PITs: Physics-Informed Transformers for Predicting Chemical Phenomena

Mayank Nagda[1](✉), Phil Ostheimer[1], Thomas Specht[2], Frank Rhein[3], Fabian Jirasek[2], Marius Kloft[1], and Sophie Fellenz[1]

[1] Department of Computer Science, RPTU Kaiserslautern-Landau, Germany
{nagda,ostheimer,kloft,fellenz}@cs.uni-kl.de
[2] Laboratory of Engineering Thermodynamics, RPTU Kaiserslautern-Landau, Germany {thomas.specht,fabian.jirasek}@rptu.de
[3] Karlsruhe Institute of Technology, Germany frank.rhein@kit.edu

Abstract. Accurate prediction of chemical phenomena is crucial for optimizing and understanding chemical processes. Physics-Informed Neural Networks (PINNs) have emerged as a promising data-driven method for predicting chemical phenomena using deep learning. However, PINNs, which are based on multilayer perceptrons (MLPs), employ point-wise predictions, overlooking the implicit dependencies within the physical system. Sophisticated architectures, such as Transformers, can model these complex dependencies, but they traditionally lack the integration of physical constraints essential for accurate predictions. To address this, we present Physics-Informed Transformers (PITs), a novel approach that combines the strengths of Transformers with the incorporation of physical constraints through Partial Differential Equations (PDEs). PITs generate task-specific input sets for the chemical data that allow for incorporating the dependencies inherent in the physical system while also incorporating physical constraints and replacing point-wise PINN loss with a set-wise loss. Our experiments demonstrate that PITs achieve superior generalization and accuracy in predicting activity coefficients and agglomerate breakage, mitigating failure modes, and converging faster compared to existing state-of-the-art approaches.

Keywords: Chemical Engineering · Physics Informed Neural Networks · Transformers

1 Introduction

Accurate prediction of chemical phenomena such as particle aggregation [7] and activity coefficients [17, 34] is crucial for understanding and optimizing chemical processes, impacting fields such as chemical engineering, pharmaceuticals, and environmental science [10, 19]. For instance, predicting activity coefficients is essential for the design and operation of separation processes [26] and for describing the properties of aqueous solution droplets relevant to atmospheric science [29]. Additionally, particle aggregation (coagulation/flocculation) is widely used in

various industries, including papermaking, mineral processing, and wastewater treatment [14].

For accurately predicting chemical phenomena, data-driven models are often preferred over simulations as data-driven models can more accurately represent chemical processes by incorporating external factors that influence predictions, unlike simulations, which are purely mathematical models [9, 10]. Therefore, having experimental data is advantageous for these models. However, obtaining experimental data is typically expensive and limited in availability, posing a challenge to the accurate prediction of chemical phenomena. For instance, even extensive chemical property databases like the Dortmund Datenbank (DDB) [2] have experimental data for the activity coefficients of only 31,000 binary systems, representing a small fraction of all possible molecular combinations [42].

Physics-Informed Neural Networks (PINNs) represent a novel approach for predicting chemical phenomena by combining data-driven methods with physical laws [31]. PINNs integrate the governing equations of chemical processes, such as PDEs, directly into the training process of neural networks. This integration ensures that the predictions not only fit the experimental data but also adhere to known physical laws, providing more reliable and interpretable results. The primary benefit of using PINNs is their ability to generalize well from limited experimental data, leveraging the underlying physics (from PDEs) to make accurate predictions even in data-scarce scenarios. This makes PINNs particularly useful for applications where obtaining experimental data is challenging or expensive [11].

Despite their advantages, current implementations of PINNs face several limitations. One major challenge is the computational cost associated with training these networks, as they require solving complex PDEs repeatedly during the learning process [18]. Additionally, the accuracy of PINNs heavily depends on the quality and completeness of the physical models they incorporate; inaccuracies or simplifications in these models can lead to erroneous predictions [46]. Furthermore, PINNs may struggle with highly nonlinear or chaotic systems where small errors in the physical model or data can propagate and amplify, reducing the reliability of predictions [4]. Addressing these limitations is crucial to enhance the efficiency, robustness, and applicability of PINNs in diverse chemical engineering contexts.

Existing PINNs are usually based on Multi-Layer Perceptrons (MLPs) and only offer point-wise predictions [31]. This leads to the limitations previously mentioned, as MLPs can only process limited inputs at once and have proven to be less robust on complex high-frequency PDEs, leading to failure modes. This limitation is addressed in the literature by PINNsFormer [44], which is a Transformer-based architecture[36]. As Transformers are seq-to-seq networks, the authors propose creating pseudo-sequences based on the temporal index of the domain. Although PINNsFormer observed significant success over traditional MLP-based PINNs in overcoming these limitations, PINNsFormer, by default, is not generalized for the chemical tasks, where domain can include temperature and pressure and is independent of time.

To overcome the limitations of MLP-based PINNs and the lack of generalizability of PINNsFormer, we propose Physics-Informed Transformers (PITs). In this approach, we suggest a set generation process specifically designed for the chemical domain, creating task-specific sets for predicting chemical phenomena. We implement PITs for predicting activity coefficients and agglomerate breakage. Our experiments show that PITs outperform existing state-of-the-art MLP-based PINN architectures in predicting chemical phenomena.

1.1 Main Contributions

This work presents a significant advancement in predicting chemical phenomena by introducing Physics-Informed Transformers (PITs), a novel approach that combines the robust set modeling capabilities of Transformers with the physical constraints provided by PDEs. Our contributions can be summarized as follows:

- **Integration of Physical Constraints with Transformers**: We bridge the gap between data-driven modeling and physical law incorporation by integrating PDEs directly into the Transformer architecture for chemical domain. This ensures that the predictions adhere to known physical laws, enhancing the reliability and interpretability of the results.
- **Benchmark state-of-the-art methods**: Along with PITs, we benchmark existing PINN methods on predicting chemical phenomena. The benchmarking process involves a comprehensive evaluation of various PINN approaches, assessing their performance across multiple datasets.
- **Performance in Predicting Chemical Phenomena**: Our experiments demonstrate that PITs significantly outperform existing state-of-the-art PINNs in predicting chemical phenomena, such as activity coefficients and agglomerate breakage. PITs show superior generalization and accuracy, mitigate failure modes, and achieve faster convergence.

