätsplatz 2, 39106 Magdeburg, Germany
* Corresponding Authors: sanchez@mpi-magdeburg.mpg.de , sundmacher@mpi-magdeburg.mpg.de.

## ABSTRACT

Modeling process systems by use of partial differential equations is often complex and computationally expensive, especially for inverse problems such as optimization, state identification, or parameter estimation. Data-driven methods typically provide efficient alternatives with lower computational cost. One such method is Graph Neural Networks (GNNs), which can be used to model dynamical systems as graphs. However, dynamic GNNs often face challenges with extrapolation and representability. Integrating mechanistic insights in surrogate models can improve both prediction accuracy and interpretability. This study compares different strategies for embedding physics-based insights into GNNs to model the dynamic behavior of a catalytic $CO_2$ methanation reactor. The hybrid integration of physics-informed GNNs aims to improve the predictive ability and interpretability while reducing the model development time, thereby facilitating faster deployment. Results show that penalizing the predicted change in reactor states from one time step to the next during training reduces the Frobenius error norm of GNN predictions from about 2.2 % to 1.4 %. In contrast, using a traditional physics-based loss function during training results in a Frobenius norm as high as 8.7 %. This highlights the importance of comparing different strategies for embedding physical laws into data-driven models.

**Keywords**: Hybrid modeling, Scientific Machine Learning, Graph Neural Networks, $CO_2$ Methanation

## INTRODUCTION

Process systems are often spatially distributed, dynamic, and exhibit highly nonlinear behavior. Modeling these systems requires the abstraction of process knowledge into a mathematical representation.

Accurate mechanistic models, often represented by mass and energy balances, combined with thermodynamic and kinetic laws, and abstracted as a system of partial differential equations (PDEs), require significant effort in terms of development time. The expertise and deep understanding required for this type of modeling adds to its complexity. Due to the intricate nature of process systems, analytical solutions are rarely achievable, necessitating the use of numerical methods. These methods rely on discretization schemes, in terms of temporal and spatial coordinates of the system. Furthermore, the complexity of problems arising from the numerical treatment increases with finer discretization, especially in inverse applications such as optimization, state identification, or parameter estimation. Despite these challenges, mechanistic models offer significant advantages. Their parametric nature allows sensitivity and scenario analysis, as well as critical flexibility analysis for dynamically operated process systems.

Recent developments in machine learning (ML), especially deep learning (DL) methods, have facilitated surrogate modeling of complex systems. Data-driven models are relatively easy to develop and take less time to build, provided sufficient data is available in both quantity and quality. However, they do require tuning of hyperparameters, which, while not trivial, is generally less time consuming than building mechanistic models from scratch. However, data-driven surrogate models face challenges related to interpretability and the ability to make meaningful extrapolative predictions.

Hybrid methods that integrate ML with knowledge-based models are often referred to as scientific machine

learning (SciML). SciML models leverage the strengths of both mechanistic and data-driven approaches, integrating domain expertise with data-driven insights. They contain embedded physical laws, can be used for extrapolative predictions, and generally provide better interpretability [1, 2].

The principles of modeling of spatially distributed dynamical systems have practical applications in the design and optimization of energy conversion technologies capable of storing renewable energy. Renewable energy plays a critical role in addressing climate change and increasing energy independence, but its intermittent nature presents significant challenges. To overcome these challenges, renewable energy can be converted into other forms, such as chemical energy, to facilitate its long-term storage, transportation, and distribution. One such conversion process is methanation, in which carbon dioxide and hydrogen are converted to methane, with water as a byproduct. The hydrogen, often referred to as green hydrogen, is typically produced by electrolysis of water using renewable energy [3]. This process allows methane to be used as an efficient chemical energy carrier that can be easily stored and transported within the existing infrastructure. The methanation reaction is as follows:

$$\mathrm{CO_2} + 4\,\mathrm{H_2} \rightleftharpoons \mathrm{CH_4} + 2\,\mathrm{H_2O} \tag{1}$$

$CO_2$ methanation is assumed to take place in a one-dimensional polytropic reactor which can be mechanistically described by the mass and energy balances [4]:

$$\frac{\partial X}{\partial t} = -\frac{\mathrm{u}}{\epsilon_\mathrm{R}}\frac{\partial X}{\partial z} + \frac{M_{\mathrm{CO_2}}}{\epsilon_\mathrm{R}\,\rho\,y_{\mathrm{CO_2,in}}}(1-\epsilon_\mathrm{R})\sigma_\mathrm{eff}, \tag{2}$$

$$\frac{\partial T}{\partial t} = -\frac{u_\mathrm{in}\rho_\mathrm{in}C_p}{(\rho C_p)_\mathrm{eff}}\frac{\partial T}{\partial z} + \frac{1}{(\rho C_p)_\mathrm{eff}}\frac{\partial}{\partial z}\left[\Lambda_\mathrm{ax}\frac{\partial T}{\partial z}\right] + \frac{4U}{D\,(\rho C_\mathrm{p})_\mathrm{eff}}(T-T_\mathrm{cool}) - \frac{\Delta H_r}{(\rho C_p)_\mathrm{eff}}(1-\epsilon_\mathrm{R})\sigma_\mathrm{eff}, \tag{3}$$

with the initial and boundary conditions:

$$X|_{z=0} = 0, \quad X|_{t=0} = X_0, \quad T|_{t=0} = T_0, \quad \Lambda_\mathrm{ax}\frac{\partial T}{\partial z} = u_\mathrm{in}\rho_\mathrm{in}C_\mathrm{p}(T-T_\mathrm{cool}), \quad \left.\frac{\partial^2 T}{\partial z^2}\right|_{z=L} = 0. \tag{4}$$

Hereby, $X$ and $T$ denote carbon dioxide conversion and reactor temperature, respectively. $z$ stands for the spatial (axial) and $t$ for the time coordinate. The constants $\epsilon_\mathrm{R}$, $M_{\mathrm{CO_2}}$, $y_{\mathrm{CO_2,in}}$, $u_\mathrm{in}$, $\rho_\mathrm{in}$, $D$ and $L$ describe the specific reactor dimensions and settings. The variables $u$, $\rho$, $\sigma_\mathrm{eff}$, $C_\mathrm{p}$, $\Lambda_\mathrm{ax}$, $U$, $T_\mathrm{cool}$ and $\Delta H_r$ are modeled by algebraic equations.

The initial conditions $X_0$ and $T_0$ are set for a load change scenario where the hydrogen depends on the time-varying renewable energy generation. The synthetic data used in this work, visualized in Figure 1, is generated using the mechanistic model reported above, with the parameter tuned as reported in [5].

This work extends the graph neural network (GNN) study presented in [5] by optimizing the hyperparameters, modifying the network architecture, and exploring different strategies for embedding physical information into the loss function. Three different models are investigated: (i) a baseline GNN, (ii) a GNN incorporating the time derivatives of the data into the loss function, and (iii) a GNN incorporating the underlying physics described by Equations 2 and 3 into the loss function.