2 Related Work

In this section we discuss related work in physics-informed neural networks and prediction of chemical phenomena.

2.1 Physics-informed neural networks (PINNs)

Methods to solve partial differential equations (PDEs) have existed for some time [22, 27], but the recent advancements in deep learning have revitalized this concept, leading to the development of PINNs [31]. PINNs represent a novel approach to solving PDEs using neural networks by embedding the PDE structure directly into the loss function. These networks introduce a residual term to the loss function that penalizes predictions not conforming to the underlying PDE, and have led to extensive research and applications. PINNs have been extended with successful applications in diverse fields such as simulating blood flow in cardiovascular structures [32], climate forecasting [38], and fluid mechanics [6].

PINNs primarily follow MLP-based architecture. However, MLP-based PINN architectures often struggle with stable training and accurate predictions. Wang et al. [39] identified that multi-scale interactions within the PINN loss function cause gradient flow stiffness, necessitating stringent stability requirements on the learning rate. Other research has pointed to inherent failure modes, especially with high-frequency or multiscale PDEs, where predictions tend to collapse into overly smooth, trivial solutions [30, 13, 21, 41, 44, 24].

To address these challenges, researchers have explored several strategies, including different training schemes, data interpolation techniques, new model architectures, and incorporating domain-specific dependencies. Training schemes can be computationally expensive, as exemplified by Krishnapriyan et al.'s [21] seq-to-seq approach, which requires sequential training of multiple neural networks. Data interpolation methods [15, 25, 40, 41], while useful, often rely on simulations or real-world scenarios, which can be difficult to obtain. In terms of alternative architectures, Bu and Karpatne [5] proposed Quadratic Residual Networks (QRes), introducing quadratic non-linearity before applying activation functions. Wong et al. [43] advocated for learning in sinusoidal spaces with PINNs using the First-Layer Sine (FLS) method. Wang et al. [41] integrated the Neural Tangent Kernel (NTK) with PINNs, constructing scalable kernels, though scalability issues remain as sample size or model parameters increase. Recently, Zhao et al. [44] introduced PINNsFormer, which accounts for implicit temporal dependencies and outperforms existing methods. However, PINNsFormer is limited to temporal domains and does not provide mechanism to generalize on chemical domains. Therefore, in our experiments, we compare PITs against PINNs, FLS, and QRes.

2.2 Predicting Chemical Phenomena

PINNs have become increasingly popular in chemical engineering and computational physics for their ability to integrate physical laws directly into neural network training, enhancing accuracy and generalization where traditional data-driven models often fail [11]. They have been applied to optimizing chemical processes and improving reaction models, showing significant improvements in predictive accuracy for processes such as pig iron desulfurization [28]. In dynamic modeling of chemical and biotechnological processes, PINNs offer promising results [37]. In computational physics, PINNs are used to solve complex fluid dynamics problems governed by the Navier-Stokes equations, effectively predicting velocity and pressure fields in laminar flow scenarios around particles, providing a competitive alternative to traditional CFD methods [16]. PINNs have also been used in predicting activity coefficients and particle agglomeration.

Predicting Activity Coefficients. Recent advancements have introduced PINNs using the Gibbs–Duhem equation for predicting binary activity coefficients across various compositions [34, 3]. This approach embeds the Gibbs–Duhem equation directly into the neural network's loss function, utilizing automatic differentiation within standard machine learning frameworks. Unlike hybrid ML methods

that incorporate specific thermodynamic models into the neural network, this method avoids associated prediction constraints. PINNs have shown thermodynamic consistency and generalization in activity coefficient predictions. Additionally, the choice of model architecture, particularly the activation function, has been found to significantly influence prediction accuracy. This methodology is also extendable to other thermodynamic consistency conditions, highlighting its versatility.

Predicting Particle Agglomeration. Particle agglomeration and breakage are common phenomena in industries such as chemical, agricultural, and pharmaceutical sectors. Recent studies have introduced PINNs to tackle both forward and inverse problems in these processes [7]. PINN approaches have shown significant potential for solving inverse problems in particle aggregation and breakage, even with noisy data. By integrating the population balance equation into the neural network's loss function, this method improves training efficiency and ensures compliance with physical laws. For forward problems, PINNs provide solutions that closely match analytical results. For inverse problems, data-driven approaches are used to discover model parameters of population balance equations. Additionally, it has been reported that the choice of different neural network structures significantly affects outcomes.

Despite their success, traditional PINNs based on MLPs often face challenges in capturing the implicit dependencies within physical systems, leading to limitations in their robustness. These limitations have led to the development of advanced architectures, such as the PINNsFormer [44], which leverages the sequence modeling capabilities of Transformers to address these issues. However, the generalizability of PINNsFormer to chemical domains remains limited, particularly where domain indices such as temperature and pressure are independent of time. Furthermore, recent advanced MLP-based PINN methods such as FLS [43] and QRes [5] have not yet been applied and benchmarked on predicting chemical phenomena.

To overcome the limitations of MLP-based PINNs and the lack of generalizability of PINNsFormer, in our approach, we suggest a set generation process specifically designed for the chemical domain, creating task-specific sets for predicting chemical phenomena leading to improved and robust prediction.

3 Preliminaries

In this section, we will start with the problem statement and hypothesis and then introduce PINNs and the Transformer architecture. Note that, for simplicity, we present preliminaries and our method by exemplifying the spatio-temporal domain. In general, as shown in our experiments, the proposed method is versatile and can be extended to specific chemical domains.

3.1 Problem statement and hypothesis

Problem statement We consider the problem of incorporating physical laws into the Transformer architecture using PDEs to predict chemical phenomena.

Hypothesis We hypothesize that by exploiting the implicit dependencies in the chemical domain using the Transformer architecture and integrating physical laws in the form of PDEs, we can accurately predict chemical phenomena, avoid failure modes, and achieve faster convergence compared to MLP-based architectures.

While MLP-based architectures perform point-wise predictions, Transformers can predict multiple points simultaneously, learning the affinities between them. By generating set inputs that incorporate physical laws, this approach leverages the efficiency and accuracy of Transformers, enhancing the overall prediction performance and robustness in modeling chemical phenomena.