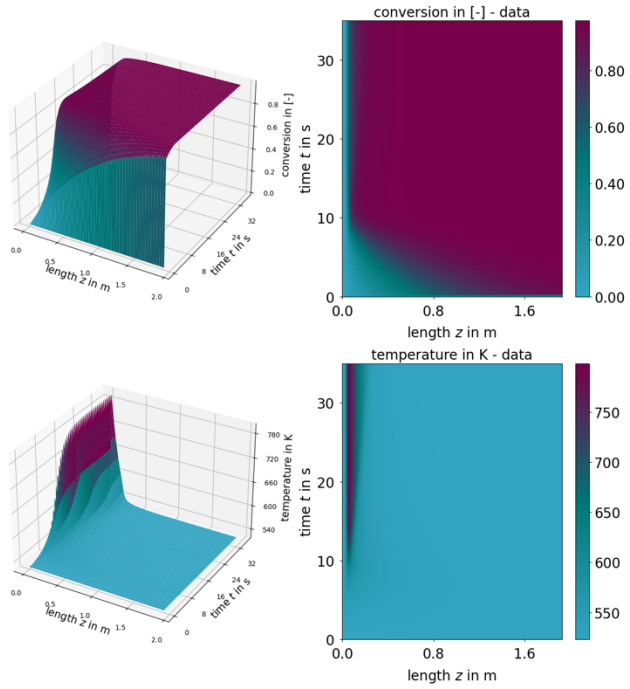


**Figure 1.** Conversion and temperature profiles generated from the mechanistic model and used to train and test GNNs.

# METHODS

## Graph Neural Networks

GNNs, introduced in [6], are a class of DL methods that operate on graphs. GNNs require the system to be represented as a graph '$\mathcal{G}$' with a set of nodes '$V$' and their inter-connectivities as edges '$E$'. A graph is then defined as the tuple of its node attributes and edge attributes along with the information of connectivity $\mathcal{G} = (V, E)$. The individual nodes and edges are denoted as $\mathrm{v}_i \in V$ for the $i^{th}$ node and $\mathrm{e}_{ij} \in E$ for the edge connecting the $i^{th}$ and $j^{th}$ node. The learning in GNNs is based on a message passing strategy where each target node receives the embeddings of all the neighbouring nodes and edges. These embeddings are then aggregated using a permutation invariant operator. Afterwards, this aggregated message is combined with the target node's own embedding using an update function, usually non-linear. This