3.2 Physics-informed Neural Networks (PINNs)

Let us consider nonlinear partial differential equations (PDEs) of the general form:

$$u_t + \mathcal{N}[u] = 0, \quad x \in \Omega, t \in [0, T], \tag{1}$$

where $u(t, x)$ is the latent solution of the PDE, u_t is the partial derivative of u w.r.t. t, $\mathcal{N}[\cdot]$ is a nonlinear differential operator, $\Omega \in \mathbb{R}^d$ is an open, bounded, connected spatial domain, and $[0, T]$ is the time interval.

In the PINNs approach introduced by [31], a *physics-informed neural network* $f := u_t + \mathcal{N}[u]$ is defined, where $u(t, x)$ is approximated by a fully connected neural network with parameters θ. $u(t, x)$ and $f(t, x)$ share the same parameters θ, and the inputs (t, x) are randomly sampled from the domain. They employ automatic differentiation, thus avoiding the need for discretizing the (space-time) domain and relying instead on random sampling. The weights of the neural networks are optimized during training with the following loss function:

$$\mathcal{L}_{\text{PINNs}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left\| f(t_f^i, x_f^i) \right\|^2 + \frac{1}{N_u} \sum_{i=1}^{N_u} \left\| u(t_u^i, x_u^i) - u^i \right\|^2, \tag{2}$$

where $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ denotes the initial and boundary data on $u(t, x)$, and $\{t_f^i, x_f^i\}_{i=1}^{N_f}$ denotes the collocation points for the residual $f(t, x)$. N_u and N_f denote the number of initial and boundary points, and collocation points, respectively. The differential operator $\mathcal{N}$ and other derivatives are evaluated using automatic differentiation.

3.3 Transformer Architecture

Transformers, initially proposed by [36], have become a cornerstone in sequence modeling tasks, particularly in natural language processing. The key innovation

of Transformers is the self-attention mechanism, which allows the model to weigh the importance of different elements in a sequence dynamically. This ability to capture long-range dependencies and contextual relationships makes Transformers suitable for a wide range of applications beyond NLP, including time series forecasting, image processing, and, in our case, predicting chemical phenomena.

A standard Transformer consists of an encoder and a decoder. Both comprise a multi-head self-attention mechanism that is the backbone of the architecture. The self-attention mechanism computes attention scores or affinities between all pairs of elements in the sequence, allowing the model to focus on relevant parts of the input.

Self-Attention Mechanism The core innovation of Transformers is the self-attention mechanism, which allows each element of a sequence to attend to every other element. This is mathematically defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3}$$

where $Q \in \mathbb{R}^{n \times d_k}$ is the query matrix, $K \in \mathbb{R}^{m \times d_k}$ is the key matrix, $V \in \mathbb{R}^{m \times d_k}$ is the value matrix, d_k is the dimension of the key vectors, n is the length of the query sequence, and m is the length of the key and value sequences.

Each input sequence element is projected into three vectors: a query, a key, and a value. The attention score for each pair of query and key is computed by taking the dot product of the query with all keys, dividing by $\sqrt{d_k}$ (to stabilize gradients), and applying the softmax function to obtain the weights on the values.

These components work together to process and transform input sequences, enabling the model to capture complex dependencies and generate accurate predictions. Sets can replace sequences as input to the Transformer because the Self-Attention mechanism is permutation-equivariant and supports set modeling, as demonstrated by the Set Transformers [23].

4 PITs: Physics-Informed Transformers

We introduce a novel method featuring Transformer architecture, namely Physics-Informed Transformers (PITs), which (1) exploit the implicit dependencies in a domain by learning affinities between multiple points and (2) accurately approximate solutions of multiple points in the domain simultaneously.

We illustrate our main idea in Figure 1. First, we generate sets based on the implicit dependencies in the domain. Then, we learn embeddings for each element of the set and process them using the Transformer network. Finally, we perform prediction using the Output Layer. The parameters of PITs are learned using a set-wise physics loss.

Our method consists of four main components: Set Generator, Embedding Network, Transformer, and Output Layer, which we will explain in turn. Then, we present the learning scheme.

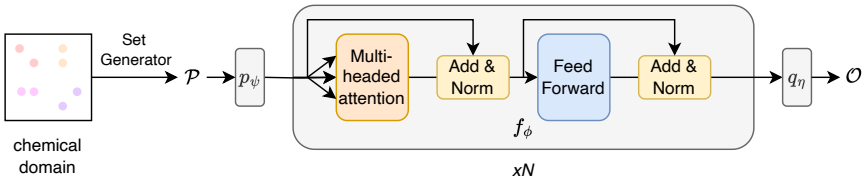

Fig. 1. Physics-Informed Transformers (PITs). An overview of the PITs architecture, illustrating the integration of Transformer networks with physical constraints for predicting chemical phenomena. The process begins with the Set Generator, which creates task-specific sets based on the domain's implicit dependencies. These sets are then processed by the Embedding Network p_ψ, transforming low-dimensional inputs into high-dimensional vectors. The Transformer Network f_ϕ captures the affinities between set elements, and the Output Layer q_η generates the final predictions. The learning scheme incorporates a set physics loss, ensuring adherence to physical laws and enhancing prediction accuracy.

4.1 Model

Set Generator. Transformers can efficiently and accurately learn affinities between elements of the input set. Therefore, we design the input set such that the elements exhibit implicit dependencies rather than being independent. In our focus on predicting activity coefficients and particle agglomeration, the set generation process differs for each task, and we discuss these specific processes in detail in the Section 5.

To generalize the set generation process: we sample N sets from the domain Ω and represent them as $\mathcal{P}$. Considering a spatio-temporal domain, the set generator can be defined as:

$$\Omega \xrightarrow{\text{set generator}} \{\mathcal{P}\}_{i=1}^{N} \quad \text{s.t.} \quad \mathcal{P}_l : \{(x_l^0, t_l^0), (x_l^1, t_l^1), \ldots, (x_l^K, t_l^K)\}, \tag{4}$$

where (x_l^i, t_l^i) is the i-th element of the l-th set of $\mathcal{P}$, and K is the number of elements in a set.