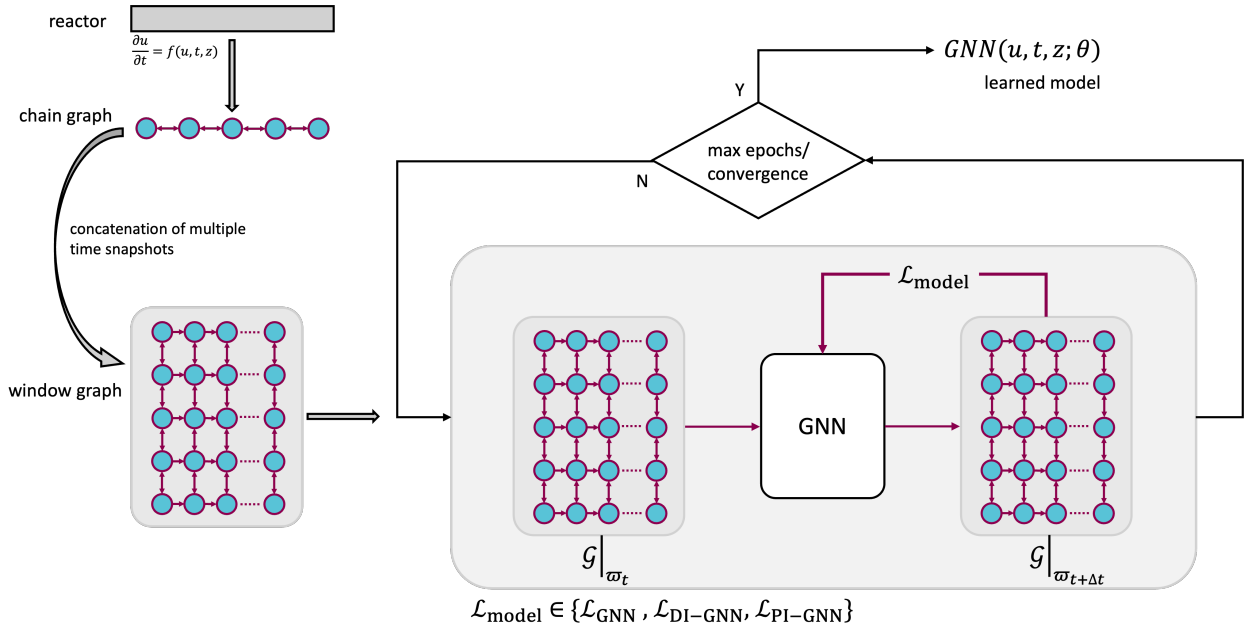


**Figure 2:** Generic schematic diagram of hybrid Graph Neural Networks.

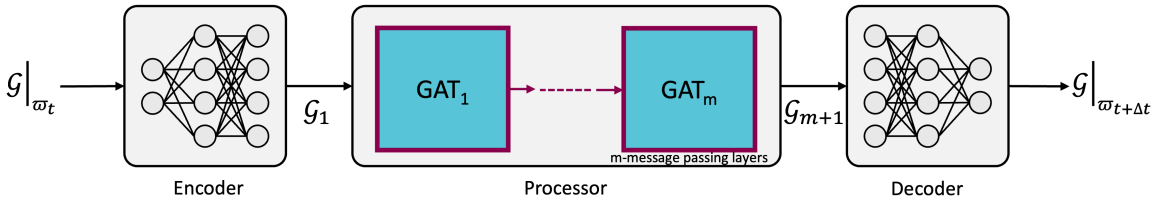


**Figure 3:** Network architecture of Graph Neural Networks.

mechanism, mathematically represented in Equation 5, makes GNNs spatially aware models

$$h_i^{k+1} = \phi\left(h_i^k, \zeta^k(\{h_j^k, \forall j \in \mathcal{N}(i)\})\right). \quad (5)$$

Here, $h$ represents the node embeddings, $\phi$ is the update function, $\zeta$ is the aggregate function, $i$ and $j$ denote the target node $i$ with nodes $j$ in its neighborhood $\mathcal{N}$, $k$ denotes the $k^{th}$ message passing layer.

For spatio-temporal systems, GNNs require the integration of a temporal model to effectively capture their dynamic behavior, leading to the development of spatio-temporal GNNs. Graph attention networks (GATs) [7] are one of these methods, where the message passing equation is modified using an attention coefficient resembling cognitive attention. The mathematical form is given in Equation 6

$$h_i^{k+1} = \phi\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\, \mathcal{W}^k\, h_i^k\right). \quad (6)$$

Here, $\mathcal{W}^k$ stands for the weights matrix at layer $k$ and $\alpha_{ij}$ is the attention coefficient for the connection between node $i$ with its neighboring nodes $j$. The integration of other temporal models, like recurrent neural networks, with GNNs has also been explored in the context of dynamical systems modeling [8].

## Graph representation of dynamical systems

Spatial dependence in systems can be represented structurally using grids for evenly spaced data or meshes for more general arrangements. Graph based models for learning complex physics simulations have been reported in [9-11]. In this work, we represent the dynamical system as a graph, where grid or mesh points are translated into nodes, and their connections are represented by edges. In the case of a methanation reactor, since the system is assumed to be one-dimensional in space, it is represented as a one-dimensional grid or a chain graph, as reported in [5]. Each graph corresponds to a snapshot of the domain at a specific time point. For improved learning, multiple time snapshots or chain graphs are concatenated to form a window graph, as shown in Figure 2. Adjacent nodes at each time point are connected by bi-directional edges, while corresponding nodes in two chain graphs are connected by unidirectional edges from the past to the future. The outputs of the mechanistic model, namely conversion and temperature, are used as node features, while the distances between two

neighboring spatial points are used as edge embeddings. The data is pre-processed using min-max scaling and split sequentially before embedding.