Embedding Network. An element of a set (x_l^i, t_l^i) typically contains low-dimensional information regarding its state. Since PITs focus on modeling interactions between multiple elements in the set, directly feeding low-dimensional data is insufficient for accurately learning the affinities between the elements. To address this, we propose using an Embedding Network in conjunction with the Transformer network. The Embedding Network learns a parameterized function p_ψ that mixes state variables and upscales low-dimensional state representations into high-dimensional vectors, akin to word embeddings in NLP. These vectors represent the state within a high-dimensional space. For our use cases, we opt for a simple fully-connected layer for the Embedding Network, but it can also be a more sophisticated architecture depending on the problem:

$$\mathcal{E}_l = p_\psi(\mathcal{P}_l) \quad s.t. \quad \mathcal{E}_l : \{\mathcal{E}_l^0, \mathcal{E}_l^1, \ldots, \mathcal{E}_l^K\}, \tag{5}$$

where $\mathcal{E}_l^i$ represents the embedding of the element (x_l^i, t_l^i) in the set $\mathcal{P}_l$. ψ are the learnable parameters of the Embedding Network. Notably the Embedding Network transforms $\mathcal{P}_l \in \mathbb{R}^{K\times r}$ to $\mathcal{E}_l \in \mathbb{R}^{K\times d}$ where $d \gg r$.

Transformer Network. For each set $\mathcal{P}_l$, we now seek an approximate corresponding solution set u_l that satisfies the PDE in the domain Ω. Using the Set Generator and Embedding Network, we have generated sets $\mathcal{P}$ and upscaled them to $\mathcal{E}$, which is a set of high-dimensional vectors representing each element (x_l^i, t_l^i). We seek to learn a solution u_l^i for each element $\mathcal{E}_l^i$ of the set $\mathcal{E}_l$. To achieve this, we use the Transformer network to process $\mathcal{E}_l$ as it operates as an efficient set learning network. Transformers are originally used in NLP tasks, but can be applied to any set learning task. In our case, Transformers can efficiently and accurately capture affinities among elements in the input set:

$$\mathcal{O}_l = f_\phi(\mathcal{E}_l) \quad s.t. \quad \mathcal{O}_l : \{\mathcal{O}_l^0, \mathcal{O}_l^1, \ldots, \mathcal{O}_l^K\}, \tag{6}$$

where $\mathcal{O}_l^i$ represents the transformed representation of each element $\mathcal{E}_l^i$ in set $\mathcal{E}_l$. ϕ are the learnable parameters of the Transformer Network. Notably the Transformer Network transforms $\mathcal{E}_l \in \mathbb{R}^{K\times d}$ to $\mathcal{O}_l \in \mathbb{R}^{K\times d}$.

Output Layer. After processing $\mathcal{E}_l$ using the Transformer Network we get a transformed representation $\mathcal{O}_l$ where each element now has interacted and has information regarding other elements in the set $\mathcal{P}_l$. We now predict solution u_l using an Output Layer:

$$u_l = q_\eta(\mathcal{O}_l) \quad s.t. \quad u_l : \{u_l^0, u_l^1, \ldots, u_l^K\}, \tag{7}$$

where u_l^i represents the solution of each element $\mathcal{O}_l^i$ in set $\mathcal{O}_l$. η are the learnable parameters of Output Layer. Notably the Output Layer transforms $\mathcal{O}_l \in \mathbb{R}^{K\times d}$ to $u_l \in \mathbb{R}^{K\times 1}$.

4.2 Learning Scheme

Traditional PINN methodologies concentrate on point-wise predictions and employ a point-wise PINN loss, as defined in Eq. 2. Given that the proposed PITs approach involves set predictions, we have modified the conventional PINN loss to accommodate sets. In PITs, each set $\mathcal{P}_l$ corresponds to a predicted solution u_l. This allows for the calculation of the n-th order gradient with respect to the state variables x and t. For example, for a given set $\mathcal{P}_l$ and its solution u_l, the first-order gradients are given by $\frac{\partial u_l}{\partial x_l} = \left\{\frac{\partial u_l^i}{\partial x_l^i}\right\}_{i=0}^K$ and $\frac{\partial u_l}{\partial t_l} = \left\{\frac{\partial u_l^i}{\partial t_l^i}\right\}_{i=0}^K$.

This method of calculating gradients for predicted sets relative to input sets can be extended to higher-order derivatives and is applicable to residual, boundary, and initial point sets.

Consequently, we adapt the point-wise PINN objective to the set-wise PITs objective as follows:

$$\mathcal{L}_{\text{PITs}} = \frac{\lambda_f}{N_f} \sum_{e=1}^{|N|} \sum_{i=1}^{K} \left\| f_\theta(t_e^i, x_e^i) \right\|^2 + \frac{\lambda_u}{N_u} \sum_{e=1}^{|M|} \sum_{i=1}^{K} \left\| u_\theta(t_e^i, x_e^i) - u_e^i \right\|^2, \quad (8)$$

where N and M are the sets derived from residual, boundary, and initial data points, respectively, and K is the number of elements in set e. The parameters λ_f and λ_u are weighting factors, and $\theta := \{\psi, \phi, \eta\}$. N_f and N_u represent all points in N and M, respectively.

5 Experiments

In this section, we empirically demonstrate that PITs are robust and accurate across multiple tasks of predicting chemical phenomena. First, we briefly describe the comparison models in Section 5.1. Then, we benchmark all the models on predicting activity coefficients and particle agglomeration in Sections 5.2 and 5.3, respectively.

5.1 Model setup

For our baseline models, we selected the standard MLP-based PINNs [31], First-Layer Sine (FLS) [43], and Quadratic Residual Networks (QRes) [5], which represent the current state-of-the-art. To ensure fairness, we maintained approximately the same number of parameters across all baseline models. Our proposed PITs model utilizes a straightforward Transformer architecture. For training, we adhered to standard practices, initially using the Adam optimizer [20] followed by L-BFGS [35]. We evaluated our models using the standard metrics: relative Mean Absolute Error (rMAE) and relative Root Mean Squared Error (rRMSE). PITs were trained according to the objective specified in Eq. 8 with $\lambda_f = \lambda_u = 1$. Additional details on model architectures and hyperparameter selection can be found in Appendix A.1.

5.2 Predicting Activity Coefficients

Activity coefficients are crucial for modeling and simulating reaction and separation processes, as they reflect the behavior of molecular components in mixtures. Influenced by molecular structure, temperature, and concentration, these coefficients are challenging and expensive to measure, resulting in limited experimental data. State-of-the-art physical prediction models like UNIFAC [12], which comply with thermodynamic consistency criteria, are preferred over machine learning models [17]. More details on the equations are provided in Appendix B.1.

In our experiments, we predict activity coefficients using experimental data while enforcing the Gibbs-Duhem equation constraints. Data from the Dortmund

Model	Activity Coefficient		Agglomerate Breakage	
	MAE	MSE	rMAE	rRMSE
PINNs	0.175 ± 0.00	0.158 ± 0.00	0.587 ± 0.01	0.451 ± 0.00
QRES	0.188 ± 0.00	0.169 ± 0.00	0.657 ± 0.08	0.512 ± 0.03
FLS	0.176 ± 0.00	0.152 ± 0.00	0.589 ± 0.00	0.451 ± 0.00
UNIFAC	0.156	0.166	-	-
PITs (ours)	$\mathbf{0.137 \pm 0.00}$	$\mathbf{0.090 \pm 0.00}$	$\mathbf{0.318 \pm 0.02}$	$\mathbf{0.220 \pm 0.01}$

Table 1. Prediction of activity coefficients using experimental data as well as physical constraints and agglomerate breakage. PITs significantly outperform other baselines. Significant results (p-value < 0.05) are highlighted in bold, and the variance is captured over ten runs.

Data Bank (DDB) [2] includes direct and calculated activity coefficients. We excluded low-quality data and only used components with retrievable SMILES strings, converting them to canonical SMILES with RDKit [1]. We conducted a system-wise train-test split, using 10% of systems for testing. RDKit generated molecular descriptors using a count-based Morgan fingerprint (radius 0, bit size 128). For set generation, we represent each mixture as a set with elements as tuples of SMILES, temperature, and proportion.

We compare the performance of PITs with physical models (UNIFAC) and PINN baselines: PINNs, QRES, and FLS. PITs consistently outperformed the other methods, including the physical UNIFAC model, by achieving the lowest error rates. These results highlight the effectiveness of PITs in modeling complex thermodynamic properties and offer a promising approach for enhancing the precision and reliability of chemical process simulations.

5.3 Predicting Particle Agglomerations

In this study, we use PINNs to address the forward problem of agglomerate breakage. By embedding the governing population balance equation directly into the loss function, we ensured that the network adheres to physical constraints. Details about the underlying equations are provided in Appendix B.2. For set generation, we sample four neighboring points from the domain and represent them as one set. We benchmark the proposed PITs model against PINN baselines: PINNs, QRES, and FLS. The results, shown in Table 1, indicate that PITs achieve significantly lower error rates compared to the baseline models. This superior performance demonstrates the robustness and accuracy of PITs in capturing the complex dynamics involved in particle aggregation processes, making them a reliable tool for practical applications.

6 Discussions

Our experiments demonstrate the robustness and precision of PITs in predicting chemical phenomena. PITs consistently achieve lower error rates compared to

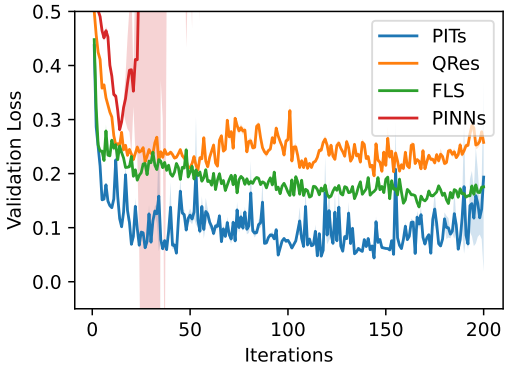

Fig. 2. PITs achieve faster convergence as opposed to other PINN approaches. Plotted is the validation loss vs. the number of training iterations.

other state-of-the-art models. The performance improvements of PITs can be attributed to their capability to model dependencies between different points within the physical domain. Unlike traditional PINNs that rely on point-wise predictions, PITs utilize a set-based approach to capture implicit dependencies in the domain.

A significant challenge in traditional PINNs is their tendency to converge to overly smooth and trivial solutions. As illustrated in Figure 2, PINNs have the highest error rates peaking in the plot. PITs address this issue by incorporating dependencies between different points within a set, effectively capturing the local interactions and variations that are critical for accurately modeling complex physical phenomena. By processing these sets using an attention mechanism, PITs can focus on relevant features and relationships, avoiding the pitfall of trivial solutions. This is also evident in Figure 2, which shows that PITs achieve faster convergence and achieve lowest error rates as opposed to baseline methods.

By effectively incorporating physical constraints and leveraging set-based learning, PITs can model intricate behaviors with high accuracy and are robust against failure modes across a diverse range of physical systems, making them a valuable tool for industrial and scientific applications.

7 Conclusion

In this work, we introduced Physics-Informed Transformers (PITs), a novel approach that leverages the robust sequence modeling capabilities of Transformers and integrates them with physical constraints through PDEs. This novel method addresses several limitations of existing PINN architectures, particularly those based on Multi-Layer Perceptrons (MLPs), by providing a more comprehensive and accurate prediction model for chemical phenomena. The integration of physical constraints with advanced Transformer architectures represents a substantial

step forward in the field of chemical process modeling. Our approach not only ensures that predictions adhere to known physical laws but also improves the interpretability and robustness of the results. The successful benchmarking against existing methods further validates the efficacy of PITs in real-world scenarios.

Our study demonstrated that PITs significantly outperform current state-of-the-art PINN methods in predicting activity coefficients and particle agglomeration. By generating task-specific pseudo-sequences, PITs effectively incorporate the dependencies inherent in physical systems, leading to superior generalization, accuracy, and faster convergence. These advancements mitigate common failure modes and enhance the reliability of predictions in chemical engineering applications. In summary, PITs offer a promising solution for accurately predicting complex chemical phenomena, providing significant improvements over traditional methods. This work paves the way for future research and applications in chemical engineering, pharmaceuticals, environmental science, and other related fields, where precise modeling and understanding of chemical processes are crucial.