As shown in Figure 3, an encoder-processor-decoder architecture is used for GNN processing. The window graph is passed to an encoder, essentially a multilayer perceptron (MLP), which encodes the node features to the hidden dimension of the architecture. The encoded information is then processed using multiple GAT message passing layers. Finally, a decoder model of structure akin to the encoder is used to get the learned node features back to the original dimensions. The optimized hyperparameters are the hidden dimension of the architecture (228) and the window size (number of chain graphs processed at a time) of 20.

The output from the GNN block is used to calculate the loss function,

$$\mathcal{L}_{\mathrm{GNN}}(\theta) = \sum_{m=1}^{N_t}\sum_{n=1}^{N_z} \| \overline{u}(\varpi_m, z_n) - u(\varpi_m, V_n;\ \theta) \|_2^2. \quad (7)$$

In Equation 7, $\theta$ are the learned parameters of the GNN, $\varpi_m$ is the $m^{th}$ window graph, $V_n$ is the $n^{th}$ node, $N_t$ and $N_z$ denote the total number of window graphs and nodes in each graph, respectively. A stochastic optimization method from [12] with a learning rate of $5 \times 10^{-5}$ is used for GNN-training. This workflow is used in an iterative roll-out manner to predict multiple future time levels.

## Derivative-Informed Graph Neural Networks

The framework of a simple GNN implementation is modified by incorporating the time derivatives of the node features as additional information for the GNN. The architecture is also updated by replacing the MLPs in the encoder and decoder with deep neural networks (DNNs) and including more message-passing steps to enhance the system's awareness of neighboring nodes. Simply using the time derivative information as additional node features does not improve predictions, likely because the model perceives it as additional noise. Therefore, a more robust implementation of the derivative-informed GNN (DI-GNN) was developed by modifying the loss function to incorporate the derivative loss. This approach is based on the hypothesis that, at each time step or for each window graph, although the GNN is informed of the neighboring node embeddings, it remains unaware of how the node embeddings have changed from one time step or window graph to another.

The modified loss function reads:

$$\mathcal{L}_{\mathrm{DI-GNN}}(\theta) = (\gamma_{\mathrm{d}}) \sum_{m=1}^{N_t}\sum_{n=1}^{N_z} \|\overline{u}(\varpi_m, z_n) - u(\varpi_m, V_n;\ \theta)\|_2^2 + (1-\gamma_{\mathrm{d}}) \sum_{m=1}^{N_t}\sum_{n=1}^{N_z} \left\| \frac{d\overline{u}(\varpi_m, z_n)}{dt} - \frac{du(\varpi_m, V_n;\ \theta)}{dt} \right\|_2^2, \quad (8)$$

where $\gamma_{\mathrm{d}}$ denotes the data to derivatives weight.

## Physics-Informed Graph Neural Networks

To make the model truly semi-parametric in nature, the mechanistic model was integrated with the GNN implementation, into a physics-informed GNN (PI-GNN) framework. The modified GNN architecture in DI-GNN was used and the loss function was modified to resemble a physics-informed neural network (PINN) [13]. Ideally in a PINN loss function, the total sum is built of the data loss and physics loss which includes initial conditions loss and boundary conditions loss. In our case, the initial conditions are already satisfied with the first window graph in the train split, while boundary conditions are met within the equation system itself as equations are declared in their discretized form, i.e., as sets of ordinary differential equations (ODEs).

The modified loss function reads:

$$\mathcal{L}_{\mathrm{PI-GNN}}(\theta) = (\gamma_{\mathrm{p}}) \sum_{m=1}^{N_t}\sum_{n=1}^{N_z} \|\overline{u}(\varpi_m, z_n) - u(\varpi_m, V_n;\ \theta)\|_2^2 + (1-\gamma_{\mathrm{p}}) \sum_{m=1}^{N_t}\sum_{n=1}^{N_z} \left\| \frac{d\overline{u}(\varpi_m, z_n)}{dt} - PDE_{\mathrm{sys}(u(\varpi_m, V_n;\ \theta))} \right\|_2^2, \quad (9)$$

where $\gamma_{\mathrm{p}}$ denotes the data to physics weight. $PDE_{\mathrm{sys}}$ denotes the right-hand side of the mechanistic model given by Equations 2 and 3.

## COMPARATIVE ANALYSIS

The comparison between the predictions of the three models is based on the relative Frobenius error between the synthetic data and the model predictions. The Frobenius norm is defined as the square root of the sum of the absolute squares of all matrix elements:

$$\|E\| := \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |E_{ij}|^2}. \quad (10)$$

Each model uses a different GNN architecture and loss function. To reduce computation time and avoid overfitting, we used early stopping, which stops training when validation performance stops improving. This resulted in different training epochs for each model.

Both DI-GNN and PI-GNN are evaluated for two weight factors of 0.5 and 0.7 to evaluate the effect of individual loss terms on the predictions. Table 1 lists the relative Frobenius norms for GNN, DI-GNN ($\gamma_d = \{0.5, 0.7\}$), and PI-GNN ($\gamma_{\mathrm{p}} = \{0.5, 0.7\}$). The profiles predicted by these models have been graphically visualized in Figures 5-7. Out of the simulated time of 35 seconds, the first 20 seconds were used for model training while the remaining 15 seconds were reserved for model testing/performance evaluation. This split has been marked with a horizontal black line on all the figures.

GNNs are able to capture the overall dynamics of the data, but we observe an error propagation with time. This error propagation is reduced in case of DI-GNN, which performs best amongst the three models.

The predictive capability of DI-GNN is influenced by

the data to derivative weight ($\gamma_d$). On increasing the influence of data over derivatives, the DI-GNN theoretically approaches a simple GNN implementation, but because of the presence of an additional derivative loss term, its loss value increases, resulting in poorer predictions than possible with a simple GNN.

PI-GNN performs poorly as compared to simple GNN or DI-GNN, and it is also not trivial to tune. The discretized equations contain non-linear polynomial expressions which are dependent on the discretization of the system, the node distribution in a graph, as well as the number of graphs in the complete domain. The effect of data to physics weight ($\gamma_p$) is opposite to that observed in the case of DI-GNN. One possible cause is that once the model is physics-informed, prioritizing data over physics would make the predictions worse.

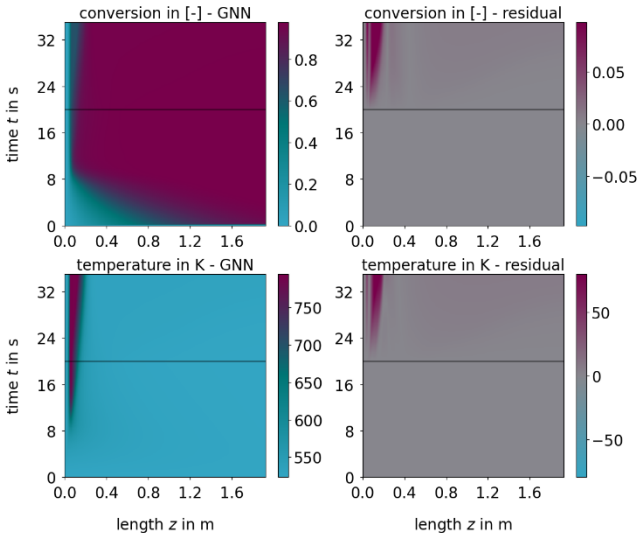


**Figure 5.** Conversion and temperature profiles predicted by the GNN and their comparison to the original data.

**Table 1:** Relative Frobenius Norms

| Method | Weight ($\gamma_-$) | $\|E\|$ for T | $\|E\|$ for X |
|---|---|---|---|
| GNN | - | 2.199 % | 2.165 % |
| DI-GNN | $\gamma_d$ = 0.5 | 1.389 % | 1.077 % |
| | $\gamma_d$ = 0.7 | 2.933 % | 3.845 % |
| PI-GNN | $\gamma_p$ = 0.5 | 8.715 % | 7.015 % |
| | $\gamma_p$ = 0.7 | 4.612 % | 6.755 % |