Acknowledgements

The authors gratefully acknowledge financial support by Carl Zeiss Foundation in the frame of the project 'Process Engineering 4.0'. Furthermore, FJ gratefully acknowledges financial support by DFG in the frame of the Emmy-Noether program (JI 401/5-1). Part of this work was conducted within the DFG research unit FOR 5359 (BU 4042/2-1, KL 2698/6-1, BO 2538/6-1, and KL 2698/7-1). The authors acknowledge support by the DFG awards BU 4042/1-1, KL 2698/2-1, and KL 2698/5-1.

References

1. Rdkit: Open-source cheminformatics. (Last accessed: 04.04.2024), https://www.rdkit.org
2. Dortmund data bank (2023), https://www.ddbst.com/
3. Babaei, M.R., Stone, R., Iv, T.A.K., Hedengren, J.: Physics-informed neural networks with group contribution methods. Journal of Chemical Theory and Computation **19**(13), 4163–4171 (2023)
4. Bajaj, C., McLennan, L., Andeen, T., Roy, A.: Recipes for when physics fails: recovering robust learning of physics informed neural networks. Machine Learning: Science and Technology **4**(1), 015013 (2023)
5. Bu, J., Karpatne, A.: Quadratic residual networks: A new class of neural networks for solving forward and inverse problems in physics involving pdes. In: Proceedings of the 2021 SIAM International Conference on Data Mining (SDM). pp. 675–683. SIAM (2021)
6. Cai, S., Mao, Z., Wang, Z., Yin, M., Karniadakis, G.E.: Physics-informed neural networks (pinns) for fluid mechanics: A review. Acta Mechanica Sinica **37**(12), 1727–1738 (2021)

7. Chen, X., Wang, L.G., Meng, F., Luo, Z.H.: Physics-informed deep learning for modelling particle aggregation and breakage processes. Chemical Engineering Journal **426**, 131220 (2021)
8. Chen, X., Wang, L.G., Meng, F., Luo, Z.H.: Physics-informed deep learning for modelling particle aggregation and breakage processes. Chemical Engineering Journal **426**, 131220 (2021). https://doi.org/https://doi.org/10.1016/j.cej.2021.131220, https://www.sciencedirect.com/science/article/pii/S1385894721028011
9. Chew, A.K., Sender, M., Kaplan, Z., Chandrasekaran, A., Chief Elk, J., Browning, A.R., Kwak, H.S., Halls, M.D., Afzal, M.A.F.: Advancing material property prediction: using physics-informed machine learning models for viscosity. Journal of Cheminformatics **16**(1), 31 (2024)
10. Cova, T.F., Pais, A.A.: Deep learning for deep chemistry: optimizing the prediction of chemical patterns. Frontiers in chemistry **7**, 809 (2019)
11. Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics–informed neural networks: Where we are and what's next. Journal of Scientific Computing **92**(3), 88 (2022)
12. Fredenslund, A., Jones, R.L., Prausnitz, J.M.: Group-contribution estimation of activity coefficients in nonideal liquid mixtures. AIChE Journal **21**(6), 1086–1099 (Nov 1975). https://doi.org/10.1002/aic.690210607, http://dx.doi.org/10.1002/aic.690210607
13. Fuks, O., Tchelepi, H.A.: Limitations of physics informed machine learning for nonlinear two-phase transport in porous media. Journal of Machine Learning for Modeling and Computing **1**(1) (2020)
14. Gregory, J.: Monitoring particle aggregation processes. Advances in colloid and interface science **147**, 109–123 (2009)
15. Han, J., Jentzen, A., E, W.: Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences **115**(34), 8505–8510 (2018)
16. Hu, B., McDaniel, D.: Applying physics-informed neural networks to solve navier–stokes equations for laminar flow around a particle. Mathematical and Computational Applications **28**(5), 102 (2023)
17. Jirasek, F., Bamler, R., Fellenz, S., Bortz, M., Kloft, M., Mandt, S., Hasse, H.: Making thermodynamic models of mixtures predictive by machine learning: matrix completion of pair interactions. Chemical Science **13**(17), 4854–4862 (2022)
18. Kashefi, A., Mukerji, T.: Physics-informed pointnet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. Journal of Computational Physics **468**, 111510 (2022)
19. Kim, J., Gu, G.H., Noh, J., Kim, S., Gim, S., Choi, J., Jung, Y.: Predicting potentially hazardous chemical reactions using an explainable neural network. Chemical Science **12**(33), 11028–11037 (2021)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., Mahoney, M.W.: Characterizing possible failure modes in physics-informed neural networks. Advances in Neural Information Processing Systems **34**, 26548–26560 (2021)
22. Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE transactions on neural networks **9**(5), 987–1000 (1998)

23. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: International conference on machine learning. pp. 3744–3753. PMLR (2019)
24. Leiteritz, R., Pflüger, D.: How to avoid trivial solutions in physics-informed neural networks. arXiv preprint arXiv:2112.05620 (2021)
25. Lou, Q., Meng, X., Karniadakis, G.E.: Physics-informed neural networks for solving forward and inverse flow problems via the boltzmann-bgk formulation. Journal of Computational Physics **447**, 110676 (2021)
26. Pierotti, G., Deal, C., Derr, E.: Activity coefficients and molecular structure. Industrial & Engineering Chemistry **51**(1), 95–102 (1959)
27. Psichogios, D.C., Ungar, L.H.: A hybrid neural network-first principles approach to process modeling. AIChE Journal **38**(10), 1499–1511 (1992)
28. Pylypenko, A., Demeter, P., Bul'ko, B., Hubatka, S., Fogaraš, L., Legemza, J., Demeter, J.: Optimizing pig iron desulfurization using physics-informed neural networks (pinns). Engineering Proceedings **64**(1), 3 (2024)
29. Raatikainen, T., Laaksonen, A.: Application of several activity coefficient models to water-organic-electrolyte aerosols of atmospheric interest. Atmospheric Chemistry and Physics **5**(9), 2475–2495 (2005)
30. Raissi, M.: Deep hidden physics models: Deep learning of nonlinear partial differential equations. Journal of Machine Learning Research **19**(25), 1–24 (2018)
31. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics **378**, 686–707 (2019)
32. Raissi, M., Yazdani, A., Karniadakis, G.E.: Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science **367**(6481), 1026–1030 (2020)
33. Ramkrishna, D., Singh, M.R.: Population balance modeling: Current status and future prospects. Annual Review of Chemical and Biomolecular Engineering **5**, 123–146 (2014). https://doi.org/10.1146/annurev-chembioeng-060713-040241
34. Rittig, J.G., Felton, K.C., Lapkin, A.A., Mitsos, A.: Gibbs–duhem-informed neural networks for binary activity coefficient prediction. Digital Discovery **2**(6), 1752–1767 (2023)
35. Schmidt, M.: minfunc: unconstrained differentiable multivariate optimization in matlab. Software available at http://www. cs. ubc. ca/˜ schmidtm/Software/minFunc. htm (2005)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
37. Velioglu, M., Mitsos, A., Dahmen, M.: Physics-informed neural networks (pinns) for modeling dynamic processes based on limited physical knowledge and data. In: 2023 AIChE Annual Meeting. AIChE (2023)
38. Verma, Y., Heinonen, M., Garg, V.: ClimODE: Climate forecasting with physics-informed neural ODEs. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=xuY33XhEGR
39. Wang, S., Teng, Y., Perdikaris, P.: Understanding and Mitigating Gradient Flow Pathologies in Physics-Informed Neural Networks. SIAM Journal on Scientific Computing (Sep 2021). https://doi.org/10.1137/20M1318043, https://epubs.siam.org/doi/10.1137/20M1318043, publisher: Society for Industrial and Applied Mathematics

40. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient flow pathologies in physics-informed neural networks. SIAM Journal on Scientific Computing **43**(5), A3055–A3081 (2021)
41. Wang, S., Yu, X., Perdikaris, P.: When and why pinns fail to train: A neural tangent kernel perspective. Journal of Computational Physics **449**, 110768 (2022)
42. Winter, B., Winter, C., Schilling, J., Bardow, A.: A smile is all you need: predicting limiting activity coefficients from smiles with natural language processing. Digital Discovery **1**(6), 859–869 (2022)
43. Wong, J.C., Ooi, C.C., Gupta, A., Ong, Y.S.: Learning in sinusoidal spaces with physics-informed neural networks. IEEE Transactions on Artificial Intelligence **5**(3), 985–1000 (2022)
44. Zhao, Z., Ding, X., Prakash, B.A.: PINNsformer: A transformer-based framework for physics-informed neural networks. In: The Twelfth International Conference on Learning Representations (2024), https://openreview.net/forum?id=DO2WFXU1Be
45. Ziff, R.M., McGrady, E.D.: The kinetics of cluster fragmentation and depolymerisation. Journal of Physics A: Mathematical and General **18**(15), 3027 (1985). https://doi.org/10.1088/0305-4470/18/15/026, https://dx.doi.org/10.1088/0305-4470/18/15/026
46. Zou, Z., Meng, X., Karniadakis, G.E.: Correcting model misspecification in physics-informed neural networks (pinns). Journal of Computational Physics **505**, 112918 (2024)

A Model setup

A.1 Hyperparameters

Model hyperparameters. Table 2 outlines the hyperparameters for the different models evaluated in this study, including Physics-Informed Neural Networks (PINNs), Quadratic Residual Networks (QRes), First-Layer Sine (FLS), and PITs. Each model is configured with a specific number of hidden layers and hidden sizes. The PITs model also includes additional parameters such as the number of encoders and decoders, embedding size, and the number of attention heads. These configurations are crucial for defining the model architectures and their capacities to learn from the data. Table 3 shows the total parameters of all models. For a fair comparison, all models have relatively similar numbers of trainable parameters. For implementation, we follow the same implementation pipeline as of PINNsFormer[4] and use their implementation of PINNs, QRes, and FLS. The model architecture of PITs is kept standard and consistent with PINNsFormer (Although we process sets instead of psuedo sequences).

Model	Hyperparameter	Value
PINNs	hidden layers	4
	hidden size	512
QRes	hidden layers	4
	hidden size	256
FLS	hidden layers	4
	hidden size	512
PITs	set size	4
	# of encoder	1
	# of decoder	1
	embedding size	32
	head	2
	hidden size	512

Table 2. Hyperparameters for the different models evaluated in the study, including the number of hidden layers, hidden sizes, and additional parameters for PITs.

Training hyperparameters. The training process for the models utilized specific hyperparameters as listed in Table 4. The optimization involved a combination of the Adam optimizer and the L-BFGS optimizer, with a set number of iterations for each. Additionally, the line search function used in the L-BFGS optimization was the strong Wolfe condition. Parameters λ_f and λ_u were set to specific values to control the weighting of residual points and boundary points, respectively. These parameters were kept consistent across all models for a fair comparison.

[4] https://github.com/AdityaLab/pinnsformer

Model	Total trainable parameters
PINNs	527K
FLS	527K
QRes	397K
PITs	454K

Table 3. Total number of trainable parameters for all models. For a fair comparison, all models have relatively similar numbers of trainable parameters.

These training parameters were chosen to ensure efficient and effective convergence of all models during training.

Hyperparameter	Value
Adam Iterations	100
L-BFGS Iterations	1000
L-BFGS line search function	strong wolfe
λ_f	1
λ_u	1
train:val:test	50:21:101

Table 4. Training hyperparameters used for the models, including the number of iterations for the Adam and L-BFGS optimizers, the line search function, and the weighting parameters λ_f and λ_u. These parameters were kept consistent across all models to ensure fair comparison and effective convergence.

A.2 Compute

All models are implemented in PyTorch, and are trained separately on single NVIDIA Tesla V100 GPU. In general, with hyperparameters mentioned in Appendix A.1 the runtime for one run using PINNs, FLS, QRes, PINNsFormer, and PITs were: 103, 89, 149, 308, and 215 seconds respectively. The memory utilization of GPUs are provided in Table 5.

Compute	Value
PITs GPU Memory (set size = 4)	1.8MiB
PINNs GPU Memory	2.1MiB
FLS GPU Memory	2.1MiB
QRes GPU Memory	1.6MiB

Table 5. Approximate memory utilization by each model.