PI-GNN shows promising results in scenarios where we have sparse experimental data or certain boundary conditions must be explicitly enforced. They also ensure that the physical/scientific limitations are not violated (reactor temperature is below the maximum design temperature, conversion is not above 100%, etc.). But, as shown here, for the construction of surrogate models from mechanistic ones, their development can be cumbersome. The computation time/ epoch required for training of these models also varies with changing complexity ($t_{\mathrm{PI-GNN}} > t_{\mathrm{DI-GNN}} > t_{\mathrm{GNN}}$).

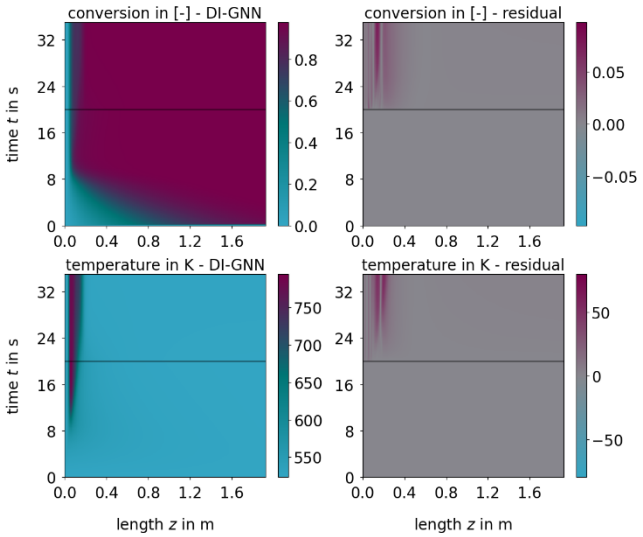


**Figure 6.** Conversion and temperature profiles predicted by the DI-GNN ($\gamma_d$ = 0.5) and their comparison to the original data.

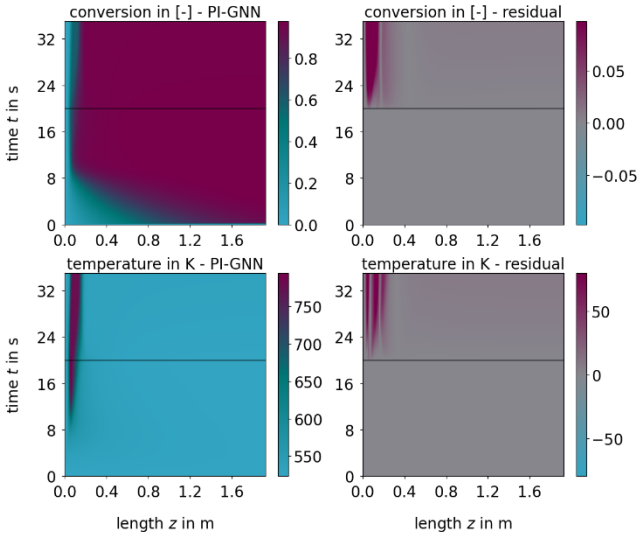


**Figure 7.** Conversion and temperature profiles predicted by the PI-GNN ($\gamma_p$ = 0.5) and their comparison to the original data.

## CONCLUSIONS

Efficient modeling of dynamical systems is critical for insightful predictions and effective decision making. To explore broader process alternatives, a faster model development and deployment pipeline is needed, which can be achieved through surrogate modeling techniques. However, the quality of a surrogate model is highly dependent on the representativeness of the available data. In our case, the training data may not fully capture the dynamics of the system, which could limit the accuracy

and generalizability of the model.