A.3 Training and Evaluation

Training Algorithm. The training algorithm for PITs, outlined in Algorithm 1, involves initializing the model parameters for the Set Generator, Embedding Network, Transformer Network, and Output Layer. During each training iteration, the algorithm processes each element in the discretized domain by sampling points to generate a set, transforming this set into a high-dimensional space using the Embedding Network, and then processing it with the Transformer Network. The transformed set is used by the Output Layer to predict solutions. The set-wise physics loss, defined in Equation 8, is computed to ensure adherence to physical laws. The model parameters are updated iteratively using the Adam optimizer, followed by fine-tuning with the L-BFGS optimizer, to achieve efficient and accurate convergence. This approach leverages set-based processing and transformer networks to simultaneously approximate solutions for multiple points in the domain.

Algorithm 1 Training Algorithm for PITs

Require: Sampled N sets from the domain Ω
Require: Set Generator
Require: Embedding Network p_ψ
Require: Transformer Network f_ϕ
Require: Output Layer q_η
Require: Hyperparameters λ_f, λ_u
Initialize model parameters ψ, ϕ, η
for each training iteration **do**
  for each set $\mathcal{P}_l$ **do**
    Transform $\mathcal{P}_l$ into high-dimensional space $\mathcal{E}_l$ using Embedding Network p_ψ
    Process set $\mathcal{E}_l$ and obtain $\mathcal{O}_l$ using Transformer Network f_ϕ
    Predict solutions u_l using Output Layer q_η
  end for
  Compute set-wise physics loss as defined in Equation 8
  Update model parameters ψ, ϕ, η
end for

Evaluation. The performance of the models was evaluated using two key metrics: Relative Mean Absolute Error (rMAE) and Relative Root Mean Square Error (rRMSE). The rMAE metric, as defined in Equation 9, calculates the mean absolute difference between the predicted and actual values, normalized by the mean of the actual values. This metric provides insight into the average magnitude of prediction errors relative to the actual values. The rRMSE, defined in Equation 10, measures the square root of the mean squared differences between predicted and actual values, normalized by the root mean square of the actual values. Both metrics are essential for assessing the accuracy and robustness of the models' predictions across different datasets and scenarios.

$$\text{rMAE} = \frac{\Sigma_{n=1}^{N} |\hat{y}_n - y_n|}{\Sigma_{n=1}^{N} |y_n|} \tag{9}$$

$$\text{rRMSE} = \sqrt{\frac{\Sigma_{n=1}^{N} |\hat{y}_n - y_n|^2}{\Sigma_{n=1}^{N} |y_n|^2}} \tag{10}$$

B Experiment Setup

B.1 Predicting Activity Coefficients

Activity coefficients are a central thermodynamic property describing the behavior of components in mixtures. They are crucial for modeling and simulating reaction and separation processes, such as distillation, absorption, and liquid-liquid extraction. Each component in a mixture has an individual activity coefficient that depends on the molecular structure of the components, temperature, and concentration in the mixture (typically given in mole fractions ranging from 0 to 1).

Measuring activity coefficients is time-consuming and expensive, resulting in scarce experimental data. Consequently, prediction methods are commonly used, often based on physical theories. Compared to machine-learning models, physical models have the advantage of complying with thermodynamic consistency criteria, such as the Gibbs-Duhem equation, which relates the activity coefficients within a mixture. For binary mixtures at constant temperature and pressure, the Gibbs-Duhem equation is:

$$x_1 \left(\frac{\partial \ln \gamma_1}{\partial x_1} \right)_T + (1 - x_1) \left(\frac{\partial \ln \gamma_2}{\partial x_1} \right)_T = 0 \tag{11}$$

where $\ln \gamma_1$ and $\ln \gamma_2$ are the logarithmic activity coefficients of the two components, T is the temperature, and x_1 is the mole fraction of the first component.

The experimental activity coefficient data set was obtained from the Dortmund Data Bank (DDB). Data on activity coefficients at infinite dilution were directly adopted from the DDB. Activity coefficients were also calculated from vapor-liquid equilibrium data from the DDB. During preprocessing, data points labeled as poor quality by the DDB were excluded. Additionally, only components with retrievable SMILES (simplified molecular-input line-entry system) strings using the CAS number (preferred) or component name from the Cactus database were considered. SMILES were then converted into canonical SMILES using RDKit, resulting in the exclusion of some SMILES that could not be converted.

A system is defined as the combination of two components. The train-test split was done system-wise, with 10% of the systems used for the test set. RDKit was used to create molecular descriptors for each component, specifically using a count-based Morgan fingerprint with zero radius and a bit size of 128.

B.2 Predicting Particle Agglomeration

Population balance equations (PBE) are a well-established method for calculating the temporal evolution of particle property distributions $u(x,t)$, with applications in various fields [33]. Generally, PBEs are integro-differential equations requiring numerical solutions. We investigate the one-dimensional case of pure agglomerate breakage, where analytical solutions exist for specific boundary conditions. The PBE is given by:

$$\frac{\partial u(x,t)}{\partial t} = \int_x^\infty f(x,x')r(x')u(x',t)\mathrm{d}x' - r(x)u(x,t), \tag{12}$$

$$\begin{aligned} \text{IC:} \quad & u(x,0) = \delta(x-L) \\ \text{BC:} \quad & r(x) = x \\ & f(x,y) = \frac{2}{y} \end{aligned}$$

Here, the breakage rate r and breakage function f are the so-called kernels and define the system's physical behavior. The analytical solution for this special case is formulated as [45]:

$$u(x,t) = \exp\left(-tx\right)\left(\delta(x-L) + \left[2t + t^2(L-x)\right]\theta(L-x)\right) \tag{13}$$

with δ being the Dirac delta function and

$$\theta(x-L) = \begin{cases} 1, & x < L \\ 0, & \text{otherwise} \end{cases} . \tag{14}$$

In a real-world setting, where only experimental data is available, the kernel values are generally unknown. Although empirical equations exist, they must be calibrated to experiments, making benchmarking solely on experimental data impossible. Therefore, we used the provided special case. However, it should be noted that PINNs have already been applied for solving the inverse problem, i.e., estimating unknown kernels [8]. Improved accuracy on synthetic data will likely correspond to higher accuracy in the inverse problem when applied to experimental data.