Present implementation of physics-informed models shows comparatively poor predictive capabilities but might outperform other purely data-driven models in scenarios where we have sparse experimental data and in inverse problem constructs for optimization or parametric adaptation of the model. These physically-informed models require more computing resources than a purely data-driven surrogate model. This computational burden can be alleviated by replacing the full mechanistic model with a reduced-order model, an approach particularly relevant for higher-order and multi-dimensional spatial domains.

To maximize the potential of current implementation, further work should ensure thorough hyperparameter optimization for GNN architecture as well as the loss function. The framework can also be extended and evaluated on different dynamical spatio-temporal systems, e.g. fluid mechanics, distribution networks, etc. A well-tuned, parametrized framework should be able to precisely capture the underlying dynamics and be well suited for use in optimization, uncertainty quantification and control.

## DIGITAL SUPPLEMENTARY INFORMATION

The code and data used in this study are available at the accompanying GitHub repository at https://github.com/mmkhalid-pse/hybrid-GNN.

## ACKNOWLEDGEMENTS

Md Meraj Khalid is also affiliated to the International Max Planck Research School for Advanced Methods in Process and Systems Engineering (IMPRS ProEng), Magdeburg, Germany. This work was partly funded by the Bundesministerium für Bildung und Forschung (BMBF) and Project Management Jülich (PtJ) under grant 03HY302R.

## REFERENCES

1. M. von Stosch, R. Oliveira, J. Peres, S. F. de Azevedo. Hybrid semi-parametric modeling in process systems engineering: Past, present and future. Comput. Chem. Eng. 60:86-101 (2014) https://doi.org/10.1016/j.compchemeng.2013.08.008
2. A. M. Schweidtmann, D. Zhang, M. von Stosch. A review and perspective on hybrid modeling methodologies. Digit. Chem. Eng. 100136 (2023) https://doi.org/10.1016/j.dche.2023.100136
3. K. Ghaib, F.-Z. Ben-Fares. Power-to-Methane: A state-of-the-art review. Ren. Sus. E. Reviews 81:433-446 https://doi.org/10.1016/j.rser.2017.08.004
4. R. T. Zimmermann, J. Bremer, and K. Sundmacher. Load-flexible fixed-bed reactors by multi-period design optimization. Chem. Eng. J. 428:130771 (2022) https://doi.org/10.1016/j.cej.2021.130771
5. L. Peterson, A. Forootani, E. I. Sanchez Medina, I. V. Gosea, K. Sundmacher, P. Benner. Towards digital twins for Power-to-X: Comparing surrogate models for a catalytic $CO_2$ methanation reactor. TechRxiv (2024) https://doi.org/10.36227/techrxiv.172263007.76668955/v1
6. M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. Proc. 2005 IEEE IJCNN 2: 729–734 (2005) https://doi.org/10.1109/IJCNN.2005.1555942
7. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio. Graph attention networks. ICLR 2018 (2018) https://doi.org/10.48550/arXiv.1710.10903
8. Y. Seo, M. Defferrard, P. Vandergheynst, X. Bresson. Structured sequence modeling with graph convolutional recurrent networks. arXiv (2016) https://doi.org/10.48550/arXiv.1612.07659
9. A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. W. Battaglia. Learning to simulate complex physics with graph networks. ICML 2020 (2020) https://doi.org/10.48550/arXiv.2002.09405
10. T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. W. Battaglia. Learning mesh-based simulation with graph networks. arXiv (2021) https://doi.org/10.48550/arXiv.2010.03409
11. H. Zhang, L. Jiang, X. Chu, Y. Wen, L. Li, Y. Xiao, L. Wang. Combining physics-informed graph neural network and finite difference for solving forward and inverse spatiotemporal PDEs. arXiv (2024) https://doi.org/10.48550/arXiv.2405.20000
12. D. Kingma and J. Ba. Adam: A method for stochastic optimization. ICLR 2015 (2015) https://doi.org/10.48550/arXiv.1412.6980
13. 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. 378:686-707 (2019) https://doi.org/10.1016/j.jcp.2018.10.045